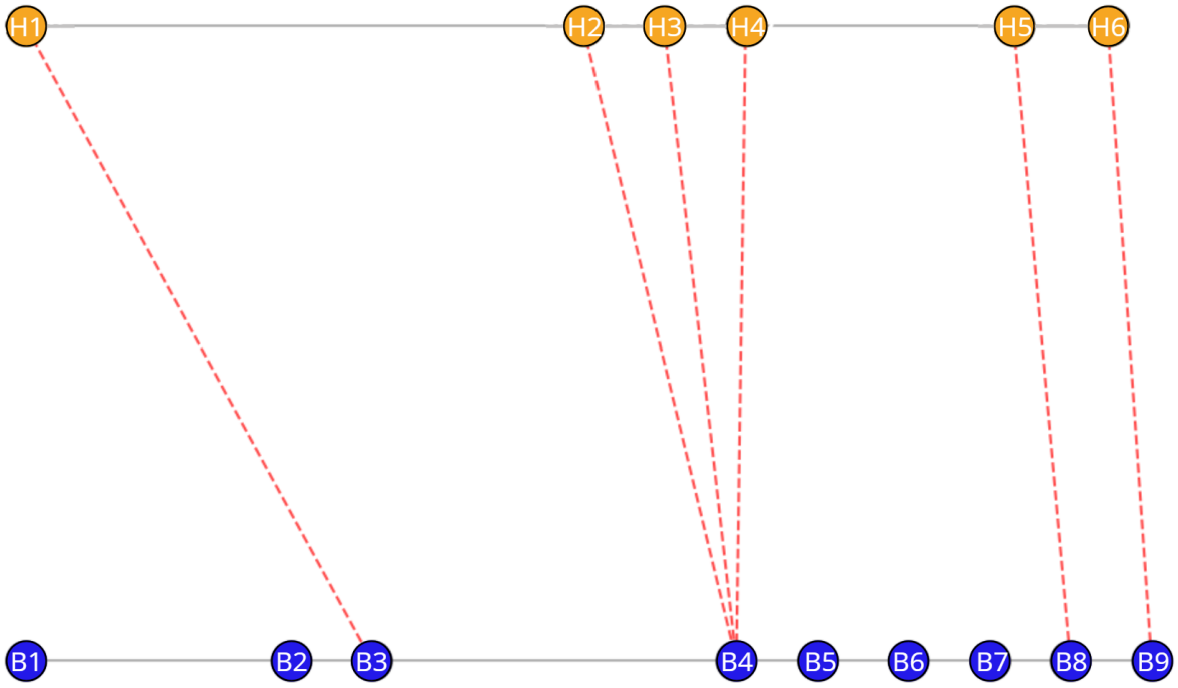
CHALMERS
UNIVERSITY OF TECHNOLOGY



Graph-Based Alignment of Heterogeneous Tracking Events

A Process Mining and Structural Time Warping Approach to Canonicalizing Shipment Lifecycles

Master's thesis in Master Engineering mathematics and computational science

ELINA MÖLLER

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Graph-Based Alignment of Heterogeneous Tracking Events

A Process Mining and Structural Time Warping Approach to Canonicalizing Shipment Lifecycles

Elina Möller

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Algebra and Geometry
Chalmers University of Technology
Gothenburg, Sweden 2026

Graph-Based Alignment of Heterogeneous Tracking Events
A Process Mining and Structural Time Warping Approach to Canonicalizing Shipment Lifecycles
ELINA MÖLLER

Supervisors:
Martin Raum, Department of Mathematical Sciences
Mikeal Böörs, Centiro
Daniel Dalevi, Centiro
Examiner: Martin Raum, Department of Mathematical Sciences

Master's Thesis 2026
Department of Mathematical Sciences
Division of Algebra and Geometry
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An alignment between two "golden paths" shipment life-cycles from carrier X_1 and X_2.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Graph-Based Alignment of Heterogeneous Tracking Events
A Process Mining and Structural Time Warping Approach to Canonicalizing Shipment Lifecycles
ELINA MÖLLER
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

In last-mile logistics, tracking shipment progress is obstructed by data heterogeneity. Different companies and services report tracking events using disparate naming conventions, frequencies, and operational sequences. This inconsistency prevents efficient and reliable multi-carrier analysis of shipment lifecycles and the detection of deviant shipment behaviors. To bridge this gap, this thesis introduces a unified pipeline to align and canonicalize heterogeneous tracking events into a standardized shipment lifecycle.

The proposed methodology integrates process mining with structural graph theory. First, an optional pre-processing layer utilizes the HeuristicsMiner algorithm to isolate the main underlying process from noise, outliers, and low-frequency traces. Next, the pipeline uses Weisfeiler-Lehman (WL) subtree kernels to project the resulting directed networks into a vector space. This representation enables K-means clustering to group structurally similar lifecycles. These clusters are then used to extract representative reference models known as "golden paths." Finally, cross-carrier node-to-node event mapping is achieved by applying Structural Dynamic Time Warping (SDTW), followed by a customized heuristic layer.

Empirical validation shows that pre-processing through process mining substantially strengthens cluster stability. Furthermore, alignment fidelity was confirmed utilizing a specialized validation framework, demonstrating that the pipeline successfully bypasses superficial string-naming variations to resolve the true logistical intent of ambiguous cross-carrier event sequences.

Keywords: Process Mining, HeuristicMiner, Weisfeiler-Lehman Graph Kernels, Structural Dynamic Time Warping, Clustering, Canonicalization, Shipment Alignment.

Acknowledgements

I would like to thank my academic supervisor, Prof. Martin Raum, and my Centiro supervisors, Daniel Dalevi and Mikeal Böörs, for their invaluable guidance, support, and mentorship during the course of this project.

Elina Möller, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

DTW	Dynamic Time Warping
PGE	Polar Graph Embedding
PGEd	Polar Graph Embedding distance
SDTW	Structural Dynamic Time Warping
STW	Structural Time Warping
WL	Weisfeiler-Lehman

Nomenclature

Below is the nomenclature of core indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

h	Total number of iterations (height of the subtree kernel)

Sets

Σ	Alphabet of node labels for Weisfeiler-Lehman
W	Optimal warping path $\{(i_1, j_1), \ldots, (i_K, j_K)\}$

Parameters

p, c, m, r	Counters (produced, consumed, missing, remaining) recorded during the token replay of a trace
$\ell(v)$	Initial label function, $\ell : V \to \Sigma$ for WL kernels
$\mathcal{N}(v)$	Neighborhood of node v
f	Hash function used for label compression, $f : \Sigma^* \to \Sigma$
$c_i(G, \sigma_{ij})$	Count parameter. The frequency of label σ_{ij} in graph G at iteration i
$k^{(h)}_{\text{WLsubtree}}$	The resulting kernel between two graphs
(x_c, y_c)	Polar center
P_r, P_ϕ	Divisions for radial and angular segments
P_{bins}	Bins per polar segment
γ	Scaling factor for the bounding circle radius ρ_γ
w	Parameter for sliding window movement

Variables

$fitness(L, N)$	The fitness score of the event log L relative to model N
$l_i(v)$	Refined label of node v at iteration i for WL kernels
$M_i(v)$	Multiset of neighbor labels for node v at iteration i for WL kernels
$s_i(v)$	Concatenated string representing the sorted multiset of neighbors for node v at iteration i for WL kernels
$\phi^{(h)}_{\text{WLsubtree}}(G)$	The feature vector representation of graph G
H	Global histogram vector for a sliding window
C	Cost matrix, where $C_{i,j} = PGEd(H_i, H'_j)$

Contents

List of Acronyms **ix**

Nomenclature **xi**

List of Figures **xvii**

List of Tables **xxi**

1 Introduction **1**
1.1 Problem Statement 2

2 Theory **3**
2.1 Process Discovery and Lifecycle Modeling 3
2.1.1 Event Log Fundamentals 3
2.1.2 Modeling Formalisms 4
2.1.3 The HeuristicMiner Algorithm 4
2.1.3.1 Dependency relations 5
2.1.3.2 Construction of the Dependency Graph 5
2.1.3.3 Handling Loop-Induced Bias in Dependency Graph . 5
2.1.3.4 Threshold values in the HeuristicMiner algorithm . . 6
2.1.3.5 AND/XOR - split/join 6
2.1.4 Fitness Score using Token replay 7
2.1.4.1 Quantifying Fitness 8
2.2 Structural Similarity and Graph Representation 9
2.2.1 The Weisfeiler-Lehman Isomorphism Test 9
2.2.2 Feature Extraction for Heterogeneous Networks 10
2.3 Semantic Alignment via Structural Context 11
2.3.1 Polar Graph Embedding 12
2.3.2 Structural Dynamic Time Warping 14

3 Methods **17**
3.1 Specification of the problems being investigated 17
3.2 Overview of the pipeline and the main objectives 17
3.3 Process Mining 18
3.3.1 Model Faithfulness and Survival Metrics 18
3.3.1.1 Survival Rate 18
3.3.1.2 Filtered Survival Rate 18

3.3.2 Sensitivity Analysis on HeuristicMiner algorithm 19
3.3.2.1 Sensitivity Analysis on Positive Observation Parameters 19
3.3.2.2 Sensitivity Analysis on Dependency threshold 20
3.3.2.3 Sensitivity Analysis on Relative Observation 20
3.3.2.4 Sensitivity Analysis of Clustering by Process Mining Validation 20
3.3.3 Process Mining before Clustering 21
3.3.4 Implementation comment on choice of the first element in HeuristicMiner 21
3.4 Weisfeiler-Lehman Kernels 21
3.4.1 Structural Feature Extraction using Weisfeiler-Lehman Kernels 22
3.5 Structural Dynamic Time Warping and subsequent heuristic 23
3.5.1 Time-Normalized Coordinate Projection 23
3.5.2 Heuristic Refinement for Node-to-Node Mapping 24
3.5.3 Evaluation Metrics for Heterogeneous Event Mapping 25
3.5.4 Parameter Tuning via Grid Search 29

4 Results **31**
4.1 Stability Analysis 31
4.1.1 Positive observation threshold and Positive observation threshold Backbone 31
4.1.2 Dependency threshold 31
4.1.3 Relative Observation threshold 31
4.2 Structural Clustering and Golden Path Discovery 33
4.2.1 Representative Case Study: Cluster Quality of Carrier X_1 . . 33
4.2.2 Multi-Carrier Generalization 34
4.2.3 Impact of Process Pre-processing on Cluster Quality 34
4.2.4 Analysis of Clustering Limitations: The Case of Carrier Z_4 . . 35
4.3 Performance 38
4.3.1 Weight Parameters 38
4.3.2 Examples Alignments between two Carriers 39
4.3.3 Example Alignments between three Carriers 39
4.3.4 Representative Case Study: Alignment between X_2, Z_1 and Z_4 41

5 Conclusion **45**
5.1 Summary of findings and Fulfilment of the Desiderata 45
5.2 Limitations of current pipeline 46
5.3 Further Studies 46

Bibliography **49**

A Appendix 1 **I**
A.1 Appendix regarding Results I
A.1.1 Multi-carrier Generalization: Average Survival Rates with K-Means Clustering I
A.1.2 Performance III

A.1.3 Representative Case Study: Alignment between X_2, Z_1 and Z_4 VI

List of Figures

2.1 Example of a Petri net (left) and corresponding Causal Matrix (right). 4

2.2 Example of a Petri net. Transitions have a label name of a single letter and places are denoted by p in conjunction with a number, except the start and end. 7

2.3 Data processing pipeline executing a single vertex label refinement iteration for v_1, seen in Figure 2.4. 10

2.4 Structural transformation of a network over a single vertex refinement iteration. 12

2.5 Construction of final concatenated feature vector after one single vertex refinement iteration. 13

2.6 Example of a polar graph embedding, of a subgraph, into $n = P_r \times P_\phi$ polar segments ($16 = 2 \cdot 8$.). 14

2.7 Example positions of the 'sliding window' and the corresponding histograms, represented as vectors, for each step.. 14

2.8 Visual intuition of the SDTW cost matrix (right). Darker regions indicate lower structural distance calculated via $\mathrm{PGE}_d(H, H')$. The red line highlighted the optimal warping path. To the left the corresponding graph alignment is shown. 16

3.1 An example of a naive node-to-node alignment between two unlabeled shipment networks where structural dynamic time warping fails due to significant temporal shifts between localized node clusters. 26

3.2 An optimized node-to-node alignment of the networks from Figure 3.1, demonstrating how incorporating structural position and topological contexts corrects the misalignment caused by temporal variance. . . . 26

3.3 Structural alignment between network A and network B mapping Centiro descriptions. Each node displays both its structural identifier (A_1, B_2, etc.) and its assigned Centiro label description. Green edges represent valid alignment pairs according to the Accuracy Score criteria, whereas red edges denote pairs flagged as inaccurate. The illustrated alignment yields an overall Accuracy Score of 0.5. 28

3.4 Structural alignment between network A and network B utilizing abstract-level Centiro descriptions. Each node displays both its structural identifier (A_1, B_2, etc.) and its corresponding abstract description. Green edges represent valid alignment pairs according to the Accuracy Score criteria, while red edges denote flagged mismatches. Due to the generalized mapping, all pairs are validated, resulting in an abstract Accuracy Score of 1.0. 28

4.1 Sensitivity analysis of the Positive Observation parameters evaluated across carriers X_1, X_2, X_3, and X_4. 32
4.2 Sensitivity analysis of the Positive Observation parameters evaluated across carriers X_1, X_2, X_3, and X_4 when Positive Observation Threshold Backbone = 0.1 * Positive Observation Threshold. Illustrating a more stable model in comparison to Figure 4.1. 32
4.3 Sensitivity analysis of the Dependency Threshold parameter evaluated across carriers X_1, X_2, X_3, X_4, Z_1, Z_3 and Z_4 illustrating a distinct drop-off in Strict Survival Rate as the threshold exceeds 0.8. . 33
4.4 Sensitivity analysis of the Relative Observation Threshold parameter evaluated across carriers X_1, X_2, X_3, X_4, Z_1, Z_3 and Z_4 illustrating a stable process models after a value of 0.2. 34
4.5 K-means Elbow for Carrier X_1 35
4.6 Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier X_1. The distinct plateau observed between $K = 5$ and $K = 6$ validates the optimal cluster selection. Filtering threshold was set at 0.6. 36
4.7 Initial K-means elbow for carrier X_4. 37
4.8 K-means elbow for carrier X_4 after Process Mining Pro-processing. . 37
4.9 K-means elbow for carrier Z_4 37
4.10 Node-to-node Alignment between carriers X_2 and Z_4. 42
4.11 Node-to-node Alignment between carriers Z_4 and Z_1. 43

A.1 Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier X_2. The distinct plateau observed between $K = 5$ and $K = 6$ validates the optimal cluster selection. Filtering threshold was set at 0.6. I
A.2 K-Means elbow for carrier X_2. II
A.3 Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier Z_1. The distinct plateau observed between $K = 5$ and $K = 6$ validates the optimal cluster selection. Filtering threshold was set at 0.6. II
A.4 K-Means elbow for carrier X_1. III
A.5 Alignment between Z_1, X_2 and Z_4 maximizing their total Accuracy Score and Transitive Consistency. IV
A.6 Alignment between Z_1 and X_1. The alignment is the alignment with highest Accuracy Score for all aligned 'golden paths' between those 2 carriers. V

A.7 Alignment between Z_1 and X_2. The alignment is the alignment with highest Accuracy Score for all aligned 'golden paths' between those 2 carriers. V
A.8 Alignment between X_2 and Z_4. The alignment is the alignment with highest Accuracy Score for all aligned 'golden paths' between those 2 carriers. VI
A.9 Alignment between Carriers X_2, Z_1 and Z_4. VII

List of Tables

4.1 Optimal K elbow versus Survival Plateau for different carriers 35
4.2 Average Survival Rate of clusters (second rate excludes the lowest value) before and after pre processing the networks using Process Mining. 35
4.3 Process mining on clusters for carrier Z_4 after initial process mining . 38
4.4 Carrier Accuracy Comparison with and without weight parameters. The weight parameters used were $w_A = 75$, $w_t = 100$, $w_T = 0.01$ and $w_f = 10$. 38
4.5 Carrier Accuracy Comparison. Parameter weights used were $w_A = 75$, $w_t = 100$, $w_T = 0.01$ and $w_f = 10$. 39
4.6 Alignments between carriers X_1, X_2 and X_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$ 40
4.7 Alignments between carriers X_1, Z_1 and Z_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$ 40
4.8 Alignments between carriers X_3, Z_1 and Z_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$ 40
4.9 Alignments between carriers X_2, Z_1 and Z_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$ 41
4.10 Successfully Paired nodes for alignment $X_2 - Z_4$.. References alignment in Figure 4.10. 42
4.11 Flagged node alignments for alignment $X_2 - Z_4$. References alignment in Figure 4.10. 42
4.12 Successfully Paired Labels using the abstract descriptions. References alignment in Figure 4.10. Crucially, "Dropped off at Service" are not attributed to any abstract description. 43
4.13 Flagged Labels using the abstract descriptions. References alignment in Figure 4.10.Crucially, "Dropped off at Service" are not attributed to any abstract description. 43
4.14 Successfully Paired nodes for alignment $Z_4 - Z_1$. References alignment in Figure 4.11. 44
4.15 Flagged node alignments for alignment $Z_4 - Z_1$. References alignment in Figure 4.11. 44

A.1 Successfully Paired Labels. References alignment in Figure 4.11. . . . VI
A.2 Flagged Labels. References alignment in Figure 4.11. VI
A.3 Carried over. References alignment in Figure A.9. VIII

A.4 Carried over. References alignment in Figure A.9. VIII

1
Introduction

Centiro, a technology company specializing in networks, empowers global market leaders and enterprise-level organizations by centralizing and optimizing delivery data from a vast array of logistics providers. In last-mile logistics, carriers report shipment progress through event updates such as "Picked up," "In transit," or "Delivered". However, because each carrier operates using its own internal logic, these updates vary significantly in naming, frequency, and sequence.

This inconsistency creates a major barrier for brands: an event labeled "Hub Arrival" by one carrier may be functionally identical to "Sorted at Facility" by another, yet without a unified mapping, systematic analysis is impossible. This thesis utilizes Process Mining to bridge this gap. By extracting knowledge from event logs, process mining allows us to discover the relationship between observed behavior and formal process models. By combining these techniques with structural graph kernels and alignment algorithms, this project seeks to infer the meaning of unspecified or heterogeneous event labels by mapping them to a standardized shipment lifecycle.

The key contributions of this Master's thesis are the adaption and expansion of established methodologies to unstructured shipment cycles. The thesis provides the following advancements:

- Several standard algorithms operating on conflicting data types, including HeuristicMiner, Weisfeiler-Lehman graph kernels and Structural Dynamic Time Warping (SDTW), are successfully integrated into a single, unified pipeline. Furthermore, novel structural concepts, such as the Positive Observation Backbone, are introduced to better adapt these mathematical algorithms to the constraints of logistical tracking data.
- To evaluate model efficacy and alignment accuracy in the absence of absolute ground truth, a specialized validation framework was developed. The utilization of Survival Rate metrics and the introduction of a Transitive Consistency Score provide necessary quantitative measures to verify global alignment integrity.
- This thesis demonstrates that Weisfeiler-Lehman graph kernels together with classical clustering methods effectively group structurally similar shipment cycles. Furthermore, by integrating process mining techniques, a robust structural filter is established. This filter is capable of isolating high-quality shipment behaviors.

1.1 Problem Statement

The goal of this thesis is to develop an algorithm pipeline for cross-carrier event sequence alignment and canonicalization, mapping heterogeneous tracking events into a unified shipment lifecycle. The project will use this to flag deviant behaviors. The main objectives of the project are thus to produce the following deliverables:

- Utilizing process mining and clustering to identify the "canonical" routes for a given carrier, serving as the reference for alignment.
- Designing an alignment method that can handle reordered events and directional graphs (with cycles) to find correspondences between different carrier datasets.
- Quantifying how far a specific shipment sequence deviates from the inferred canonical order, using metrics like Survival Rate, to surface problematic shipments systematically.
- Developing a method to map raw, often ambiguous event names into a set of standardized phrases by analyzing their structural context within a shipment lifecycle.

