Context-Aware Performance Deviation Analysis for Logistics Systems

Boender, J.M.A.

Award date:
2020

Department of Mathematics and Computer Science
Process Science Charter

Context-Aware Performance Deviation Analysis for Logistics Systems

Master Thesis

Jaimini M. A. Boender

Supervisors:
Dr. Dirk Fahland
Dr. Cinzia Cappiello
Ir. Hilda Bernard
Ir. Koen Verhaegh

European Institute of Innovation & Technology
The EIT is a body of the European Union

Eindhoven, 21-07-2020

Abstract

Vanderlande is a company that provides value-added logistic process automation solutions and is looking to use Process Mining to help offer increased value to their customers. Process Mining is the discipline of extracting value from event logs. We have identified a gap where there is currently no effective technique to abstract event logs of logistic process data. In this thesis, we develop a novel technique to abstract event logs of logistic process data. This technique uses PCA and clustering to group traces together to abstract the event log. This technique is then applied to one of Vanderlande's largest baggage handling solutions and the most interesting insights are presented. Validation of these insights are then verified with an engineer familiar with the baggage handling system and the success of the technique is then evaluated.

Contents

List of Figures

List of Tables

Abbreviations

BHS	Baggage Handling System
ID	Identification
NSD	Node Segment Diagram
PCA	Principle Component Analysis
POLIMI	Politecnico di Milano
TU/e	Eindhoven University of Technology

Acknowledgements

To all my teachers, thank you.

1 Introduction

1.1 Context

Vanderlande is a company that provides value-added logistic process automation solutions. They are major players in the parcel and warehouse industries and the undisputed global market leader in providing baggage handling systems to airports. In airports, their equipment moves more than 3.7 billion pieces of luggage per year and they have baggage handling systems active in over 600 airports: 13 of which are in the world's largest 20. To remain the market leader, it is critical for Vanderlande to have state-of-the-art techniques in order to analyze their processes and add value to their customers. Due to the increasing size and complexity of not only the baggage handling solutions but also the software solutions - there is an increasing demand for more advanced services.

Since a baggage handling system can be seen as a collection of processes, process mining is an appropriate discipline to understand and analyze these systems. However, baggage handling system processes are not exactly the same as typical information handling system processes for the following reasons:

- Many different paths can reach the same destination.
- The number of process steps are magnitudes higher than typical information handling system data.

Because of these key differences to typical information handling systems processes, we will refer to these processes of logistics systems as logistic processes. Because we are dealing with logistic processes many typical process mining solutions are simply not applicable.

1.2 Research Questions

Vanderlande is interested in analyzing the routes in their system to gain insights. However, there are too many to do this effectively. The proposed solution to this is to cluster routes together to allow proper analysis. Thus the main research question trying to be answered here is:

Can we create a route clustering technique that works on logistic process data and clusters routes meaningfully?

Already this is a pretty vague question and a large part is the fact that the word "meaningful" is quite subjective. So we interpret meaningful to be 2 different things: unusual routes and fast/slow routes. To formalize our question we break it down into two smaller questions that are more specific and precise.

RQ1: Can we create a clustering technique that allows us to identify unusual routes or abnormal behavior?

RQ2: Can we create a clustering technique that allows us to identify optimal, sub-optimal and non-optimal routes?

1.3 State of the Art

Process Mining is the discipline of extracting value from process execution using event logs [13],[8]. Event logs are the records of activities that are assumed to be meaningful to a particular process. In Process Mining, application usually is focused on one of three main areas: process discovery, process conformance and process enhancement [9].

In Process Mining, clustering routes has been done before, however the techniques tend to focus on process discovery [6][7]. Instead of process discovery, our interest is in process enhancement

since the process model is already known. Within process enhancement, our aim is log abstraction. Through log abstraction, we will be able to cluster traces that result in the same abstracted form and answer the relevant research questions. There are several different existing techniques for log abstraction in process mining [14],[15]. However, these rely on assumptions that do not hold on logistic processes. Thus, what we are missing is a log abstraction technique that works on logistic processes.

1.4 Methodology

CRISP-DM

The methodology chosen for this thesis is known as CRISP-DM (Cross Industry Standard Process for Data Mining). The steps for CRISP-DM are shown in the following diagram.