2
Theory

In this chapter necessary theory is presented. This includes Process Mining, specifically an Heuristic algorithm, Weisfeiler-Lehman kernels, and Structural Dynamic Time Warping.

2.1 Process Discovery and Lifecycle Modeling

The primary objective of process mining is to extract knowledge from event logs by bridging the gap between data science and process science. Process mining introduces a process perspective while building on other areas such as data mining, which focuses on finding hidden patterns and correlations within large datasets, and machine learning, which emphasizes the algorithmic improvement of programs through experience. This perspective concerns the explicit relationship between observed event data and formal process models [1, pp. 12–15].

In this thesis, process mining provides the contextual mapping necessary to decode ambiguous event labels. By viewing carrier updates as "observed behavior" within a shipment lifecycle, we can utilize the underlying process logic to decode ambiguous data. This allows the algorithm to look beyond the literal string of an event label and instead identify its functional identity based on its position within a discovered causal model. These models serve as the foundation for defining "golden paths", which is the reference sequences used to align heterogeneous carrier data.

2.1.1 Event Log Fundamentals

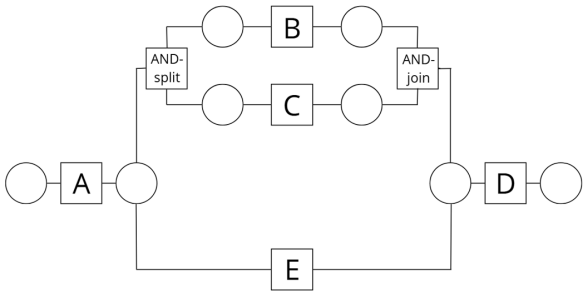
Process mining extracts underlying process information from event logs, which serve as the raw input for model discovery. An event log must contain three mandatory attributes for every recorded event:

- A unique case identifier that groups events into a single process instance. In the context of this thesis, this corresponds to a specific shipment ID.
- A description of the task or event being performed as an activity label. Examples from this thesis are "Picked up" or "In transit."
- Recorded date and time, as a timestamp, that establishes the order of events.

Event logs can contain additional features, such as geographical coordinates or carrier metadata, but these three attributes are the minimum data requirement to represent the log as a structured table for algorithmic processing [5, pp. 1–2].

2.1.2 Modeling Formalisms

The mined relationships and dependencies discovered during process mining can be expressed through various representations. A Petri net is a conventional, bipartite graph representation consisting of places (P) and transitions (T), connected by directed arcs known as the flow relation (F). The behavior of the network is governed by the movement of tokens, which "fire" through transitions to simulate process flow. Petri nets are mathematically rigorous and well-suited for fitness analysis via token replay. However, they often struggle with the "noise" found in real-world logistics data. Attempting to capture every observed behavior in a binary Petri net can lead to a "spaghetti" model; a complex, unreadable structure where meaningful patterns are obscured by infrequent or exceptional traces [1, pp. 59–60]. To address these limitations, this thesis utilizes the Causal Matrix, which is the standard output for heuristic mining algorithms. A Causal Matrix represents each activity with an input expression and an output expression, using logical operators—such as AND ($\wedge$) and OR ($\vee$) to define dependencies. For example: A start activity is defined by an empty input expression ($\emptyset$). An end activity is defined by an empty output expression ($\emptyset$). Intermediate activities use logical expressions to dictate whether preceding or succeeding tasks are required or optional. Unlike Petri nets, Causal Matrices are based on dependency scores, allowing them to focus on the "logistical core" of the process while naturally filtering out stochastic noise [5, p. 10]. While it is theoretically possible to transform a Causal Matrix back into a Petri net, such a transformation is often avoided in practical logistics applications because it reintroduces structural complexity that may hinder human interpretability.



Activity	Input	Output
A	Ø	(BvE) ∧(CvE)
B	A	D
C	A	D
D	(BvE) ∧ (CvE)	Ø
E	A	D

Figure 2.1: Example of a Petri net (left) and corresponding Causal Matrix (right).

2.1.3 The HeuristicMiner Algorithm

The HeuristicMiner algorithm is designed to handle real-world event logs that contain noise, outliers and low frequency behavior. Unlike purely formal discovery methods, it incorporates frequencies of traces to differentiate between the "main" process flow and stochastic outliers [5, p. 3]. This makes it a suitable method for identifying the typical shipment cycles required for cross-carrier alignment.

2.1.3.1 Dependency relations

The algorithm begins by establishing the basic relationships between pairs of activities (a, b) within the event log W. An event log is a multiset of traces σ. Let $a, b \in T$. These relations are based on the order in which they appear in the traces:

- $a >_W b$ iff there is a trace $\sigma = t_1 t_2 t_3 ... t_n$ and $i \in \{1, ..., n-1\}$ such that $\sigma \in W$ and $t_i = a$ and $t_{i+1} = b$. This is called a direct successor.
- $a \rightarrow_W b$ iff $a >_W b$ and $b \not>_W a$. $a \rightarrow_W b$ therefore denotes a causal relation.
- $a >>_W b$ iff there is a trace $\sigma = t_1 t_2 t_3 ... t_n$ and $i \in \{1, ..., n-2\}$ such that $\sigma \in W$ and $t_i = a$ $t_{i+1} = b$ and $t_{i+2} = a$. This is a length-two-loop.

Let us clarify some of the notation above since the concepts are easier to capture using words. The first one, $a >_W b$, means that activity b follows a directly at least once in the event log. The third one, $a >>_W b$, denotes that the sequence aba is present [5, pp. 5–7].

2.1.3.2 Construction of the Dependency Graph

Now that we have notation for explaining different types of dependencies we can move on to construction of the dependency graph. For each pair of activities, a frequency-based metric is calculated that indicates the certainty that there is a relationship between that pair. These values will be stored in the dependency graph. A dependency relation between two events A and B is denoted by $A \Rightarrow_W B$. It is calculated as follows. Let W be an event log over T, and $a, b \in T$. Define $|a >_W b|$ as the number of times $a >_W b$ appears in W. Then,

$$a \Rightarrow_W b = \left(\frac{|a >_W b| - |b >_W a|}{|a >_W b| + |b >_W a| + 1} \right). \tag{2.1}$$

We conclude that $a \Rightarrow_W b$ is a value between -1 and 1. Any value close to 1 indicates that there is a high chance of dependency for that pair of events and any value close to -1 indicates a high chance of dependency in the opposite direction. A value close to 0 tells us there is a very low chance of dependency for that pair of activities [5, p. 7].

2.1.3.3 Handling Loop-Induced Bias in Dependency Graph

While the standard formula is effective for linear processes, it produces misleading results in the presence of repetitive behavior. There are two main instances where repetitive behavior poses a problem. First, when an activity is repeated immediately (e.g., AAA) the standard formula would see identical counts for $a > a$ in both directions of the fraction. This would lead to an artificially low score. This is accounted for by:

$$a \Rightarrow_W a = \left(\frac{|a >_W a|}{|a >_W a| + 1} \right).$$

Second, the initial dependency calculation (2.1) fails for structures with repeated length-two-loops (e.g., $ACBCBCBD$). In the standard formula the dependency scores can cancel out. The differentiation of these instances from noise is accounted by:

$$a \Rightarrow_{2W} b = \left(\frac{|a >>_W b| - |b >>_W a|}{|a >>_W b| + |b >>_W a| + 1} \right).$$

is used instead [5, p. 9].

2.1.3.4 Threshold values in the HeuristicMiner algorithm

To ensure the discovered model is structurally sound and representative of a complete process, the algorithm utilizes the all-connected heuristic. This heuristic is performed on the dependency matrix, containing all pairwise scores, to establish a continuous "backbone" for the model. The process begins by identifying the initial activity, defined as the column in the dependency matrix with no positive values. The algorithm then performs a "traversal" through the matrix by selecting the highest dependency value in the current activity's row to identify the next successor. This sequence continues until a terminal node, a row with no positive values, is reached. Once this high-dependency backbone is established, all remaining dependencies are evaluated against three primary thresholds to filter out logistical noise and infrequent exceptions [5, p. 8]:

- Dependency threshold: Only connections with dependency above the set limit are considered.
- Positive Observation Threshold: Requires that a connection must have been observed with a minimal frequency to be considered.
- Relative to best threshold: Includes connections if their dependency difference in comparison to the highest dependency value is small enough.

An additional threshold is utilized during the discovery of AND/XOR splits, which dictates the logical complexity of the final causal matrix.

2.1.3.5 AND/XOR - split/join

Apart from the deficiencies a mining must handle parallel (AND) and exclusive (XOR) relationships. This distinction is based on the observation that if two activities, b and c, are in an AND relationship, they may follow each other in any order within the event log. Conversely, if they are in an XOR relationship, such a pattern is statistically improbable. Let W be an event log over T, $a, b, c \in T$, and b and c are depending relation with a. Then following measurement can be formulated,

$$a \Rightarrow_W b \vee c = \left(\frac{|b >_W c| + |c >_W b|}{|a >_W b| + |a >_W c| + 1} \right). \tag{2.2}$$

The numerator represents the number of times b and c follow each other directly, indicating potential parallelism. The resulting normalized ratio score represents the relative frequency of parallel behavior and is compared against a specific threshold.

a and b are considered to be in an (AND) relationship if $a \Rightarrow_W b \wedge c$ score is above the threshold. If the score is below the threshold, they are treated as mutually exclusive, therefore in a (XOR) relationship. This pairwise evaluation effectively identifies standard branching. However, real-world logistics data can occasionally present more complex synchronization patterns. The purpose of process mining in this thesis makes it sufficient to only consider the presented relationships [5, p. 10–12].

2.1.4 Fitness Score using Token replay

Token replay is a technique used to quantify the fitness of a process model by replaying the traces from an event log back through the model. For a process modeled as a Petri net, tokens are "fired" through transitions according to the sequences in the event log. As the trace is replayed, the algorithm tracks four specific counters to measure how well the model explains the observed behavior: produced tokens (p), consumed tokens (c), missing tokens (m) and remaining tokens (r). The mechanics of the algorithm are best illustrated through an example using the Petri net in Figure 2.1 and the trace $\sigma_1 = \langle a, d, e \rangle$.

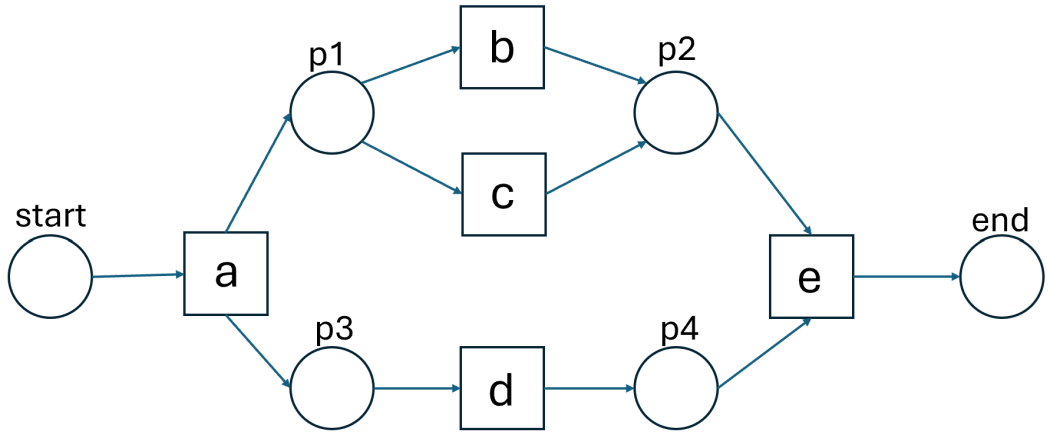


Figure 2.2: Example of a Petri net. Transitions have a label name of a single letter and places are denoted by p in conjunction with a number, except the start and end.

Initially, $p = c = 0$ and no tokens are present in the Petri net. To start the process, the initial place called start produces a token; therefore, p is increased to 1. Then, it is time to replay the trace $\sigma_1 = \langle a, d, e \rangle$. As seen by the trace, the first transition that needs to be fired is a. This is permitted since there is a token in the place before transition a. Following a are two places called place 1 and place 3. Therefore, firing transition a consumes one token, the one in place start, and produces two tokens: one in place 1 and one in place 3. Consequently, the new counts are $p = 1 + 2 = 3$

and $c = 1$. The next transition to be fired is d. Since there is a token in the place before that transition, it is enabled, resulting in $p = 3 + 1 = 4$ and $c = 1 + 1 = 2$. Then, the last transition e is fired, leading to $p = 4 + 1 = 5$ and $c = 2 + 2 = 4$. Consumed tokens are increased twice in that step because transition e has two incoming arcs from two separate places, place 2 and place 4, both of which contain tokens. Lastly, the token in the end place is consumed, and c is incremented to 5. Note that a token remains in place 1 even though the full trace has been replayed. Therefore, $r = 1$. Similarly, if one were to fire a transition without a token in the preceding place, it would indicate a missing token, and m would be increased.

Before moving on to quantify these tokens into a fitness score, consider a trace that can be replayed fully: $\langle a, b, d, e\rangle$. The environment creating a token in the start place, increases p to 1. Firing a results in $p = 1+2 = 3$ and $c = 1$. Firing b consumes the token in place 1 and produces one in place 3, resulting in $p = 3 + 1 = 4$ and $c = 1 + 1 = 2$. Then, d is fired, and $p = 4 + 1 = 5$ and $c = 2 + 1 = 3$. Currently, there is one token in place 3 and one in place 4. When the last transition is fired and the environment consumes the produced token in the end place, it results in $p = 5 + 1 = 6$ and $c = 3 + 3 = 6$. Furthermore, in this case, $m = r = 0$.

2.1.4.1 Quantifying Fitness

Let us now move on to the actual fitness score derived from this token replay. The fitness of a trace σ on a Petri net N is defined as:

$$fitness(\sigma, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right). \tag{2.3}$$

We can therefore conclude that $0 \leq fitness(\sigma, N) \leq 1$. Let us calculate the fitness for our two examples $\sigma_1 = \langle a, d, e\rangle$ and $\sigma_2 = \langle a, b, d, e\rangle$. For σ_1, which could not be replayed fully and resulted in $c = p = 5$, $r = 1$, and $m = 0$, we get:

$$fitness(\sigma_1, N) = \frac{1}{2}\left(1 - \frac{0}{5}\right) + \frac{1}{2}\left(1 - \frac{1}{5}\right) = \frac{9}{10} = 0.9.$$

Note that the fully replayable trace σ_2 yields, as expected, the highest possible fitness score [1, pp. 246–250]:

$$fitness(\sigma_2, N) = \frac{1}{2}\left(1 - \frac{0}{6}\right) + \frac{1}{2}\left(1 - \frac{0}{6}\right) = 1.$$

Equation (2.3) determines the fitness of a single trace. This can be generalized into a fitness measure for an entire event log. Let $p_{N,\sigma}$ be the number of produced tokens when replaying σ on N, with $c_{N,\sigma}$, $m_{N,\sigma}$, and $r_{N,\sigma}$ defined similarly. The measure $fitness(L, N)$ quantifies how many events can be replayed, focusing on tokens rather than individual traces. The fitness of event log L on N is thus defined as:

$$fitness(L, N) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma\in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma\in L} L(\sigma) \times c_{N,\sigma}}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma\in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma\in L} L(\sigma) \times p_{N,\sigma}}\right). \tag{2.4}$$

Observe that $\sum_{\sigma\in L} L(\sigma) \times m_{N,\sigma}$ represents the total number of missing tokens when replaying the entire event log. This fitness measure can be interpreted in two ways.

Firstly, if an event log has a fitness of P, then $1 - P$ events deviate. Secondly, if the process model has a fitness P, then $1 - P$ events cannot be explained by this model [1, pp. 253–254].

2.2 Structural Similarity and Graph Representation

When working with real-world tracking networks, a strict abstract isomorphism test is often too restrictive. This limitation is solved by incorporating graph kernels that project network topologies into a shared vector space based on continuous structural similarity. In this thesis, the Weisfeiler-Lehman subtree kernel is utilized.

2.2.1 The Weisfeiler-Lehman Isomorphism Test

Graph kernels provide a method for graph classification and for quantifying similarity between different structures. Before discussing the Weisfeiler-Lehman (WL) subtree kernel, we first define a graph by the triplet (V, E, ℓ), where V is the set of nodes, E is the set of edges, and $\ell : V \to \Sigma$ is a function that assigns labels from the alphabet Σ to the nodes. The neighborhood of a node v is denoted by $\mathcal{N}(v)$. The WL subtree kernel is based on the Weisfeiler-Lehman isomorphism test, also known as naive vertex refinement. This test investigates whether two graphs, G and G', are isomorphic by iteratively augmenting node labels with the sorted set of their neighbors' labels. These augmented labels are then compressed into new, shorter labels. This process repeats until the label sets between the two graphs diverge or a preset number of iterations is reached. If the label sets are not identical at any iteration i, the algorithm terminates and determines the graphs are not isomorphic [3, pp. 2541–2544].

Algorithm 1 One iteration of the Weisfeiler-Lehman graph isomorphism test

Multiset-label determination

- For $i = 0$, set $M_0(v) := \ell(v)$.
- For $i > 0$, assign a multiset-label $M_i(v)$ to each node v in G and G' which consists of the multiset $\{l_{i-1}(u) | u \in \mathcal{N}(v)\}$.

Sorting each multiset

- Sort elements in $M_i(v)$ in ascending order and concatenate them into a string $s_i(v)$.
- Add $l_{i-1}(v)$ as a prefix to $s_i(v)$.

Label compression

- Sort all of the strings $s_i(v)$ for all v from G and G' in ascending order.
- Map each string $s_i(v)$ to a compressed label using a hash function $f : \Sigma^* \to \Sigma$ that is locally injective for all strings occurring in G and G', such that $f(s_i(v)) = f(s_i(w))$ if and only if $s_i(v) = s_i(w)$.

Relabeling

- Set $l_i(v) := f(s_i(v))$ for all nodes in G.

Figure 2.3 provides a concrete walk-through of a single refinement iteration of the operations formalized in Algorithm 1 for vertex v_1 (on the network given in Figure 2.4). As illustrated, the process collects the downstream neighbors into a standardized multiset $M_1(v_1)$, executes alphanumeric sorting, concatenates the multiset into a string $s_1(v_1)$ with the initial label as a prefix, and uses a locally injective hash function f that maps that string to a new label as a compressed integer.

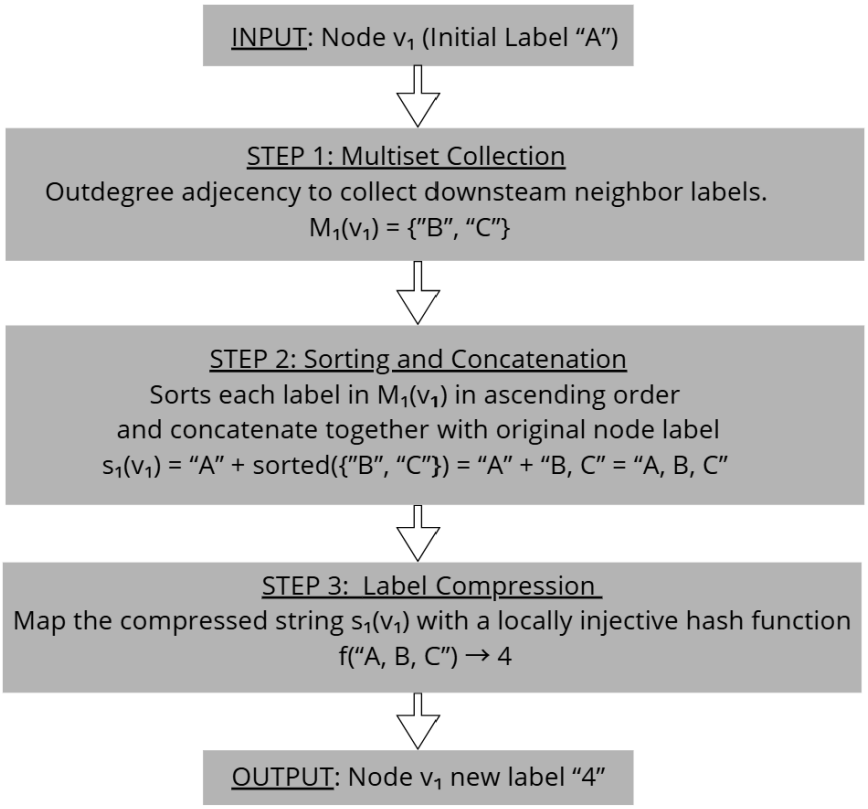


Figure 2.3: Data processing pipeline executing a single vertex label refinement iteration for v_1, seen in Figure 2.4.

2.2.2 Feature Extraction for Heterogeneous Networks

The isomorphism test is a binary check. The Weisfeiler-Lehman Subtree Kernel bridges the gap by creating a similarity measure for multiple graphs.

Definition 2.2.1 (Weisfeiler-Lehman Subtree Kernel)**.** Let G and G' be graphs. Define $\sum_i \subseteq \sum$ as the set of letters that occur as a node labels at least once in G and G' at the end of the i-th iteration of the Weisfeiler-Lehman algorithm. Let $\sum_0$ be the set of original node labels of G and G'. Assume all $\sum_i$ are pairwise disjoint. Without loss of generality, assume that every $\sum_i = \{\sigma_{i1}, ..., \sigma_{i|\sum_i|}\}$ is ordered. Define a map $c_i : \{G, G'\} \times \sum_i \to \mathbb{N}$ such that $c_i(G, \sigma_{ij})$ is the number of occurrences of the letter σ_{ij} in the graph G. The Weisfeiler-Lehman subtree kernel

on two graphs G and G' with h iterations is defined as:

$$k^{(h)}_{\text{WLsubtree}}(G, G') = \langle \phi^{(h)}_{\text{WLsubtree}}(G), \phi^{(h)}_{\text{WLsubtree}}(G') \rangle,$$

where

$$\phi^{(h)}_{\text{WLsubtree}}(G) = (c_0(G, \sigma_{01}), ..., c_0(G, \sigma_{0|\sum_0|}), ..., c_h(G, \sigma_{h1}), ..., c_h(G, \sigma_{h|\sum_h|})),$$

and

$$\phi^{(h)}_{\text{WLsubtree}}(G') = (c_0(G', \sigma_{01}), ..., c_0(G', \sigma_{0|\sum_0|}), ..., c_h(G', \sigma_{h1}), ..., c_h(G', \sigma_{h|\sum_h|})).$$

In simpler terms, this kernel represents each graph as a vector of label counts. By counting unique structural "signatures" created through relabeling, we can represent heterogeneous carrier networks in a fixed-dimensional space [3, pp. 2546–2547]. This enables the use of standard clustering algorithms like K-means, as the structural identity of the shipment lifecycle is now captured in a numerical vector.

Algorithm 2 One iteration of the Weisfeiler Lehman subtree kernel computation on N graphs

Multiset-label determination

- Assign a multiset-label $M_i(v)$ to each node v in G which consists of the multiset $\{l_{i-1}(u) | u \in \mathcal{N}(v)\}$.

Sorting each multiset

- Sort elements in $M_i(v)$ in ascending order and concatenate them into a string $s_i(v)$.
- Add $l_{i-1}(v)$ as a prefix to $s_i(v)$.

Label compression

- Map each string $s_i(v)$ to a compressed label using a hash function $f : \Sigma^* \to \Sigma$ that is locally injective for all strings occurring in G and G', such that $f(s_i(v)) = f(s_i(w))$ if and only if $s_i(v) = s_i(w)$.

Relabeling

- Set $l_i(v) := f(s_i(v))$ for all nodes in G.

Figure 2.4 clarifies the 'macro-level' structural transformation of a network as seen in Algorithm 2. The figure illustrates the immediate results for the first ($i = 1$) simultaneous node refinements enabled by Algorithm 1. Finally, Figure 2.5 demonstrates how these independent node label occurrences across both iterations are globally counted and concatenated to form the final numerical feature vector $\phi(G)$ used by the subtree kernel.

2.3 Semantic Alignment via Structural Context

Before applying Structural Dynamic Time Warping to the shipment networks, one must convert these non-linear topological networks into a sequential and comparable matrix format. This conversion is achieved by mapping neighborhood subgraphs into structural coordinate spaces, after which dynamic sequence operations are utilized.

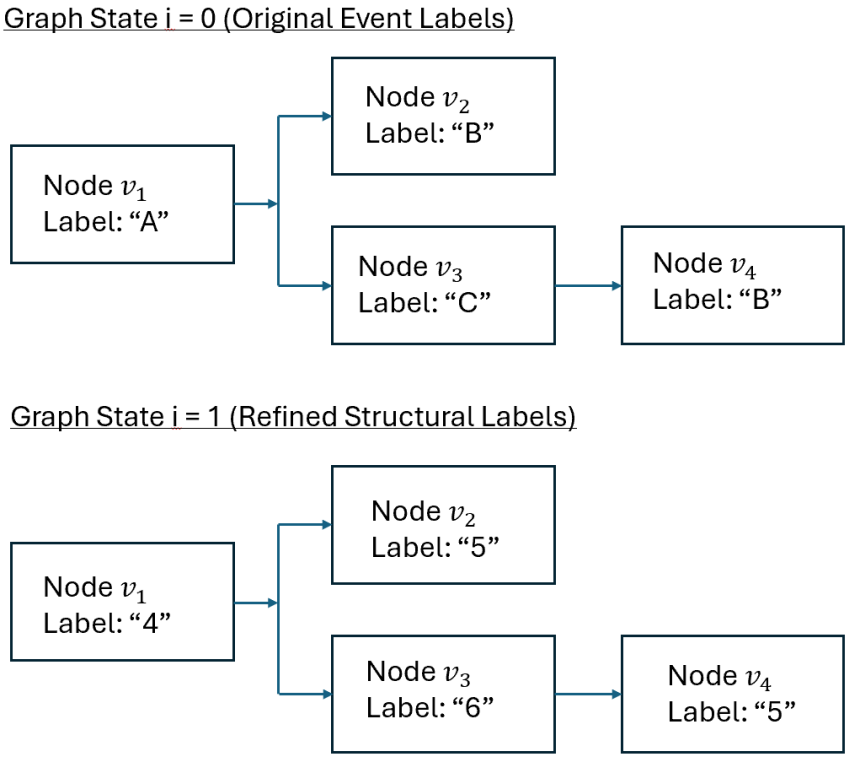


Figure 2.4: Structural transformation of a network over a single vertex refinement iteration.

2.3.1 Polar Graph Embedding

Structural Dynamic Time Warping (SDTW) is a graph matching algorithm that evaluates similarity from multiple local perspectives. The paradigm builds upon two core concepts: Polar Graph Embedding (PGE) and Dynamic Time Warping (DTW). Two graphs are matched using a sliding window; for each pair of window positions, subgraphs are compared using the Polar Graph Embedding distance (PGEd), which calculates dissimilarity between two local graph embeddings. Once a distance matrix is obtained for all window positions, the subgraphs are optimally aligned using DTW. To apply this spatial framework to the graph structures defined in Section 2.2.1, we explicitly assume that each vertex $v \in V$ is embedded in a two-dimensional space. Specifically, every node possesses spatial coordinates (x, y), where x represents the temporal position and y represents a fixed vertical anchor. The first step of the PGE process is Polar Graph Segmentation, where graphs are transformed into a polar coordinate system. Let (x_c, y_c) be the polar center, where x_c is the window position and y_c is the center of mass of the complete graph. A node v with coordinates (x, y) is transformed into polar coordinates (ρ, θ) as follows:

$$\rho = \sqrt{(x - x_c)^2 + (y - y_c)^2},$$
$$\theta = \text{atan2}(x - x_c, y - y_c).$$

Then ρ is the distance from the polar center to the node's original position and θ

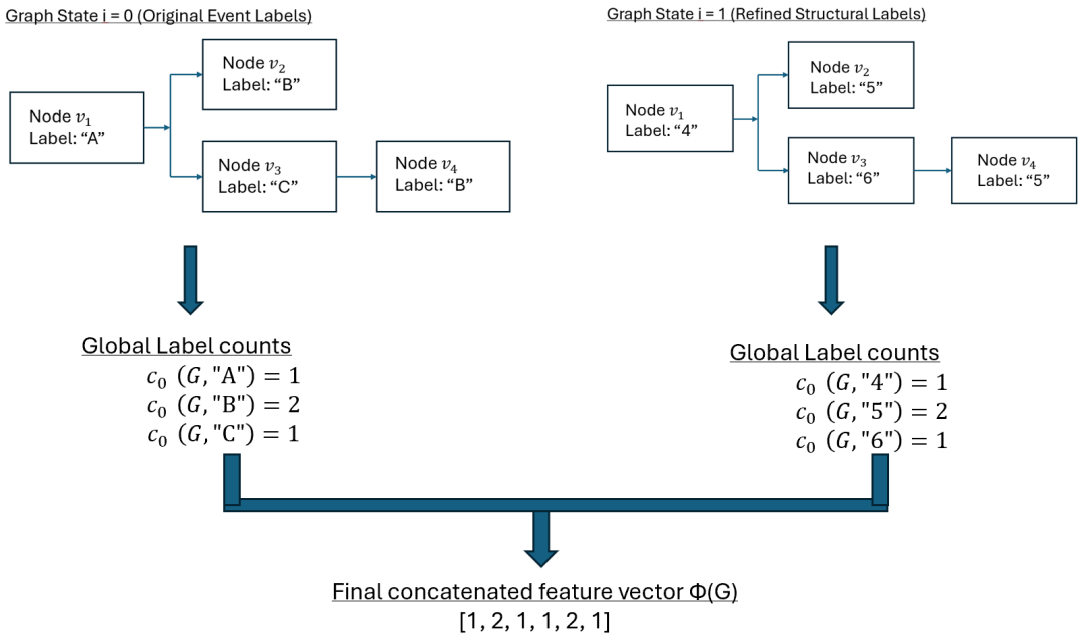


Figure 2.5: Construction of final concatenated feature vector after one single vertex refinement iteration.

similarly is the angle between the x-axis and the node's original position. Let us now define the already mentioned bounding circle C. Let its radius be ρ_γ and formally:

$$\rho_\gamma = \gamma \cdot \rho_{\max}.$$

ρ_{max} is the maximum distance between a polar center and all nodes in a graph. The parameter $\gamma \in [0, 1)$ is defined manually. Now, the bounding circle C is divided into n polar segments. These polar segments represent divisions of the circular window. The bounding circle C is divided is divided into $n = P_\phi \times P_r$ segments, where P_ϕ and P_r are the number of different angles and radii, respectively. Every polar segment b_k is defined by $\rho_{k,min}$, $\rho_{k,max}$, $\theta_{k,min}$, and $\theta_{k,max}$. These are computed using the segmentation indices $i \in \{1, \ldots, P_r\}$ and $j \in \{1, \ldots, P_\phi\}$ as follows:

$$\rho_{kmin} = \frac{\rho_\gamma}{P_r}(i-1),\ \rho_{kmax} = \frac{\rho_\gamma}{P_r}i,$$
$$\theta_{kmin} = \frac{360}{P_\phi}(j-1),\ \theta_{kmax} = \frac{360}{P_\phi}j.$$

Figure 2.6 illustrates, through an example, the polar graph embedding process where components are partitioned into discrete bins (b_1 through b_{16}) based on their absolute radial distance and angular position relative to a fixed global center.

Each node with polar coordinates can be assigned to one of the polar segments. Once the polar segments are defined for a subgraph, they are encoded into fixed histograms. To define a bin of the histogram, a maximum number of directions P_{bins} must be determined; this P_{bins} determines the number of bins for a single polar

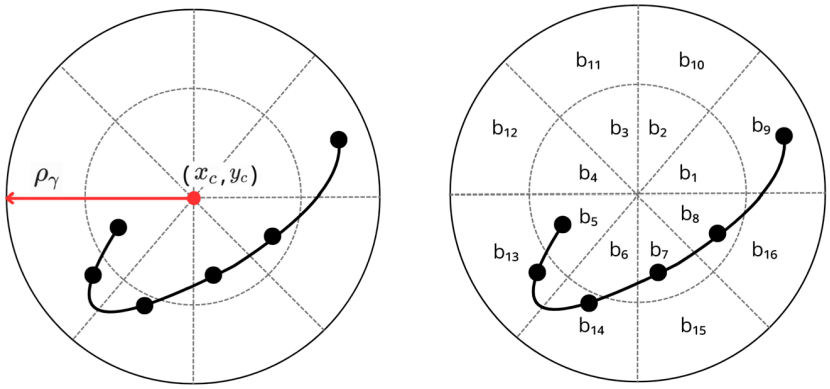


Figure 2.6: Example of a polar graph embedding, of a subgraph, into $n = P_r \times P_\phi$ polar segments ($16 = 2 \cdot 8.$).

segment. For every edge in a polar segment, the length of that edge is calculated as d and its angle θ is recorded. Then the distance d is assigned to one of two bins by:

$$b_{ki} += 1 - \frac{\theta - \theta_i}{\epsilon} d, \quad b_{kj} += \frac{\theta - \theta_i}{\epsilon} d,$$

where $\epsilon = \frac{360}{P_{\text{bins}}}$ is the radial range per bin and $\theta_i = i\epsilon$. Ultimately, the P_{bins} bins for the n polar segments are concatenated into one global histogram $H = \{b_{1,1}, \ldots, b_{1,P_{\text{bins}}}, \ldots, b_{n,1}, \ldots, b_{n,P_{\text{bins}}}\}$. To clarify, there is one global histogram for each sliding window. The global histograms are normalized with the L^2 norm [4, pp. 2–4]. Figure 2.7 illustrates how each node gets assigned to one of the polar segments and encoded into fixed histograms. Each distinct step of the 'sliding window' produces its own independent histogram, represented as vectors.

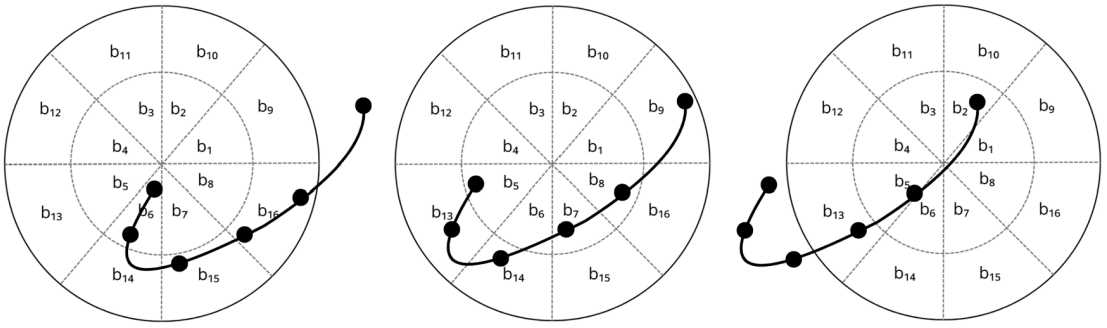


Figure 2.7: Example positions of the 'sliding window' and the corresponding histograms, represented as vectors, for each step..

2.3.2 Structural Dynamic Time Warping

Once the histograms are finalized, the graph dissimilarity is calculated. The dissimilarity between two histograms H and H' is determined by χ^2-distance. The Polar

Graph Embedding distance, or PGEd, is given by:

$$PGEd(H, H') = \sum_{i=1}^{n} \sum_{j=1}^{P_{\text{bins}}} \frac{(b_{i_j} - b'_{i_j})^2}{b_{i_j} + b'_{i_j}}.$$

To formulate the distance matrix needed for structural time warping, $PGEd(H, H')$ is calculated for every possible pair of windows. When the center position of the sliding window x_c moves along the graph, a new histogram is extracted for each position.

To perform structural time warping, the approach proposed by [4] is followed, extending it with a dynamic grid construction to ensure computational robustness. Let $w \in (0, 1]$ be a user-defined granularity parameter. As the sliding window traverses each graph, it generates sequences of histograms $X = \{H_1, \ldots, H_m\}$ and $Y = \{H'_1, \ldots, H'_n\}$, where the cardinalities m and n are determined by the total number of valid window positions identified during the extraction process for graphs g_1 and g_2, respectively.

This data-driven approach ensures the cost matrix $C \in \mathbb{R}^{m \times n}$ aligns precisely with the observed event sequences, addressing the boundary-mismatch issues inherent in static, predictive discretization. The alignment cost for each pair $(H_i, H'_j) \in X \times Y$ is defined by the distance $PGEd(H_i, H'_j)$. Finally, the structural distance $SDTW(g_1, g_2)$ is computed as the minimum cumulative cost path through the cost matrix C using dynamic programming. If K is the length of the optimal warping path $W = \{(i_1, j_1), \ldots, (i_K, j_K)\}$, then [4, pp. 2–4]:

$$SDTW(g_1, g_2) = \sum_{k=1}^{K} PGEd(H_{i_k}, H'_{j_k}).$$

Finding the optimal path can be intuitively understood through the 2D alignment grid in Figure 2.8. Darker regions indicate lower structural distance calculated via $\text{PGE}_d(H, H')$. The red line represents the optimal warping path W, which accumulates these local costs from the initial to the final state. Consequently, $\text{SDTW}(g_1, g_2)$ corresponds to the total minimal cumulative cost obtained at the terminal cell of this path.

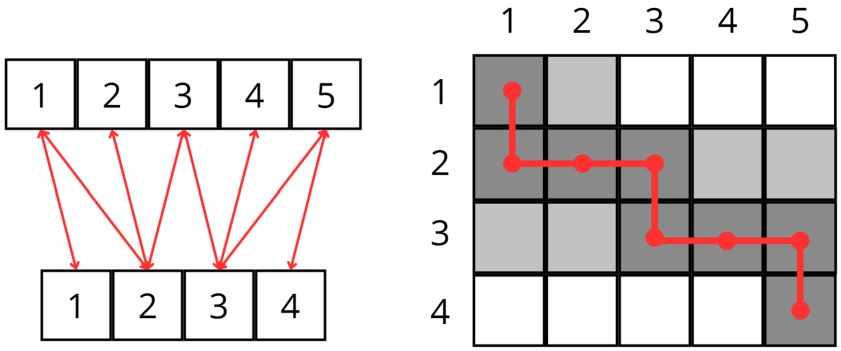


Figure 2.8: Visual intuition of the SDTW cost matrix (right). Darker regions indicate lower structural distance calculated via $\mathrm{PGE}_d(H, H')$. The red line highlighted the optimal warping path. To the left the corresponding graph alignment is shown.

3
Methods

In this chapter an overview of the algorithms are given. The algorithm and parameters choices are supported, contrasted and criticized. Specific details in implementation is also commented on.

3.1 Specification of the problems being investigated

The main objective of this master thesis is to develop an algorithm for cross-carrier event sequence alignment and canonicalization, mapping heterogeneous tracking events into a unified shipment lifecycle. The pipeline should be able to transform raw carrier data to human interpretable shipment cycles known as "golden paths". These "golden paths" should well represent the main possible shipment routes for the given carrier. Furthermore, once "golden paths" are obtain these will be used to align shipment cycles with the goal of understanding which event labels from different carriers correspond to one and other.

3.2 Overview of the pipeline and the main objectives

When investigating one of the goals, which was to find a node alignment between tracking events for different carriers, a pipeline have been developed. This subchapter gives an overview of each step. The pipeline is as follows:

- Pick two carriers, carrier A and carrier B. A carrier is defined as a combination of one company and one delivery service.
- Cluster the shipment cycle data into clusters for both carriers separately. This step can require a pre-step of process mining to filter out the noise and low observed behavior. The clustering uses Weisfeiler-Lehman kernels together with established cluster methods such as K-means.
- For each cluster obtained find a process model to acquire golden paths.
- Compare all pairwise golden paths from the two carriers, which is a single shipment network, to obtain node alignments. This is done initially by Structural Dynamic Time Warping and gets followed by a heuristic.
- Conclude which labels in carrier A correspond with which labels in carrier B to observe differently named events that might be same "true" event.

3.3 Process Mining

Process mining serve several purposes in this thesis. It provides a way to explain behavior of a carrier and thus finding the normal flow or allowed flow for that carrier. The normal flow provides an easily digestible overview of a carrier meanwhile the allowed flow explains the general rule set shipment cycles follow in that carrier. The fitness scores provide a way to flag deviant behavior in traces. And verify if new traces does follow the reestablished model for that carrier. Furthermore, for the goal of finding alignments between tracking events, process mining serve as a middle step in the pipeline. After clustering the data, process mining can pick out the golden paths that later are the networks that alignment operates on.

3.3.1 Model Faithfulness and Survival Metrics

Once a set of feasible paths through a network is obtained, it is necessary to quantify how accurately this representation captures the traces it was based on. This project utilizes two distinct metrics for evaluation Survival Rate and Filtered Survival Rate.

3.3.1.1 Survival Rate

The first approach is a rigorous binary check if a trace fulfills the processed causal dependencies. For a trace to survive it must meet three conditions:

- The starting node must correspond to an allowed initial activity.
- Each subsequent node must transition to a valid activity in its output set.
- Each node must have arrived from an activity within its input set.

If any condition is breached that trace is considered "dead". To obtained a Survival Rate for an event log together with its discovered process model one needs to replay all traces. The Survival Rate for an event log then computed as:

$$\text{Survival Rate} = \frac{\text{Number of survived traces}}{\text{Total number of traces}}.$$

While this strict replaying check often fails to represent fitness well in real-world data, sometimes resulting in 0 fitness for seemingly consistent models, it remains a valuable tool for identifying most conformant traces [1, pp. 246–247]. Utilizing a Survival Rate filter allows for removal of noise, outliers or low frequency events, though the quality of the surviving traces rely heavily on thresholds, such as Dependency threshold, used during model discovery.

3.3.1.2 Filtered Survival Rate

In contrast to the binary survival check, Filtered Survival Rate leverages token replay to quantify partial fitness. While global event log fitness measures exist (see. (2.4)) they do not identify which traces deviate. By applying a fitness value individually to each trace using (2.3), a trace is considered to survive if it surpasses a set limit.

Token replay is primarily feasible when a causal matrix is simple enough to be transformed into a Petri net. Key considerations for this metric include:

- Only traces where every event is present as a *transition* in the Petri net are considered. This is known as *Coverage*.
- Unlike the binary Survival Rate, this metric can account for traces that fail on a single event but otherwise follow the model closely.
- This model makes it possible to isolate the survived traces for further analysis, such as identifying "golden paths".

The Filtered Survival Rate is highly sensitive to both model discovery thresholds and specific fitness thresholds. This metric is particularly useful for real-world data, which often exhibits high variance that would otherwise result in a near 0 Survival Rate. However, since token replay is not always feasible, Survival Rate can be more applicable in real-world situations.

3.3.2 Sensitivity Analysis on HeuristicMiner algorithm

When performing sensitivity analysis on the HeuristicMiner, it is necessary to define a metric for algorithm success. This paper utilizes the Survival Rate, which quantifies the percentage of event traces from the original event log that the discovered model can successfully replicate. However, using Survival Rate as a proxy for accuracy presents challenges. While high Survival Rate indicates high fitness it may signal underfitting. In the case of underfitting a "spaghetti" model, a complex model with excessive allowed paths, may achieve a near perfect Survival Rate while failing to extract meaningful process. In contrast to this, a lower Survival Rate is not inherently a failure since it can represent the algorithms ability to filter out noise, low-frequency exceptions and outliers to reveal the process core behavior. Regardless, Survival Rate is a quantifiable metric that characterizes the faithfulness of the model to the raw data and highlights the extent of information loss during the discovery process. The Survival Rate is a rigorous and quite punishing metric that focuses on the allows sensitivity analysis to focus on the algorithms ability to discover the main process.

3.3.2.1 Sensitivity Analysis on Positive Observation Parameters

The heuristic miner algorithm that was implemented does only include threshold values after initializing backbone of highest dependencies. This could essentially in rare situations mean that the backbone include a high dependency connection of event that have extremely low frequency pair. While the dependency measure formula partially mitigates this by incorporating frequency, the all-activities-connected heuristic may still force a connection if an activity has only one observed (and potentially noisy) successor. To prevent this happening, since such problems arose during tests on data, a new threshold was introduced by this thesis: Positive observation threshold Backbone. This ensures that even in the backbone a connection must have a frequency above a set threshold to be included.

A sensitivity analysis was designed to test this new parameter together with the established Positive observation threshold. These parameters were tested against carriers X_1, X_2, X_3 and X_4 using Survival Rate as the success metric. The analysis was performed by incrementally increasing both the positive observation thresholds values from 0 up to 1 in 0.1 steps. To isolate effect of the other variables, other parameters were held constant: dependency threshold was set to the standard real life data value of 0.7, and a high relative threshold value of 10 was used to prevent interference with the model. Furthermore, a proportional relationship was proposed where Positive observation threshold Backbone is set to 10% of original Positive observation threshold value. Since Positive observation threshold Backbone acts on the 'all-activities-connected' which is more sensitive than the entire multi set, which Positive observation threshold effects, a lower value is expected. Setting Positive observation threshold Backbone too high would most certainly collapse the entire model. Additionally, a lower value combines possibility of finding a valid backbone with ensuring that the backbone is based on observed evidence.

3.3.2.2 Sensitivity Analysis on Dependency threshold

A sensitivity analysis was performed on the Dependency threshold to evaluate its impact on model discovery across carriers X_1, X_2, X_3, X_4, Z_1, Z_3 and Z_4. This parameter indicates the certainty of a true dependency between two events based on a frequency-based metric. In accordance with HeuristicMiners use of thresholds to differentiate between main behavior and noise, the threshold was tested from 0 up to 1 in 0.1 steps to identify a "sweet spot" where fitness remains high while noise is filtered. Fitness is quantified, as before, by Survival Rate. During these test the other parameters where held as constants as described before.

3.3.2.3 Sensitivity Analysis on Relative Observation

A sensitivity analysis was performed on the Relative Observation threshold to evaluate its impact on model discovery across carriers X_1, X_2,X_3, X_4, Z_1, Z_3 and Z_4. This process is done similarity as for sensitivity analysis on the Dependency threshold.

3.3.2.4 Sensitivity Analysis of Clustering by Process Mining Validation

Validation of clusters for networks is done using process mining Survival Rate. In this context, the goal of the process mining shifts from discovering unknown process models to serving as a quantitative success metric for the clustering phase. After clustering is performed via Weisfeiler-Lehman kernels, each cluster is processed though the HeuristicMiner to detect its Survival Rate. This secondary validation step is necessary because traditional heuristic methods, such as the K-means Elbow method, can often result in ambiguous or unstable optimal K values depending on the initial centroid placement. Unlike standard process discovery, where a high Survival Rate might indicate underfitting, a high Survival Rate instead implies high intra-cluster similarity. This is essential for finding "golden paths", as a successful cluster must contain process instances consistent enough to be represented by a single causal matrix.

3.3.3 Process Mining before Clustering

In cases where carriers include a large variety of networks, including significant levels of noise and outliers, standard clustering often fail to produce a viable divide. To address this, this paper proposed discovering of the main underlying process before clustering. This solution is based on the assumption that filtering out the most present behavior; thus effectively removing noise, outliers and low frequency events, allows for more distinct clusters. Survival Rate of traces are used as a fitness check for the clusters. By isolating the event traces that meet the strict survival criterion it is ensured that the clustering is based on observed behavior rather than noise. This is pre-process mining step is not necessary for all carriers.

To clarify the practical execution of this step: instead of clustering the event logs directly using Weisfeiler-Lehman (WL) kernels, one can perform process mining first. In this approach, a process model is discovered for the full, un-clustered data. This model is then used to filter out traces from the original data that do not fulfill the discovered model's rules, effectively removing low-frequency behavior and noise. Subsequent clustering is then performed only on the 'core' traces that satisfy the survival criteria. This ensures the structural alignment and clustering are focused on representative data.

3.3.4 Implementation comment on choice of the first element in HeuristicMiner

During the implementation of the all-connected-heuristic, the algorithm needs to find the starting and ending points of the process. Ideally, the starting node is found by looking for a column in the dependency matrix with no positive values. That means no edges flow into that node. However, because real-world event logs are often noisy and complex, there is no guarantee that a unique column like this exists. To handle this, the code first checks for a unique column and if it cannot find one, it falls back on a frequency count. This frequency count chooses whichever event appears most often as the very first node in the shipment cycles. This solution isn't perfect, mainly since the heuristic assumes every activity must have a cause, earlier preprocessing and clustering steps help by filtering out the noisy, low-frequency activities that usually cause this issue. The ending node is then found in the same way, but by searching the rows of the dependency matrix for a node with no outgoing edges.

3.4 Weisfeiler-Lehman Kernels

In this thesis, Weisfeiler-Lehman kernels serve as a critical intermediary step to facilitate the clustering of shipment networks. Direct clustering of graph structures presents computational and mathematical challenges. A robust method for distance based clustering of graphs is transforming these networks into a vector-based representation.

A notable characteristic of this implementation is the abstraction of temporal information. By utilizing the Weisfeiler-Lehman subtree kernel, the focus is shifted exclusively to the structural and topological properties of the shipment cycles. This results in the loss of specific time-duration data during the clustering phase. However, it ensures that the resulting carrier groupings are defined by their underlying operational logic and connectivity patterns rather than by temporal variance. Temporal variance can often be noisy in real-world logistics data. Additionally, it would complicate the process of clustering further.

3.4.1 Structural Feature Extraction using Weisfeiler-Lehman Kernels

The implementation of the Weisfeiler-Lehman subtree kernel algorithm (2) is tailored to transform the event logs into a feature space that captures he unique structural characteristics of shipment cycles. The initial step of the algorithm, creating of the node label multi sets, is adapted to utilize the out-degree adjacency matrix since shipment cycles are inherently directed graphs. Defining a node's neighborhood structure strictly by its outgoing edges places a deliberate emphasis on downstream structures. This enables the similarity comparison to focus on where tracking events can lead, which is more representative of logistic routing than in-degree approach. While incorporating both directions could theoretically provide more complete structural picture it risks introducing excessive sensitivity to the feature vectors. That could potentially cause structural similar flows to appear divergent due to minor variations in early stage reporting.

As the Weisfeiler-Lehman algorithm progresses through iterations, the number of unique labels grows exponentially, leading to extremely high-dimensional and sparse feature vectors. To mitigate the resulting computational complexity and memory requirements, a hashing trick is employed using the `FeatureHasher` from the `sklearn` library. This allows the high-dimensional label space to be mapped into a lower, fixed-dimensional space. Even though hashing does not guarantee distance preservation, the sheer number of features makes it unlikely that distant vectors collide under a good hash function. Following this projection, the vectors are subjected to L_2 normalization. This step is crucial because the original Weisfeiler-Lehman kernel is based on node counts; without normalization, the similarity measure would be biased by the total number of events in a shipment cycle rather than its structure. Normalizing the vectors ensures that structural patterns are measured independently of graph size, which is essential for consistent clustering of carriers with varying trace lengths [2, pp. 2827–2828].

To ensure that subsequent clustering methods, such as K-means, remain computationally efficient while capturing the most significant structural patterns, `TruncatedSVD` is used to extract the principal components of the Weisfeiler-Lehman feature space. `TruncatedSVD` is preferred over standard Principal Component Analysis (PCA) because it does not require centering the data, a process that would destroy the sparsity of the hashed feature vectors. In this project, the number of components was set

to 100, a choice supported by the cumulative explained variance ratio. This ratio consistently accounted for 90% of the structural variance, indicating that a relatively small number of components is sufficient to represent the underlying patterns in these networks. The effectiveness of this low-rank representation is further influenced by the compact nature of the shipment networks. Consequently, the iteration depth of the Weisfeiler-Lehman kernel is kept low, typically $h = 2$ or 3. Higher iteration counts for small graphs risk over-refining the labels, which can lead to a loss of the broader structural similarities necessary for identifying meaningful carrier groups [2, pp. 2827–2828].

3.5 Structural Dynamic Time Warping and subsequent heuristic

Structural Dynamic Time Warping and the subsequent heuristic layer perform the node-to-node alignment of tracking events across different carriers. While Structural Dynamic Time Warping is a well-established algorithm for graph matching, this implementation utilizes a custom heuristic to refine the output into a viable mapping for heterogeneous logistics data. By comparing the "golden paths" of two carriers, the pipeline records an alignment alongside its accuracy score and transitive stability. It is in this stage where temporal data is primarily utilized. This is done by incorporating normalized event timestamps. The alignment therefore accounts for the spacing of shipment logistics, which is a critical feature for identifying equivalent event statuses.

3.5.1 Time-Normalized Coordinate Projection

To perform structural time warping on a pair of events, represented as graphs, each node must obtain a node label of (x, y). The y position is set to the same for all the nodes since shipment cycles are straight graphs; any arbitrary number can be chosen. The x position however is what determines the nodes distance from each other and structural position in the full relevant shipment cycle. Therefore the actual times recorded in data frames of these events are inserted. Since the two shipment cycles being compared at any time might happen on vastly different days the minutes between each event is recorded. For each shipment cycle the first event is always on $x = 0$. The second is the amount of minutes between event 1 and event 2. To enable structural information and spacing between events to take forefront in the alignment all times are normalized so that the last event always is $x = 1000$ for both graphs. Furthermore, there are instances where events can happen at the exactly same time in a shipment cycle even though the events are different. This is not something the structural time warping can handle. This is solved by adding a fake extra time, specifically $+1$ to one of them, so that they are separated. This design choice does introduce a slight order dependence into the graph structure. However, any ordering inevitably represents a departure from the original simultaneous data. For this thesis, the choice of ordering is treated as arbitrary.

3.5.2 Heuristic Refinement for Node-to-Node Mapping

Structural Dynamic Time Warping (SDTW) generates a window-to-window alignment; however, this pipeline seeks a node-to-node mapping. The window-to-window alignment serves as the primary structural guide for this transformation. To achieve this, the frequency of each node mapping across all optimal warping paths is computed, effectively aggregating the structural context. This frequency-based aggregation is inherently robust, as consistent, stable matches exert a stronger influence on the final alignment than unstable ones. To determine the final node-to-node correspondence, an alignment score for each relevant pair of nodes is constructed. This score integrates the window alignment frequency with temporal, topological, and frequency-based penalties, formulated as follows:

$$\text{Score} = A \cdot w_A - t \cdot w_t - T \cdot w_T - f \cdot w_f \tag{3.1}$$

where A denotes the window alignment frequency count; t, T, and f represent the temporal, topological, and frequency penalties, respectively; and w_A, w_t, w_T, w_f are the corresponding weighting coefficients. By tuning these hyperparameters, we can prioritize specific network properties, allowing the alignment process to be tailored to the structural characteristics of the shipment lifecycles.

Structural time warping primarily considers time and position into account. However, relying solely on these features may be insufficient. Frequency of a certain event provides a indicator of its underlying functional role within the shipment lifecycle. For instance, events such as "Shipment information updated" typically exhibit a high recurrence rate, whereas terminal events such as "Delivered" occur less often. Therefore a frequency penalty f is incorporated to the alignment. This penalty f is derived by normalizing the absolute occurrence count of events event E_1 and E_2 against the total number of shipments within the dataset:

$$f = \left| \frac{\text{Count of } E_1}{\text{Count of shipment cycles}} - \frac{\text{Count of } E_2}{\text{Count of shipment cycles}} \right|.$$

By penalizing the alignment of nodes with divergent global frequencies, this metric ensures that events with distinct behavioral profiles are less likely to be mapped to one another.

Structural Dynamic Time Warping will always produce an alignment path, even in regions of high uncertainty. This is particularly problematic when nodes are distributed across significantly different temporal positions. Furthermore, SDTW is insensitive to similar topological situations separated by large time differences. Consider Figure 3.1. That shows how structural alignment might fail to align nodes with equivalent structural positions. Node 2 and 3 clearly have the same positional place in regards to the node before and after them. In this instance, the distance between node 1 and 2 is large. The distance between node 2 and 3 is small and then the distance between 3 and 4 is slightly bigger. The local picture of those 4 nodes are therefore similar even though they are shifted in time from each other. One might assume Figure 3.2 is a better an alignment. With this reasoning a topological

penalty T will be added to the alignment. Topological penalty T for E_1 and E_2 is calculated simply as follows,

$$T = ||dist_b(E_2) - dist_b(E_1)| - |dist_a(E_2) - dist_a(E_1)||\,,$$

where $dist_b(E)$ denotes the distance the the event before E and $dist_a(E)$ denotes the distance to event after E. Topological penalty is not calculated for the start or end node.

Moreover, time is included in structural time warping but since it forces a connection nodes with very high time difference might still be paired. This is especially relevant when topological and frequency can counteract the original structural time warped guided alignment. To avoid matching nodes far away in the normalized time space a time penalty t is introduced. The time penalty t for E_1 and E_2 is simply calculated as

$$t = |\text{Time of } E_1 - \text{Time of } E_2|\,.$$

To derive a discrete node-to-node correspondence from the window-to-window alignments a mapping function is utilized. Given an alignment map generated through Structural Dynamic Time Warping (SDTW), this function establishes a direct link between individual tracking nodes in the reference network (G_R) and the most statistically significant candidate node in the target network (G_T). The function iterates sequentially through nodes in G_R, evaluating candidate mappings based on alignment scores. This approach formalizes the alignment by identifying the 'winning' candidate for each node. This greedy selection mechanism introduces several structural consequences:

- The pipeline is inherently order-dependent. Because the mapping is not bijective the results differ depending on which network is defined as the reference.
- The algorithm permits many-to-one mappings, allowing multiple nodes from the reference graph G_R to anchor to a single node in the target graph G_T. This is a necessary feature for real-world data, where tracking systems often differ in detail. However, this introduces the risk that a dominant "sink" node in the target network could disproportionately influence the alignment of multiple preceding reference nodes.
- Because the loop iterates over G_R, every node in the reference network is guaranteed a match, whereas G_T is not required to be fully covered. This further reinforces the directional asymmetry.

3.5.3 Evaluation Metrics for Heterogeneous Event Mapping

A fitness or accuracy score is necessary to quantify the quality of the alignments. One approach to evaluating alignment quality without a ground truth is to leverage the standardized event descriptions developed by Centiro. Some events have no attributed description assigned to them and are labeled "Unharmonized". By systematically comparing these descriptions across aligned nodes, a validation score

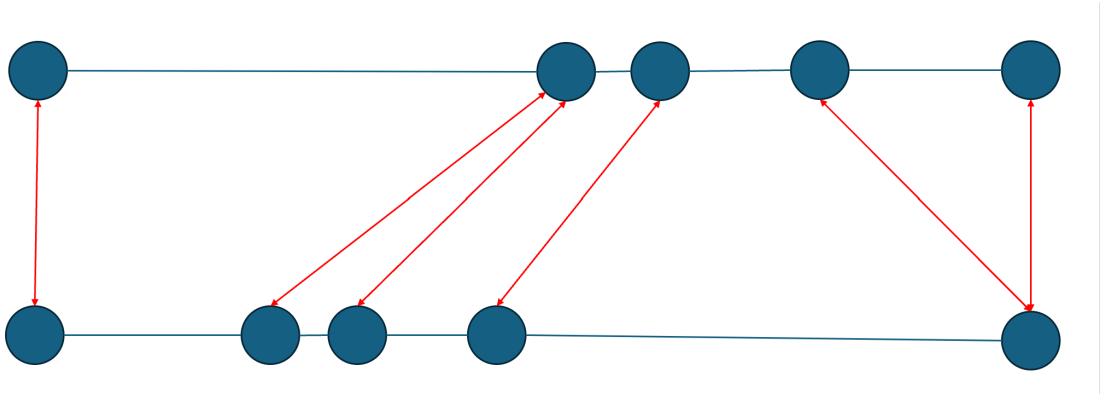


Figure 3.1: An example of a naive node-to-node alignment between two unlabeled shipment networks where structural dynamic time warping fails due to significant temporal shifts between localized node clusters.

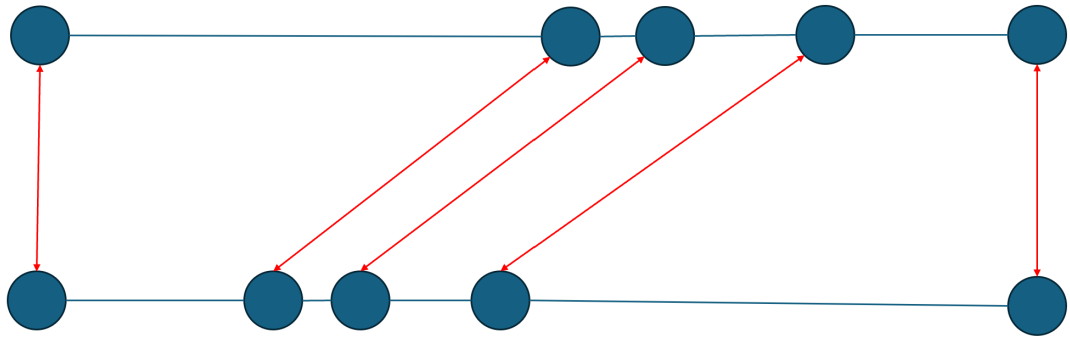


Figure 3.2: An optimized node-to-node alignment of the networks from Figure 3.1, demonstrating how incorporating structural position and topological contexts corrects the misalignment caused by temporal variance.

can be achieved. Consider a reference shipment cycle A, a target shipment cycle B, and a structural alignment between them. Let $a \in A$ be a node in the reference shipment cycle, and let $d(n)$ denote the description label of any given node n. Furthermore, let $B_a \subseteq B$ denote the set of all nodes in the target shipment cycle to which a is aligned. To determine alignment quality, every aligned node pair (a, b) for all $b \in B_a$ is evaluated. A pairing is flagged as invalid unless it satisfies one of the two conditions:

- The descriptions are identical, meaning $d(a) = d(b)$.
- The target node is unharmonized ($d(b) =$ "Unharmonized"), and the reference description $d(a)$ is not present anywhere within the target network's full set of descriptions: $d(a) \notin \{d(b') \mid b' \in B\}$.
- The reference node is unharmonized ($d(a) =$ "Unharmonized"), while the target node $d(b)$ is harmonized.

This comparison is executed systematically for every node in shipment cycle A until all relevant pairs of nodes have been examined. To acquire the score one computes:

$$\text{Accuracy Score} = \frac{\text{The number of unflagged pairs}}{\text{The total number of pairs looked at}}. \tag{3.2}$$

Note, that this does not guarantee a good alignment. It is better viewed as a insurance that the alignment is not bad. The accuracy score is based on the source network and not the target network. It is whether the nodes in the source network connect correctly that determines the score. Furthermore, wrong node-to-node alignment decrease the score with a significant amount. Say that one node is aligned wrong and the rest is correct. In a nine node graph that would mean a score of 0.9 in contrast to a six node graph which would have a score of 0.83. Figure 3.3 illustrates the systematic application of this metric to a sample alignment. As shown, the node "Shipment Information Received" in network A is aligned with the node "Unharmonized" in network B. Because "Shipment Information Received" does not exist anywhere within network B, this pairing satisfies the exception rule. It is therefore classified as valid. The alignment between A_2 and B_2 is also valid, as both nodes share the identical Centiro description. However, the alignment between A_2 and B_3 maps two mismatching descriptions and the pair is flagged as inaccurate. Evaluating all four aligned pairs using (3.2), the resulting Accuracy Score is computed as $\frac{2}{4} = 0.5$.

Furthermore, an accuracy score on a more abstract level was developed. This is done by attributing the given event descriptions to a more general description level. Thus, labels such as "Delivered to customer" and "Arrived at delivery" are both attributed as the general label "Delivered". Therefore alignment between those nodes will not flag when using logic in (3.2). There are two main purposes for an abstract accuracy score. The first one is allowing computation of some kind of accuracy score for carriers with different label names. The second reason is that alignment based on an abstract level of labels is more forgiving. This ensures the conclusion that even in certain instances where normal accuracy score is low the alignment is partly or essentially right. The abstract levels and what is attributed to which is based on

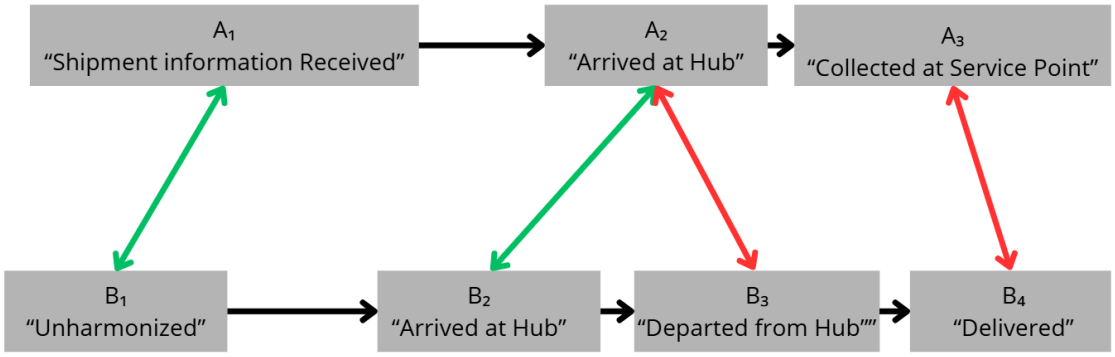


Figure 3.3: Structural alignment between network A and network B mapping Centiro descriptions. Each node displays both its structural identifier (A_1, B_2, etc.) and its assigned Centiro label description. Green edges represent valid alignment pairs according to the Accuracy Score criteria, whereas red edges denote pairs flagged as inaccurate. The illustrated alignment yields an overall Accuracy Score of 0.5.

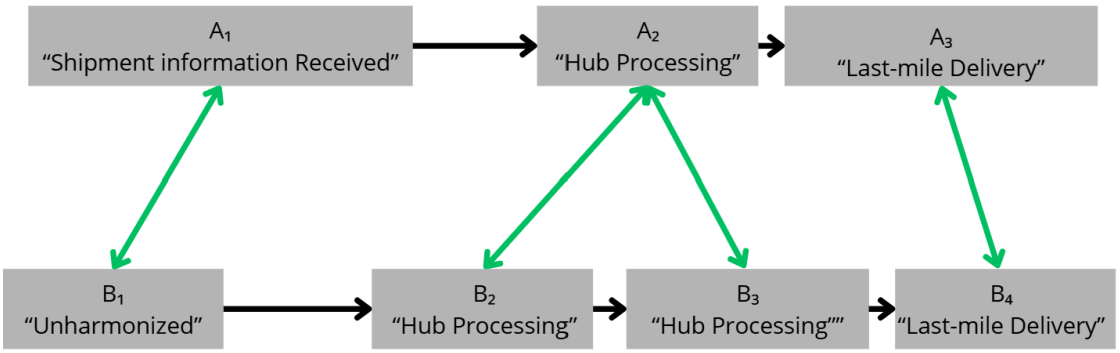


Figure 3.4: Structural alignment between network A and network B utilizing abstract-level Centiro descriptions. Each node displays both its structural identifier (A_1, B_2, etc.) and its corresponding abstract description. Green edges represent valid alignment pairs according to the Accuracy Score criteria, while red edges denote flagged mismatches. Due to the generalized mapping, all pairs are validated, resulting in an abstract Accuracy Score of 1.0.

information given by Centiro. Figure 3.4 illustrates the same two shipment cycles and structural alignment as Figure 3.3, with the key distinction that the metric is evaluated using the abstract-level description categories. Because "Arrived at Hub" and "Departed from Hub" map to the same generalized description, the alignment pair between A_2 and B_3 is recognized as valid under this framework. Using this abstract classification, all alignment pairs fulfill the validation criteria, and the abstract Accuracy Score is computed as $\frac{4}{4} = 1.0$.

Transitive consistency score is a metric used to evaluate the structural consistency of alignments across three distinct carriers: A, B, and C. By examining the pairwise alignments $A - B$ and $B - C$, the implied mapping between $A - C$ can be

logically inferred under the assumption that the transitive property holds. This inferred mapping is then compared against the empirically derived $A - C$ alignment to measure consistency. Specifically, the metric quantifies the number of nodes in A that map to the exact same counterpart nodes in both the inferred and empirical alignments. Let $A_{\text{consistent}}$ be the set of nodes that map consistently and $|\cdot|$ denote the cardinality of the sets. Then:

$$\text{Transitive consistency} = \frac{|A_{\text{consistent}}|}{|A|}.$$

Because a robust alignment framework should theoretically preserve transitivity, this metric serves as a critical indicator of global alignment integrity.

3.5.4 Parameter Tuning via Grid Search

Different pair of graphs do most likely require different weights to produce significantly better alignment. For example, a combination of two networks with exactly the same amount events might need higher topological importance in contrast to the other parameters. A combination of a longer network consisting of several repeated nodes with a shorter network will most likely require higher focus frequency than the first pair. Therefore, splitting the data into pair subgroups and then finding parameter values for each of the subgroups would be optimal. However, the small amount of data makes it dangerous to perform parameter fitting due to overfitting, even before distribution the pairs into subgroups. Therefore finding one set of global parameters by grid search is suboptimal but feasible. Regardless of the sub optimality of the solution there are several parameter values that do improve the result of the alignment. The data used to discover parameters was divided into 75% for fitting and the remaining for validation. Since the alignment score depends on the relative proportions between weights rather than their absolute values one weight parameters can be fixed as a baseline to eliminate a redundant degree of freedom. Therefore, the Structural Time Warping signal weight was fixed at $w_A = 75$. The grid search is done by weights w_t, w_T and w_f be the values of 0.01, 10 and 100. For the fitting part of data the appropriate values of each is denoted w_t, w_T and w_f, where accuracy score described in (3.2) determines performance. The parameter that yield best performance, or one of them if several give same score, is then used to perform alignment on the validation data to confirm if the chosen parameters were sufficient. Furthermore, jitter is added later in validation step as 1.2 times values and 0.8 times values to ensure that the region is somewhat stable.

4
Results

4.1 Stability Analysis

Sensitivity analysis on the process mining algorithm ensures that the chosen 'golden paths' are constructed only from sequences with significant causal relationships. This thereby prevents rare logistical anomalies from skewing the final alignments.

4.1.1 Positive observation threshold and Positive observation threshold Backbone

The sensitivity analysis on Positive observation threshold and Positive observation threshold Backbone conducted on carriers X_1, X_2, X_3 and X_4 revealed a consistent behavior pattern across these datasets. As shown in Figure 4.1 the Strict Survival Rate remains stable for lower threshold values. The figure implies that useful model discovery requires a minimum of 0.1 and that the model structurally collapses by 0.6.

An important finding is the stabilizing effect of 10% ratio between Positive observation threshold and Positive observation threshold Backbone. Figure 4.2 demonstrates the increased stability when applying the relationship. Based on these findings a optimal Positive observation threshold value of 0.3 was chosen for balancing noise filtering with model completeness.

4.1.2 Dependency threshold

The sensitivity analysis on dependency threshold reveals a high degree of parameter independence for the following carriers X_1, X_2, X_3, Z_1, Z_3 and Z_4. As illustrated in Figure 4.3 the survival rate drastically drops around 0.8 and therefore an internal stability of the process model before this value. The variations observed in certain carriers (represented by green and red lines) likely reflect noisier or more uniform distributions that offer fewer clear causal dependencies. Based on these performance trends, a Dependency threshold of 0.7 was selected.

4.1.3 Relative Observation threshold

The sensitivity analysis on relative observation threshold reveal a high degree of parameter independence for the following carriers X_1, X_2, X_3, Z_1, Z_3 and Z_4. As illustrated in Figure 4.4 the survival rate increases around 0.1 until 0.2 and afterwards remains stable. This therefore implies a internal stability of the process model after

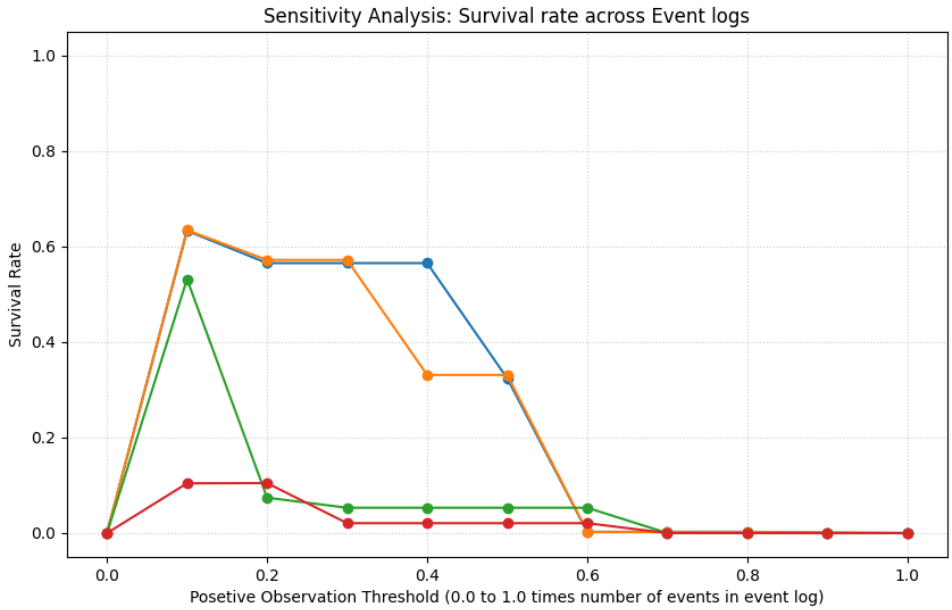


Figure 4.1: Sensitivity analysis of the Positive Observation parameters evaluated across carriers X_1, X_2, X_3, and X_4.

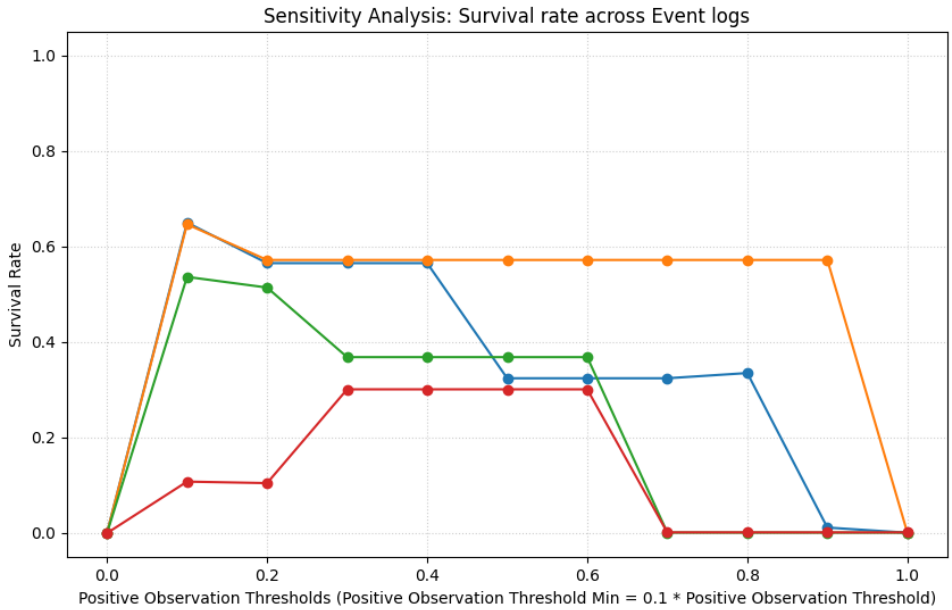


Figure 4.2: Sensitivity analysis of the Positive Observation parameters evaluated across carriers X_1, X_2, X_3, and X_4 when Positive Observation Threshold Backbone = 0.1 * Positive Observation Threshold. Illustrating a more stable model in comparison to Figure 4.1.

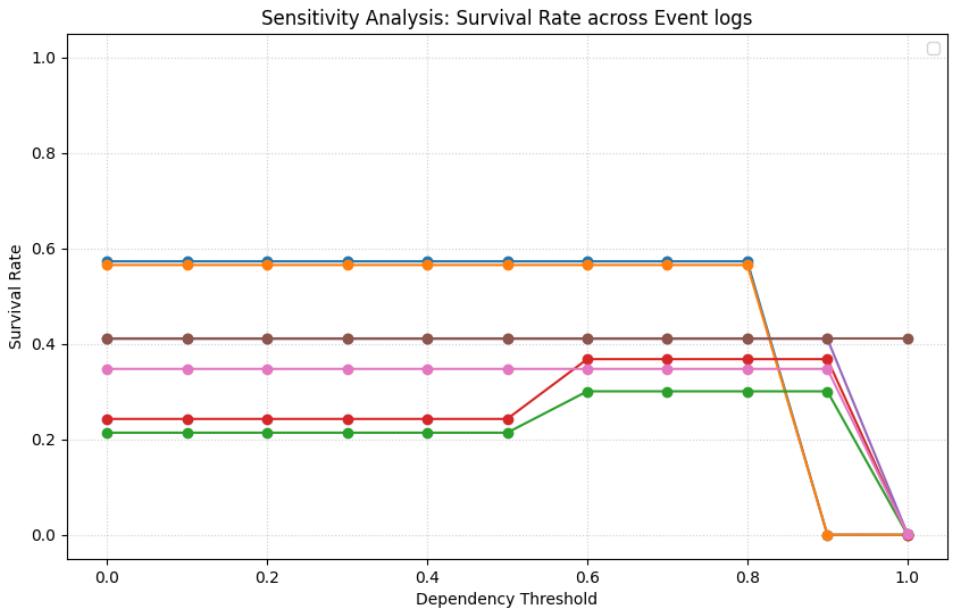


Figure 4.3: Sensitivity analysis of the Dependency Threshold parameter evaluated across carriers X_1, X_2, X_3, X_4, Z_1, Z_3 and Z_4 illustrating a distinct drop-off in Strict Survival Rate as the threshold exceeds 0.8.

this value. This suggest that on these carriers the Relative Observation threshold does not change model discovery drastically. Based on these findings a value of 0.3 was chosen.

4.2 Structural Clustering and Golden Path Discovery

Following the parameter optimization through sensitivity analysis the shipment cycles were clustered to identify different underlying process routes. These clusters are then transformed into 'golden paths' since they represent the main paths a shipment can travel from that carrier. This stage is critical for reducing raw data into consistent sub-models suitable for cross-carrier alignment. The following subsections evaluate the quality of these clusters through a representative case study of carrier X_1, illustrate the occasional necessity of process-mining-based pre-processing, and analyze the limitations of structural clustering.

4.2.1 Representative Case Study: Cluster Quality of Carrier X_1

As established in the Methodology, the quality of clusters was evaluated based on Survival Rate. Figure 4.5 illustrates the elbow for carrier X_1 which initially infers 5 clusters. However, the value is ambiguous and unstable. The secondary validation

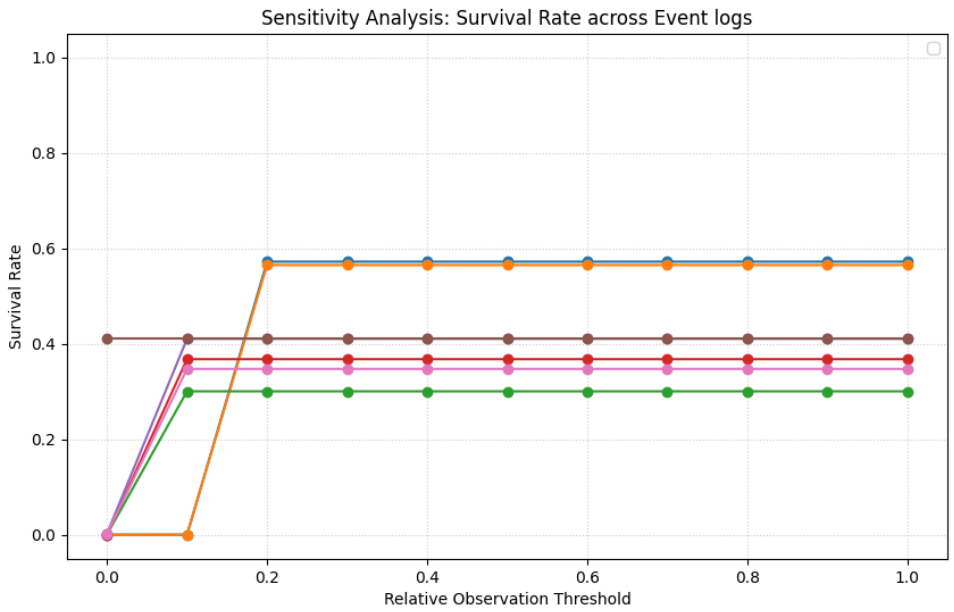


Figure 4.4: Sensitivity analysis of the Relative Observation Threshold parameter evaluated across carriers X_1, X_2, X_3, X_4, Z_1, Z_3 and Z_4 illustrating a stable process models after a value of 0.2.

using Survival Rate in process mining Figure 4.6 supports the initial choice of 5 clusters because of its plateau on $5-6$. At lower cluster counts, the overall Survival Rate is significantly lower, suggesting that the causal matrix fails to capture the diverse behaviors forced into the same cluster. Conversely, when the number of clusters exceeds the plateau survival initially decreases before rising again. The trend is likely due to overfitting, where similar networks are split into redundant clusters. Observations regarding the filtered survival rate (orange line) were adjusted to account for instances where a Petri net could not be successfully generated from the mined causal matrix.

4.2.2 Multi-Carrier Generalization

A high degree of consistency was observed across the carriers between the mathematical elbow of the Weisfeiler-Lehman kernels and the inferred once using validation through Survival Rate. While X_1 was used here as a representative example equivalent information for the remaining carriers can be found in appendix. Table 4.1 summarizes these findings.

4.2.3 Impact of Process Pre-processing on Cluster Quality

The necessity for pre-processing is demonstrated by initial clustering results for carrier X_4. The elbow in Figure 4.7 on raw carrier data is undefined and gives no clear indication of number of clusters via K-means. Furthermore, the Survival Rate is fairly low, as seen in Table 4.3, and thus suggesting lack of intra-cluster similarity.

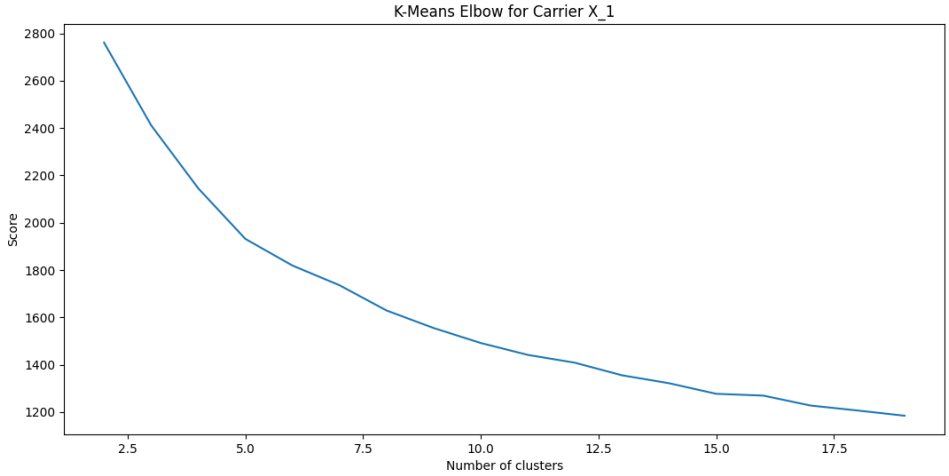


Figure 4.5: K-means Elbow for Carrier X_1

Table 4.1: Optimal K elbow versus Survival Plateau for different carriers

Carriers Group	Carrier ID	Optimal K (elbow)	Survival Plateau (K)	Final K Chosen
X	X_1	5	5-6	5
	X_2	5	5-6	5
Z	Z_1	3/6	4-5	4

However, after performing process mining to isolate the main process (30% of traces survive Strict Filtering and thus supports the assumption that the carrier included high levels of undesirable traces), the resulting K-means elbow in Figure 4.8 is significantly cleaner. This confirms that the main process clusters effectively once the outliers, noise and low frequency events are removed. The Survival Rate increases from 0.54 to 0.68 and indicates that the clusters are much more representative of 'golden paths'. Furthermore, the transformation from an undefined elbow in Figure 4.7 to the distinct transition in Figure 4.8 empirically validates the hypothesis that process mining successfully isolates the 'logistical core' from stochastic noise.

Table 4.2: Average Survival Rate of clusters (second rate excludes the lowest value) before and after pre processing the networks using Process Mining.

Carrier	X_4	X_3
Average Survival Rate	0.538/0.680	0.462/0.627
Average Survival Rate after Process Mining	0.659/0.878	0.635/0.847

4.2.4 Analysis of Clustering Limitations: The Case of Carrier Z_4

Despite the pre-processing steps, clustering results can occasionally remain ambiguous. Carrier Z_4 is taken as an example. Carrier Z_4 initial clustering yielded no

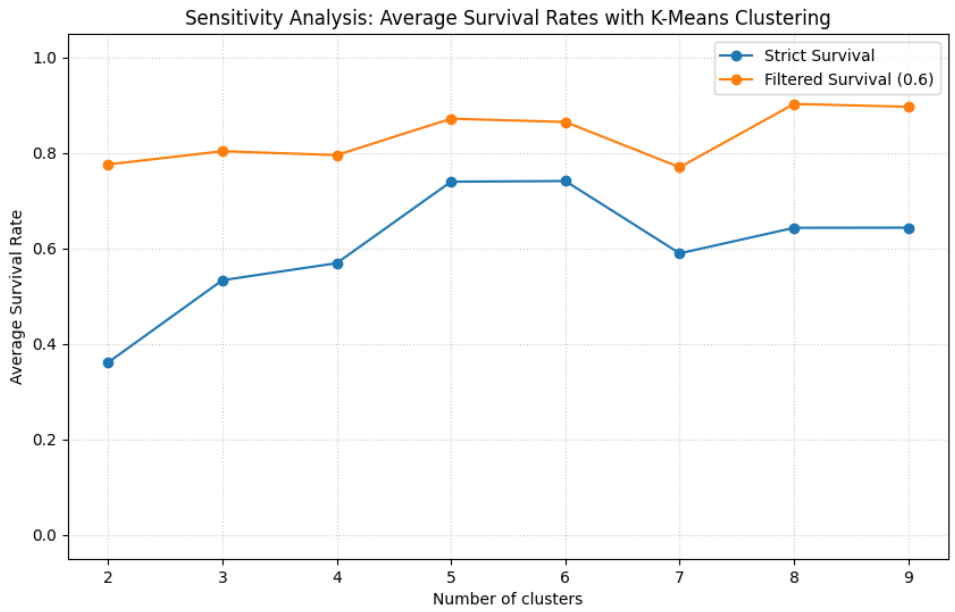


Figure 4.6: Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier X_1. The distinct plateau observed between $K = 5$ and $K = 6$ validates the optimal cluster selection. Filtering threshold was set at 0.6.

clear divide. However, by applying process mining prior to clustering, a clear elbow was identified at $K = 3$ on the 34.8% traces that survived strict filtering (elbow in Figure 4.9).

Despite these steps the resulting clustering can occasionally remain undesirable. The initial clustering of Z_4 gave no clear indications of clusters or underlying behavior in its networks. Therefore process mining was performed before clustering to investigate if the main process had clusters. In that case the resulting elbow, on the 34.8% survival traces, for K-means produces a clear indicator of 3 clusters. Evaluation on these clusters though HeuristicMiner clusters revealed diverging results, as seen in Table 4.3. Cluster 2 and 3 showed perfect Survival Rates and cluster 1 yielded a survival rate of 0. This suggest that cluster 2 and 3 consists of highly repetitive and consistent networks. Cluster 1 serves as a sink for high entropy data.

This phenomenon can be attributed to two factors. First, Weisfeiler-Lehman kernels may group networks based on structural complexity. This often results in "Spaghetti" models, complex graphs with high number of connections, are clustered together regardless of their underlying causal logic. Second, the kernels have a tendency to attribute unfinished, incomplete and truncated traces into one cluster. Since Survival Rate calculation does not account for truncated traces these appear to have 0 fitness. Consequently, the clustering remains valid as a tool for differentiating order from chaos.

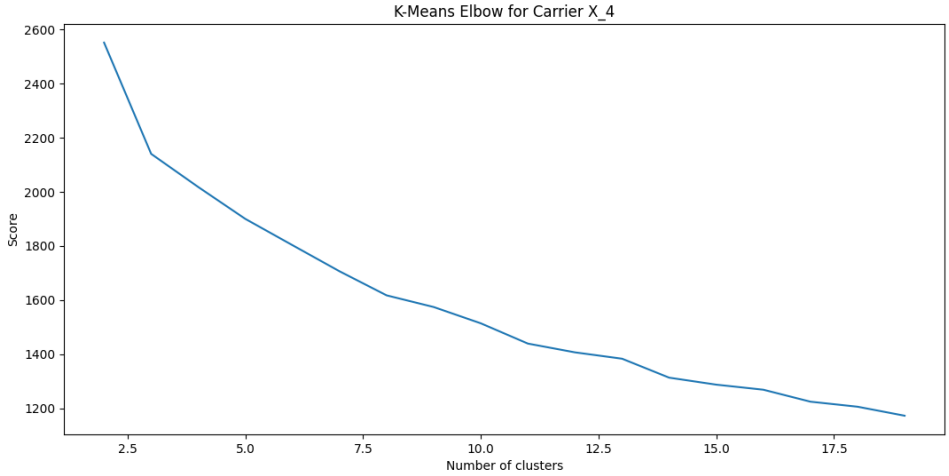


Figure 4.7: Initial K-means elbow for carrier X_4.

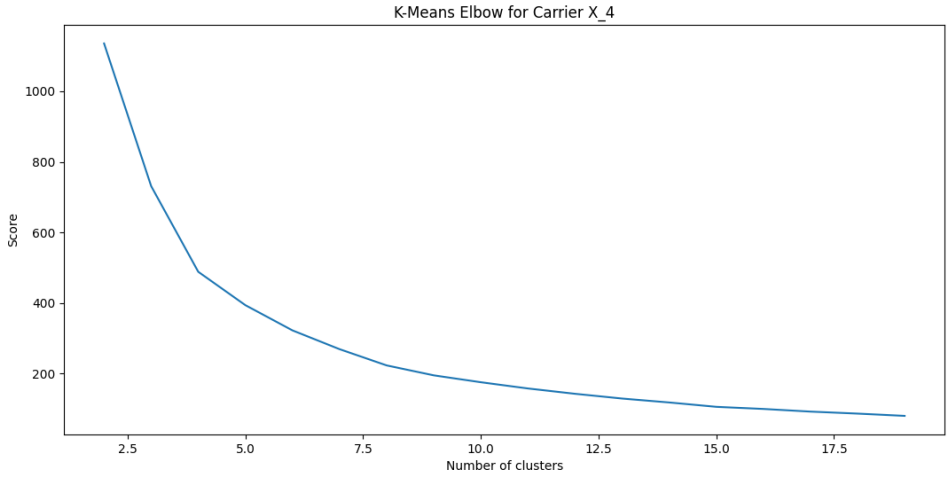


Figure 4.8: K-means elbow for carrier X_4 after Process Mining Pro-processing.

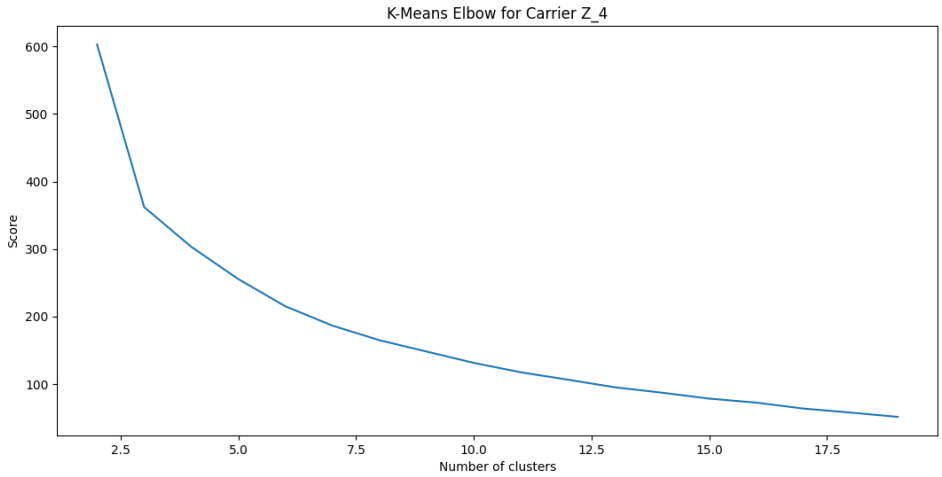


Figure 4.9: K-means elbow for carrier Z_4

Table 4.3: Process mining on clusters for carrier Z_4 after initial process mining

	Number of networks in cluster	Strict Survival rate	Survival rate
Cluster 1	936	0	Not performed
Cluster 2	12661	1.0	1.0
Cluster 3	2095	1.0	1.0

4.3 Performance

The main objective of the pipeline is the canonicalization of heterogeneous tracking events into a unified shipment lifecycle. This section moves from establishing a method for discovering 'golden paths' to evaluating the pipeline's performance in aligning those paths across carriers. Because logistical events are often named differently but share the same functional purpose, a multi-dimensional scoring approach is employed. This enables the evaluation to move beyond literal string matching and capture the underlying 'logistical truth'.

4.3.1 Weight Parameters

Including parameter weights consistently perform better, or equally, than solely relyin on the structural time warping, regardless of their sub-optimality. This is accounted for in Table 4.4. This presents a set of example alignments; comparing all possible carriers would repeat already-known information. The examples are taken to include an illustrative set of comparison of carriers from the same company and between companies. While structural time warping provides a geometric baseline for alignment, the inclusion of time, topological, and frequency penalties (w_t, w_T, w_f) allows the algorithm to distinguish between structurally similar but logistically distinct events.

Table 4.4: Carrier Accuracy Comparison with and without weight parameters. The weight parameters used were $w_A = 75$, $w_t = 100$, $w_T = 0.01$ and $w_f = 10$.

Carriers Aligned	Average Accuracy Score	Average Accuracy Score ($\forall w = 0$)	Top 3 Accuracy Score	Top 3 Accuracy Score ($\forall w = 0$)
Z_4 - Z_1	0.7778	0.7407	[0.8889, 0.7778, 0.7778]	[0.8889, 0.7778, 0.7778]
X_1 - Z_1	0.5833	0.4907	[0.7778, 0.6667, 0.6667]	[0.7778, 0.7778, 0.4444]
X_1 - X_4	0.4583	0.3056	[0.5556, 0.5556, 0.5556]	[0.3333, 0.3333, 0.3333]
X_1 - X_2	0.8681	0.7292	[1.0, 1.0, 0.8889]	[0.8889, 0.7778, 0.7778]
X_1 - Y_2	0.55	0.5333	[0.7778, 0.7778, 0.7778]	[0.7778, 0.7778, 0.7778]

4.3.2 Examples Alignments between two Carriers

Alignment "correctness" cannot be determined by solely one metric, it is a multi-dimensional problem. Accuracy Score provides a strict baseline based on literal event descriptions. The Abstract Accuracy Score is essential for capturing the "logistical truth" of the shipment life-cycle. Comparing these metrics across a diverse set of carrier pairs demonstrates the algorithm's ability to canonicalize heterogeneous events identifying when different labels actually represent the same underlying event.

Let us discuss the alignment $X_1 - X_4$. The significant disparity between the literal Accuracy Score, 0.4583, and the Greater Accuracy Score, 0.7222, demonstrates the pipelines success. It correctly aligns logistical intent even when label descriptions vary. Similarly seen in the alignment $X_4 - Z_4$.

Table 4.5: Carrier Accuracy Comparison. Parameter weights used were $w_A = 75$, $w_t = 100$, $w_T = 0.01$ and $w_f = 10$.

Alignment	Accuracy Score	Abstract Accuracy Score	Top 3 Accuracy Score	Top 3 Abstract Accuracy Score
X_1 - X_2	0.8681	0.9167	[1.0, 1.0, 0.8889]	[1.0, 1.0, 1.0]
X_1 - X_4	0.4583	0.7222	[0.5556, 0.5556, 0.5556]	[0.7778, 0.7778, 0.7778]
X_1 - Z_1	0.5833	0.6389	[0.7778, 0.6667, 0.6667]	[0.7778, 0.7778, 0.7778]
X_4 - Z_4	0.6071	1.0	[0.7142, 0.5714, 0.5714]	[1.0, 1.0, 1.0]
X_3 - Z_4	0.6667	0.9047	[0.8571, 0.7143, 0.7143]	[1.0, 1.0, 0.8571]

4.3.3 Example Alignments between three Carriers

While pairwise alignment provides a direct mapping, testing the algorithm across three carriers serves as a rigorous validation of the algorithm's internal consistency. The transitive property serves as a logical anchor. If Carrier A aligns with B, and B with C, the relationship between A and C should be predictable. In this pipeline, the transitive property is utilized as a form of "collaborative filtering" where the information from a third carrier is used to refine and verify the accuracy of the original pair. The best three-carrier alignment is determined by maximizing a composite score that accounts for both the pairwise accuracy (based on label descriptions) and the transitive overlap. This ensures that the concluded alignment is not only geometrically sound but also logically consistent across the entire carrier ecosystem. Total Accuracy is the sum of the Accuracy Scores and Transitive Consistency for one ordering of alignments between three carriers (For alignment A - B - C the Total Accuracy Score is the sum of the Accuracy Scores A - B, B - C, A - C and their Transitive Consistency).

The collapse to a transitive accuracy of 0 in $Z_1 - Z_4 - X_2$ in Table 4.9 is informative. This occurs when the intermediary carrier's 'sink' cluster (identified in Section 4.2.4) breaks the chain of evidence, reinforcing the necessity of the sink isolation logic to prevent false-positive alignments.

Table 4.6: Alignments between carriers X_1, X_2 and X_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$

A - B - C	A - B	B - C	A - C	Transitive Consistency	Total Accuracy
X_4 - X_1 - X_2	0.5556	0.3333	0.7778	0.5556	2.2223
X_4 - X_2 - X_1	0.5714	0.8889	0.5714	0.8571	2.8889
X_2 - X_1 - X_4	0.8889	0.5556	0.5556	0.7778	2.7778
X_2 - X_4 - X_1	0.5556	0.5714	0.8889	0.3333	2.3492
X_1 - X_2 - X_4	1.0	0.5556	0.5556	0.7778	2.8889
X_1 - X_4 - X_2	0.5556	0.5714	1.0	0.3333	2.4603

Table 4.7: Alignments between carriers X_1, Z_1 and Z_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$

A - B - C	A - B	B - C	A - C	Transitive Consistency	Total Accuracy
Z_4 - X_1 - Z_1	0.5556	0.3333	0.7778	0.5556	2.2222
Z_4 - Z_1 - X_1	0.7778	0.6667	0.5556	0.4444	2.4444
Z_1 - X_1 - Z_4	0.6667	0.4444	0.7143	0.2857	2.1111
Z_1 - Z_4 - X_1	0.7143	0.5556	0.6667	0.6667	2.603
X_1 - Z_1 - Z_4	0.333	0.7143	0.4444	0.3333	1.8254
X_1 - Z_4 - Z_1	0.4444	0.7778	0.3333	0.6667	2.222

Table 4.8: Alignments between carriers X_3, Z_1 and Z_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$

A - B - C	A - B	B - C	A - C	Transitive Consistency	Total Accuracy
Z_4 - X_3 - Z_1	0.5556	0.5714	0.7778	0.4444	2.3492
Z_4 - Z_1 - X_3	0.7778	0.7143	0.5556	0.5556	2.603
Z_1 - X_3 - Z_4	0.7143	0.7143	0.7143	0.5714	2.7143
Z_1 - Z_4 - X_3	0.7143	0.5556	0.7143	0.7143	2.6984
X_3 - Z_1 - Z_4	0.5714	0.7143	0.7143	0.7143	2.7143
X_3 - Z_4 - Z_1	0.7143	0.7778	0.5714	0.7142	2.7778

In summary, the results demonstrate that by isolating logistical noise through heuristic mining and clustering shipment cycles via structural kernels it is possible to achieve some-what high fidelity alignments across heterogeneous carriers.

Table 4.9: Alignments between carriers X_2, Z_1 and Z_4 with $w_A = 75$, $w_t = 10$, $w_T = 0.01$ and $w_f = 0.01$

A - B - C	A - B	B - C	A - C	Transitive Consistency	Total Accuracy
Z_4 - X_2 - Z_1	0.6667	0.3333	0.7778	0.5556	2.3333
Z_4 - Z_1 - X_2	0.7778	0.6667	0.6667	0.2222	2.3333
Z_1 - X_2 - Z_4	0.6667	0.5556	0.7143	0.1429	2.079
Z_1 - Z_4 - X_2	0.7143	0.6667	0.6667	0	2.048
X_2 - Z_1 - Z_4	0.3333	0.7143	0.5555	0.2222	1.8254
X_2 - Z_4 - Z_1	0.5556	0.7778	0.3333	0.7777	2.4444

4.3.4 Representative Case Study: Alignment between X_2, Z_1 and Z_4

As stated, alignments provide structured suggestions for node-to-node mapping. In this section, the alignments between X_2, Z_1, and Z_4 are evaluated. The specific alignments considered are $X_2 - Z_4$ and $Z_4 - Z_1$, as this combination yielded the highest cumulative accuracy when summing the top accuracy scores together with transitive consistency (Table 4.9). Figure 4.10 and Figure 4.11 illustrate these node-to-node alignments. Furthermore, Table 4.10, Table 4.11, Table 4.15 and Table 4.14 specify the Centiro descriptions attributed to each node and define whether the corresponding alignment is considered valid. Both the $X_2 - Z_4$ and $Z_4 - Z_1$ alignments successfully and correctly map the operational boundary nodes: "Shipment Information Received" and "Delivered".

According to the $X_2 - Z_4$ mapping, node K_4 (labeled "Unharmonized") matches with both nodes B_2 and B_3, which share the description "Departed from Hub" (see Table 4.10). This correlation strongly implies that K_4 is a "Departed from Hub" operational event. This conclusion is further supported by the structural topology, as both B_2 and B_3 connect directly to that specific node rather than any surrounding nodes, despite being relatively close in temporal proximity. Additionally, alignment $Z_4 - Z_1$ aligns K_4 with $H2$ ("Arrived at Hub") in Table 4.14. Both "Arrived at Hub" and "Departed from Hub" are "Hub Processing" events. Furthermore, analyzing the $X_2 - Z_4$ alignment through abstract descriptions, as detailed in Table 4.13 and Table 4.12, reveals that two of the previously misaligned nodes appear correct under generalized categories. This indicates that at a more abstract level the alignment is successful.

Notably, "Dropped off at Service Point" is not mapped to any abstract description. To draw further definitive conclusions, a deeper understanding of this specific operational event is required. In this alignment, node K_7 ("Dropped off at a Service Point") aligns with B_7 ("Delivered to Collection Points"), corresponding to the abstract label Last-mile Delivery." Interestingly, when examining the alignment between Z_4 and Z_1, nodes K_7 and K_8 both carry the Dropped off at Service Point" description and map directly to a "Delivery" node, as shown in Table 4.15. This pattern heavily suggests that Dropped off at a Service Point" serves as a functional component of

"Last-mile Delivery". Additionally, the "Unharmonized" node B_8 aligns with K_7.

The abstract versions of Tables 4.14 and 4.15 are provided in Appendix A.1.3.

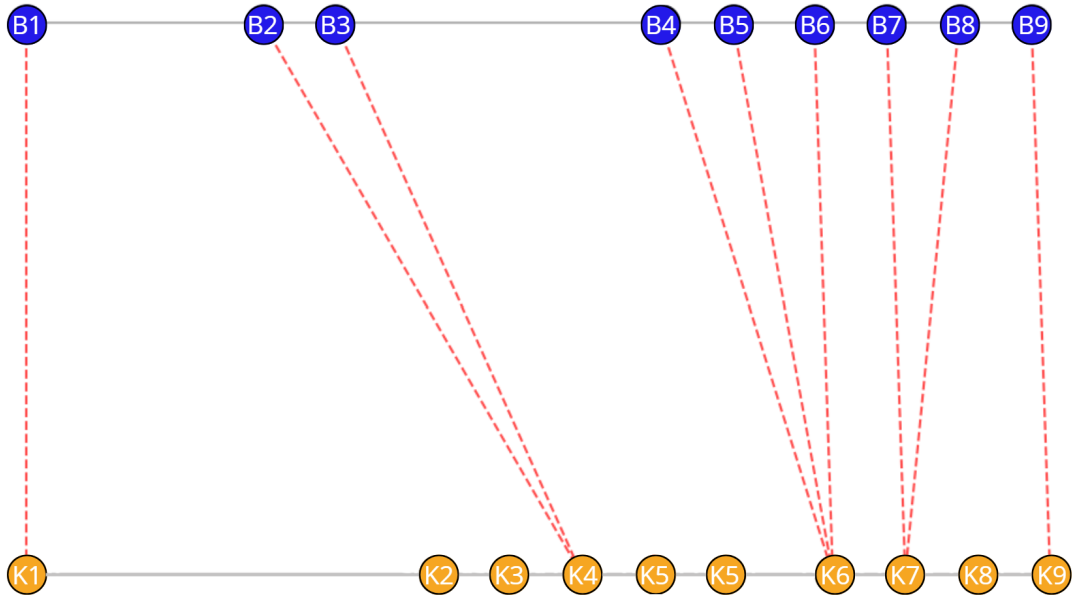


Figure 4.10: Node-to-node Alignment between carriers X_2 and Z_4.

Table 4.10: Successfully Paired nodes for alignment $X_2 - Z_4$.. References alignment in Figure 4.10.

Label	Centiro Description	Paired With	Paired Description
$B1$	Shipment Info Received	$K1$	Shipment Info Received
$B2$	Departed from Hub	$K4$	Unharmonized
$B3$	Departed from Hub	$K4$	Unharmonized
$B8$	Unharmonized	$K7$	Dropped off at Service Point
$B9$	Delivered	$K9$	Delivered

Table 4.11: Flagged node alignments for alignment $X_2 - Z_4$. References alignment in Figure 4.10.

Label	Centiro Description	Paired With	Paired Description
$B4$	Arrived at Hub	$K6$	Delivered
$B5$	Out for Delivery	$K6$	Delivered
$B6$	Delivered to Collection Point	$K6$	Delivered
$B7$	Delivered to Collection Point	$K7$	Dropped off at Service Point

Table 4.12: Successfully Paired Labels using the abstract descriptions. References alignment in Figure 4.10. Crucially, "Dropped off at Service" are not attributed to any abstract description.

Label	Centiro Description	Paired With	Paired Description
$B1$	Shipment Info Received	$K1$	Shipment Info Received
$B2$	Hub Processing	$K4$	Unharmonized
$B3$	Hub Processing	$K4$	Unharmonized
$B5$	Last-mile Delivery	$K6$	Last-mile Delivery
$B6$	Last-mile Delivery	$K6$	Last-mile Delivery
$B8$	Unharmonized	$K7$	Dropped off at Service Point
$B9$	Last-mile Delivery	$K9$	Last-mile Delivery

Table 4.13: Flagged Labels using the abstract descriptions. References alignment in Figure 4.10.Crucially, "Dropped off at Service" are not attributed to any abstract description.

Label	Centiro Description	Paired With	Paired Description
$B4$	Hub Processing	$K6$	Last-mile Delivery
$B7$	Last-mile Delivery	$K7$	Dropped off at Service Point

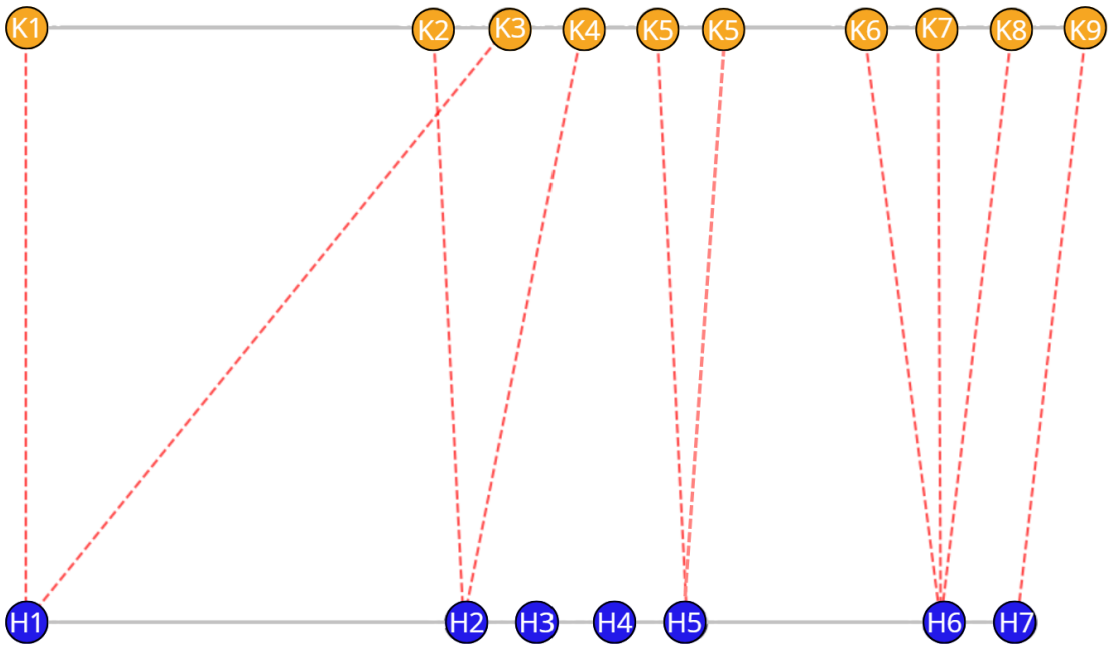


Figure 4.11: Node-to-node Alignment between carriers Z_4 and Z_1.

Table 4.14: Successfully Paired nodes for alignment $Z_4 - Z_1$. References alignment in Figure 4.11.

Label	Centiro Description	Paired With	Paired Description
$K1$	Shipment Info Received	$H1$	Shipment Info Received
$K2$	Arrived at Hub	$H2$	Arrived at Hub
$K3$	Unharmonized	$H1$	Shipment Info Received
$K4$	Unharmonized	$H2$	Arrived at Hub
$K5$	Unharmonized	$H5$	Unharmonized
$K6$	Delivered	$H5$	Delivered
$K9$	Delivered	$H6$	Delivered

Table 4.15: Flagged node alignments for alignment $Z_4 - Z_1$. References alignment in Figure 4.11.

Label	Centiro Description	Paired With	Paired Description
$K7$	Dropped off at Service Point	$H6$	Delivered
$K8$	Dropped off at Service Point	$H6$	Delivered

5
Conclusion

This chapter summarizes fulfilled thesis objectives and presents potential areas for further studies.

5.1 Summary of findings and Fulfilment of the Desiderata

The desiderata are deemed fulfilled by the implementation. As a reminder, the desiderata are briefly stated again here. The algorithm should:

- Utilize process mining and clustering to identify the "canonical" routes for a given carrier, serving as the reference for alignment.
- Design an alignment method to find correspondences between different carrier datasets.
- Quantify shipment sequences deviating from the inferred canonical order.
- Developing a method to map raw, often ambiguous event names into a set of standardized phrases by analyzing their structural context within a shipment lifecycle.

The first objective is fulfilled by pairing the Weisfeiler-Lehman (WL) subtree kernel with K-means clustering to successfully transform highly dimensional graph logs into distinct operational groups. This capability is empirically supported by the relationship between K-means elbows and Average Survival Rate plateaus across multiple carriers, as summarized in Table 4.1 and illustrated in Figure 4.6. Furthermore, the results demonstrate that process-mining-based pre-processing strengthens the clustering phase. By filtering out noise, outliers, and low-frequency anomalies, the pipeline ensures that the 'golden paths' are built on actual logistic behavior. This is empirically supported by Section 4.2.3 (see Table 4.3 and Figures 4.7 and 4.8).

The second objective is achieved through the Structural Dynamic Time Warping and the subsequently developed Heuristic Refinement for Node-to-Node Mapping in Section 3.5. Examples of such mappings can be seen in Figures 4.10 and 4.11.

The third objective is fulfilled through the application of several validation metrics. Crucially, the Accuracy Score and Abstract Accuracy Score successfully validate cross-carrier event alignments against Centiro's logistical definitions, yielding a quantifiable framework to evaluate mapping fidelity. Table 4.5 shows the difference

between the Accuracy Score and Abstract Accuracy Score across several carriers. These mathematical lifts imply that the pipeline successfully bypasses superficial naming variations to capture the underlying, true 'logistical intent' of the tracking cycles. Furthermore, the inclusion of parameter weights in the developed heuristic enables better-performing alignments (seen in Table 4.4).

The last objective is achieved though the reasoning in Section 4.3.4. By investigating exact node-to-node alignments one can identify matching to "Unharmonized" nodes. Those node-to-node alignments could infer standardized node labels.

5.2 Limitations of current pipeline

The current pipeline faces two primary limitations. First, the node-to-node alignment heuristic relies on a greedy selection mechanism that is order-dependent. Additionally, this introduces a risk of a dominant 'sink' node. Second, because the Weisfeiler-Lehman kernels do not incorporate temporal data into their feature vectors, the resulting clusters ignore valuable time indicators. Furthermore, the clustering mechanism can group data by level of entropy rather than underlying process information, resulting in an artificial 'sink' cluster.

5.3 Further Studies

Due to time constraints, certain limitations were placed on the scope of this thesis. To build upon the framework developed in this thesis, here are several potential avenues for future investigation:

- The current conversion between Causal matrices and Petri nets lacks reliability. Developing a more robust conversion framework would allow for flexible fitness calculations. This would enable greater control over all subsequent methods dependent on survival rates, such as clustering, pre-processing for cluster quality, and sensitivity analyses.
- The pipeline occasionally relies on manual tuning, particularly regarding the hyperparameters of imported Python packages such as in Section 3.5, which impact the outcome of the methods. These parameters are often proportional to the underlying data characteristics. Future work should focus on automating this process so that parameters adapt dynamically to datasets. Additionally, the consequences of suboptimal parameter selection, both in the current implementation and the automated case, should be systematically investigated.
- The current approach when separating events with identical temporal values, seen in Section 3.5.1, relies on arbitrary ordering. Future work should investigate deterministic alternatives, such as implementing a method that selects the most frequent behavior.
- The Weisfeiler-Lehman (WL) kernels used in this study do not incorporate temporal dimensions during the clustering step. Exploring temporal variants

of the WL kernel could significantly improve cluster accuracy by putting more focus on process relationships.

- The greedy heuristic used to transform the Structural Dynamic Time Warping window-to-window mapping into a node-to-node mapping introduces an unwanted order dependency. This greedy method could be replaced with a global optimization method, such as the Hungarian algorithm or maximum weight bipartite matching, to avoid this problem.

Bibliography

[1] W.M.P. van der Aalst (2016). Process Mining: Data Science in Action. Second Edition. Springer-Verlag Berlin Heidelberg, Berlin.

[2] F. Pedregosa, G. Varoquaux, A. Grama, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), pp. 2825-2830.

[3] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt (2011). *Weisfeiler-Lehman Graph Kernels*. Journal of Machine Learning Research, 12(77), pp. 2539-2561.

[4] M. Stauffer, P. Maergner, A. Fischer, R. Ingold and K. Riesen (2018). Offline Signature Verification using Structural Dynamic Time Warping. In: *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. pp. 1–8.

[5] A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves De Medeiros (2006). *Process mining with the HeuristicsMiner algorithm*, BETA publicatie: working papers, Vol. 166, Technische Universiteit Eindhoven, Eindhoven.

A

Appendix 1

A.1 Appendix regarding Results

A.1.1 Multi-carrier Generalization: Average Survival Rates with K-Means Clustering

Here follows some additional figures supporting data and conclusions in Section 4.2.2. Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier X_2 reveals a clear plateau for 5 to 6 clusters which supports its unclear elbow in Figure A.2.

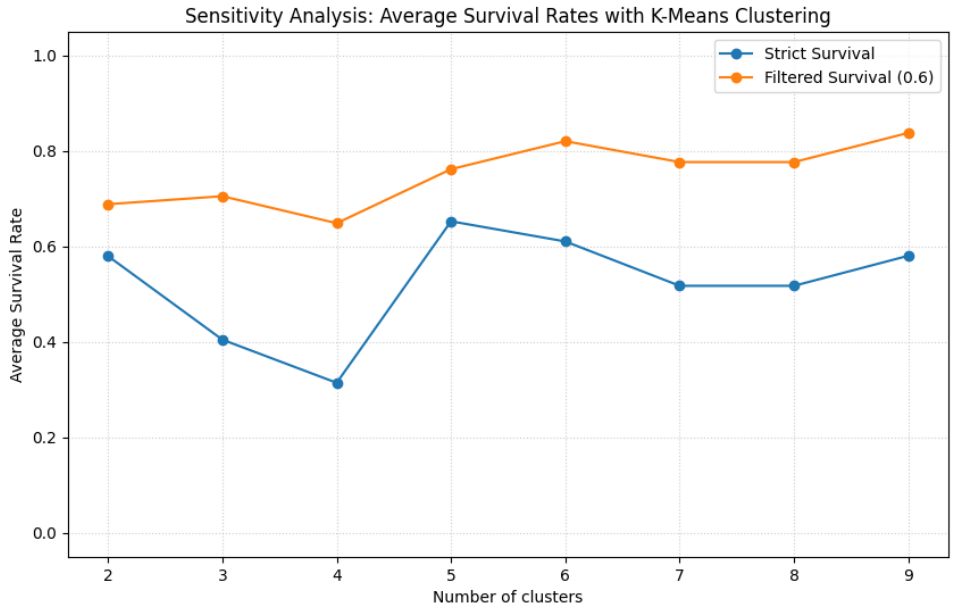


Figure A.1: Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier X_2. The distinct plateau observed between $K = 5$ and $K = 6$ validates the optimal cluster selection. Filtering threshold was set at 0.6.

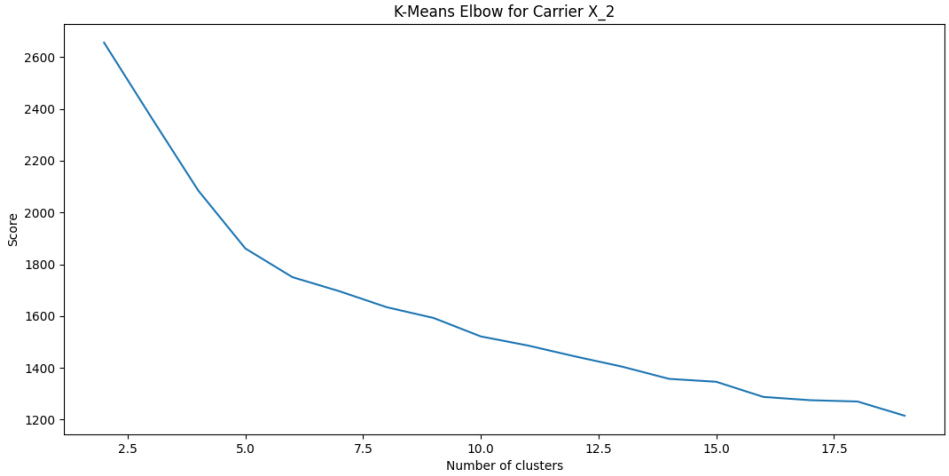


Figure A.2: K-Means elbow for carrier X_2.

Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier Z_1 shows little difference in Survival Rates. Drawing conclusion on the optimal number of clusters from elbow for carrier Z_1 in Figure A.4 is not possible; supporting the necessity for the relevant part of the pipeline. Furthermore, the overall high Survival Rates logically does seem to correspond to the fast that Figure A.4 have 2 possible elbows.

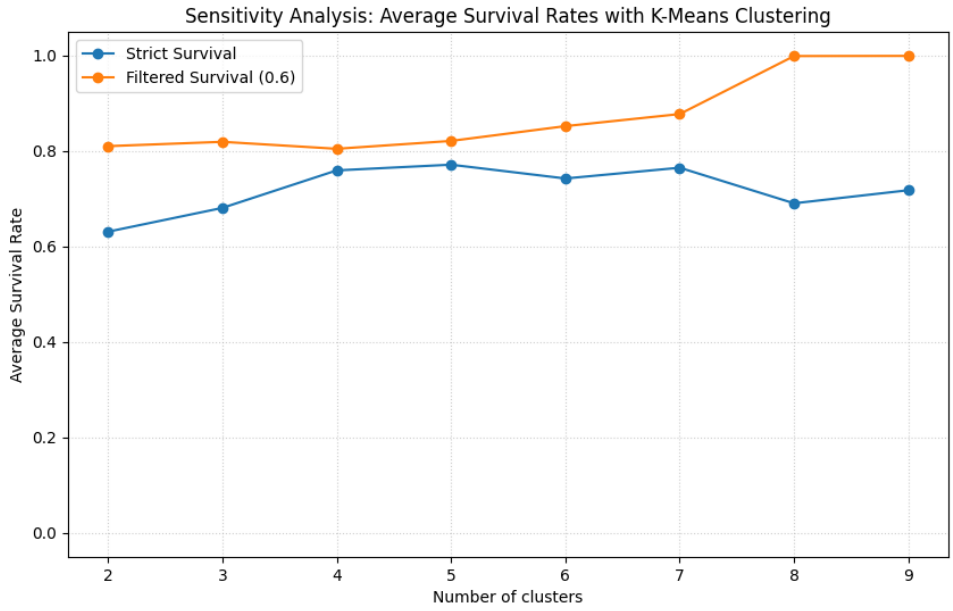


Figure A.3: Sensitivity analysis evaluating Average Survival Rates against the number of K-Means clusters for carrier Z_1. The distinct plateau observed between $K = 5$ and $K = 6$ validates the optimal cluster selection. Filtering threshold was set at 0.6.

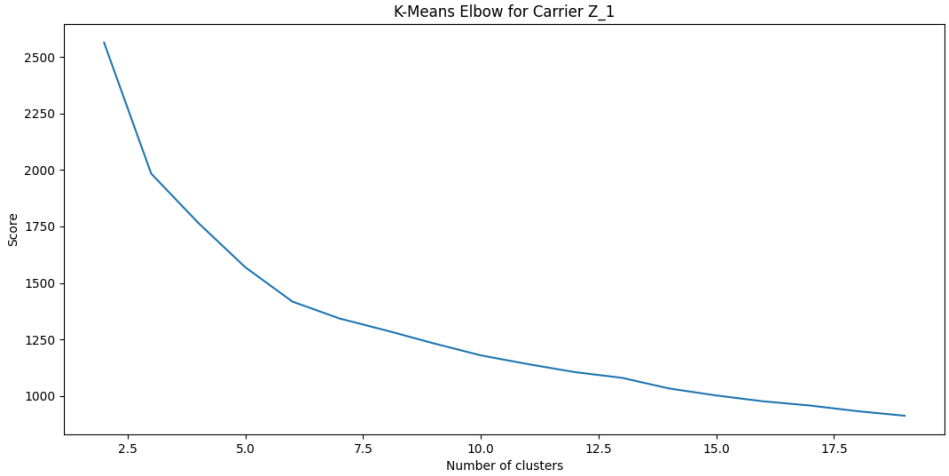


Figure A.4: K-Means elbow for carrier X_1.

A.1.2 Performance

The figures and tables provided in this appendix are intended to offer an intuitive perspective on the structural alignment results. Figure A.5 illustrates an example of an alignment between three distinct carriers, while Tables A.6, A.7, and A.8 provide representative examples of the node-to-node correspondences generated by the model. These samples are included to help the reader visualize the qualitative performance of the alignment framework.

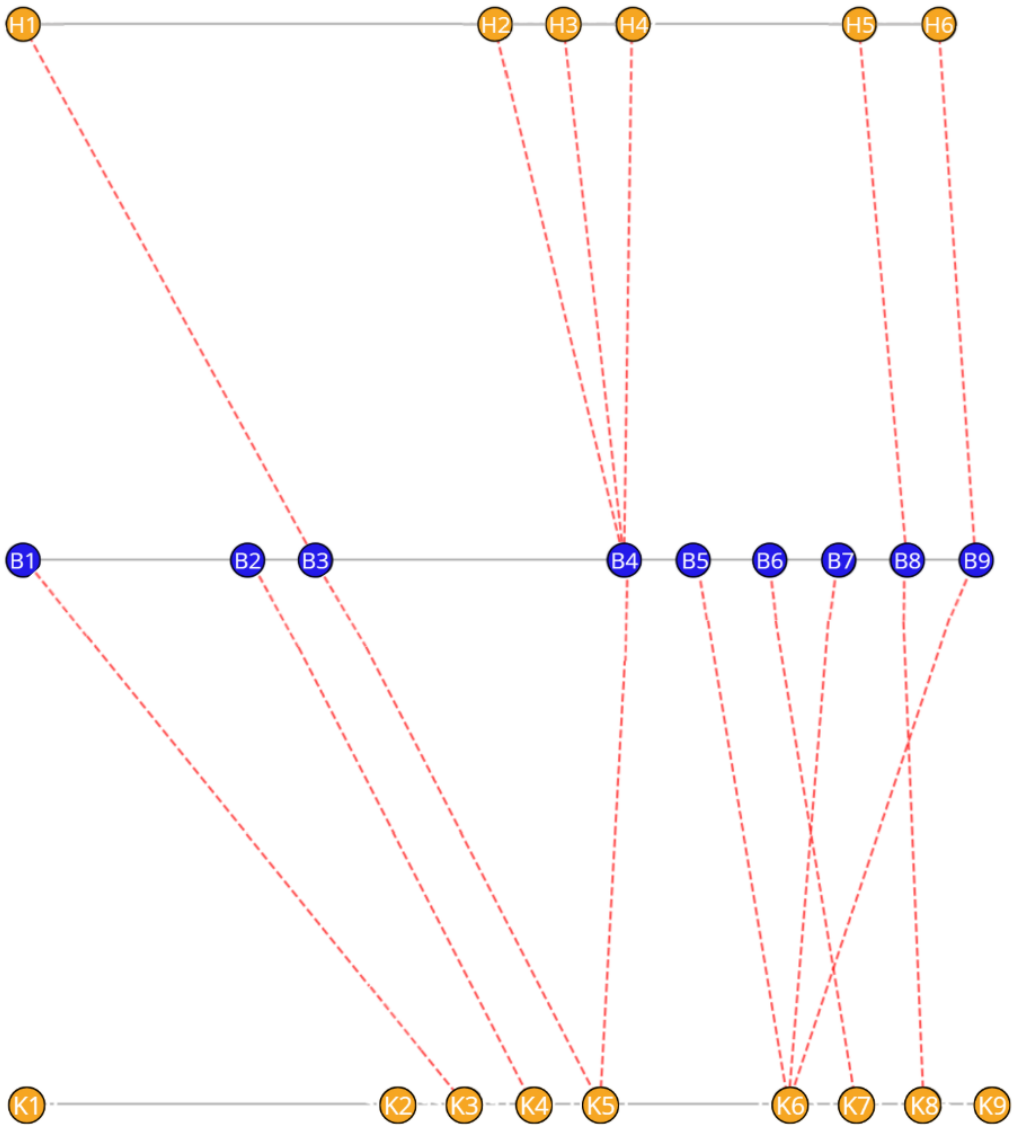


Figure A.5: Alignment between Z_1, X_2 and Z_4 maximizing their total Accuracy Score and Transitive Consistency.

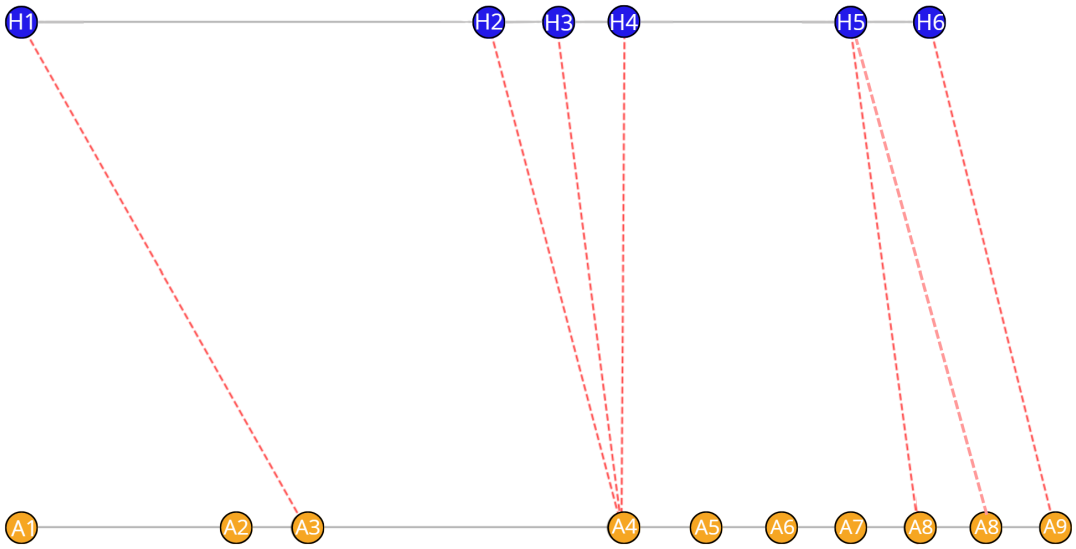


Figure A.6: Alignment between Z_1 and X_1. The alignment is the alignment with highest Accuracy Score for all aligned 'golden paths' between those 2 carriers.

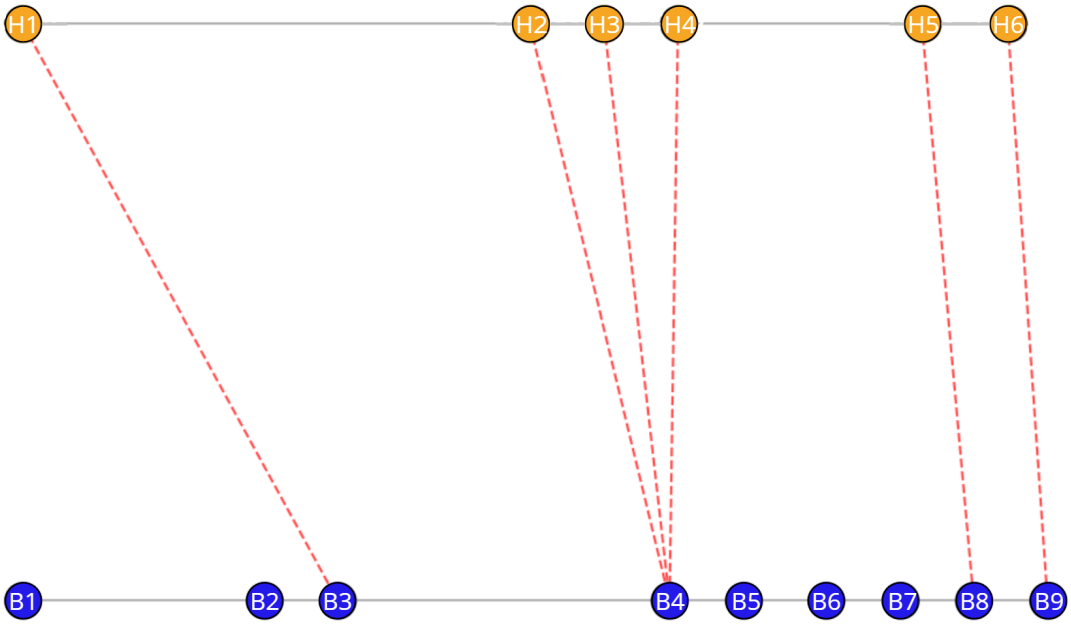


Figure A.7: Alignment between Z_1 and X_2. The alignment is the alignment with highest Accuracy Score for all aligned 'golden paths' between those 2 carriers.

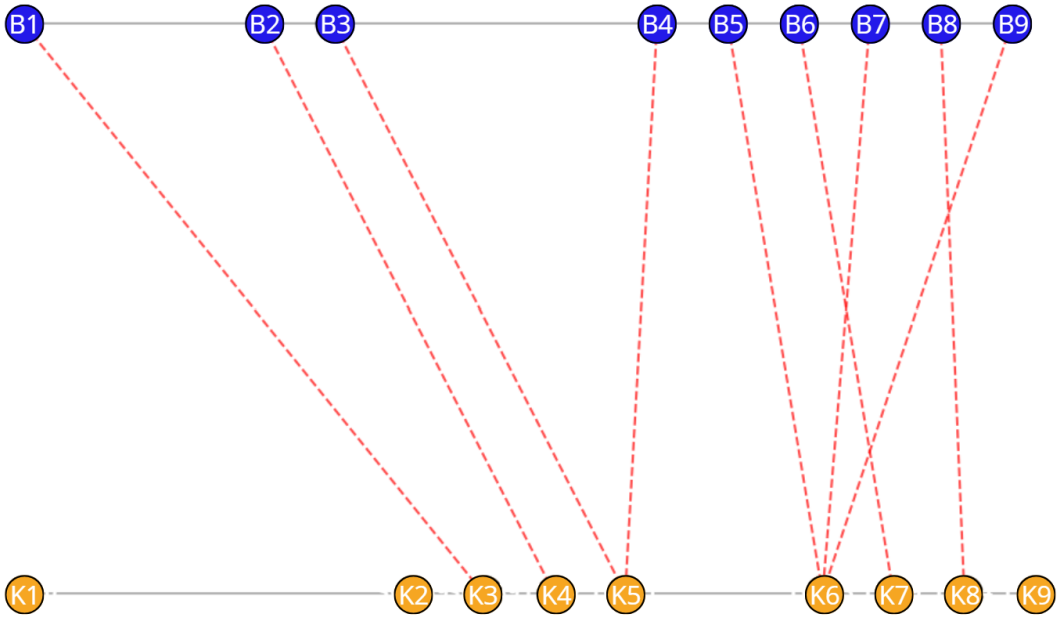


Figure A.8: Alignment between X_2 and Z_4. The alignment is the alignment with highest Accuracy Score for all aligned 'golden paths' between those 2 carriers.

A.1.3 Representative Case Study: Alignment between X_2, Z_1 and Z_4

The abstract pairings for alignment between Z_4 and Z_1. It provides little additional information in comparison to normal descriptions.

Table A.1: Successfully Paired Labels. References alignment in Figure 4.11.

Label	Centiro Description	Paired With	Paired Description
$K1$	Shipment Info Received	$H1$	Shipment Info Received
$K2$	Hub Processing	$H2$	Hub Processing
$K3$	Unharmonized	$K4$	Shipment Info Received
$K4$	Unharmonized	$H2$	Hub Processing
$K5$	Unharmonized	$H5$	Unharmonized
$K6$	Last-mile Delivery	$H5$	Last-mile Delivery
$K9$	Last-mile Delivery	$H6$	Last-mile Delivery

Table A.2: Flagged Labels. References alignment in Figure 4.11.

Label	Centiro Description	Paired With	Paired Description
$K7$	Dropped of at Service Point	$H6$	Last-mile Delivery
$K8$	Dropped of at Service Point	$H6$	Last-mile Delivery

For the three part alignment here is analysis under the assumption that transitive

property holds. Note that in Table A.4 there is only one alignment considered invalid.

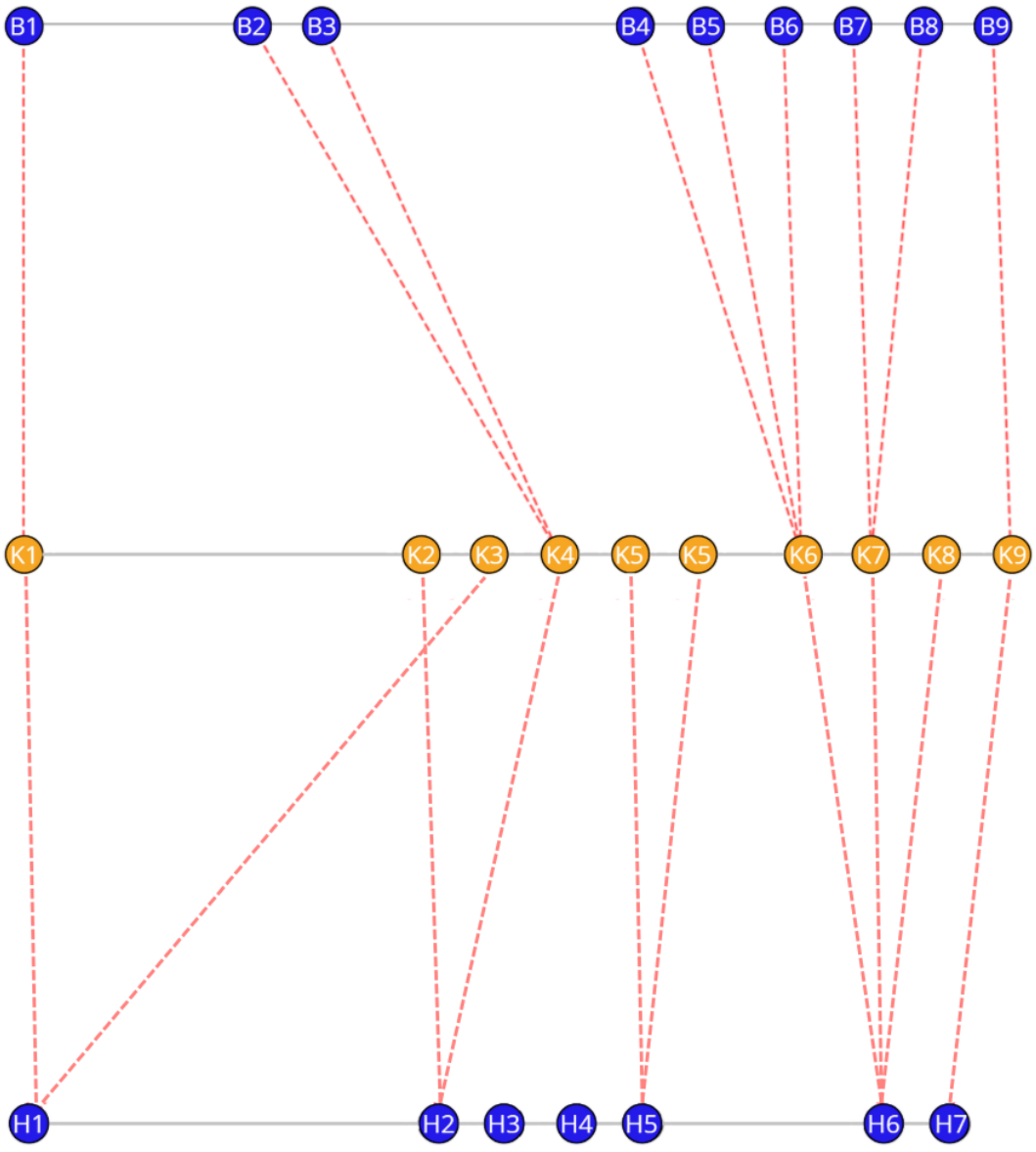


Figure A.9: Alignment between Carriers X_2, Z_1 and Z_4.

Table A.3: Carried over. References alignment in Figure A.9.

Label	Centiro Description	Paired With	Paired Description
$B1$	Shipment Information Received	$H1$	Shipment Information Received
$B2$	Departed from Hub	$H2$	Arrived at Hub
$B3$	Departed from Hub	$H2$	Arrived at Hub
$B4$	Arrived at Hub	$H6$	Delivered
$B5$	Out for Delivery	$H6$	Delivered
$B6$	Delivered to Collection Point	$H6$	Delivered
$B7$	Delivered to Collection Point	$H6$	Delivered
$B8$	Unharmonized	$H6$	Delivered
$B9$	Delivered	$H7$	Delivered

Table A.4: Carried over. References alignment in Figure A.9.

Label	Centiro Description	Paired With	Paired Description
$B1$	Shipment Information Received	$H1$	Shipment Information Received
$B2$	Hub Processing	$H2$	Hub Processing
$B3$	Hub Processing	$H2$	Hub Processing
$B4$	Hub Processing	$H6$	Last-mile Delivery
$B5$	Last-mile Delivery	$H6$	Last-mile Delivery
$B6$	Last-mile Delivery	$H6$	Last-mile Delivery
$B7$	Last-mile Delivery	$H6$	Last-mile Delivery
$B8$	Unharmonized	$H6$	Last-mile Delivery
$B9$	Last-mile Delivery	$H7$	Last-mile Delivery

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY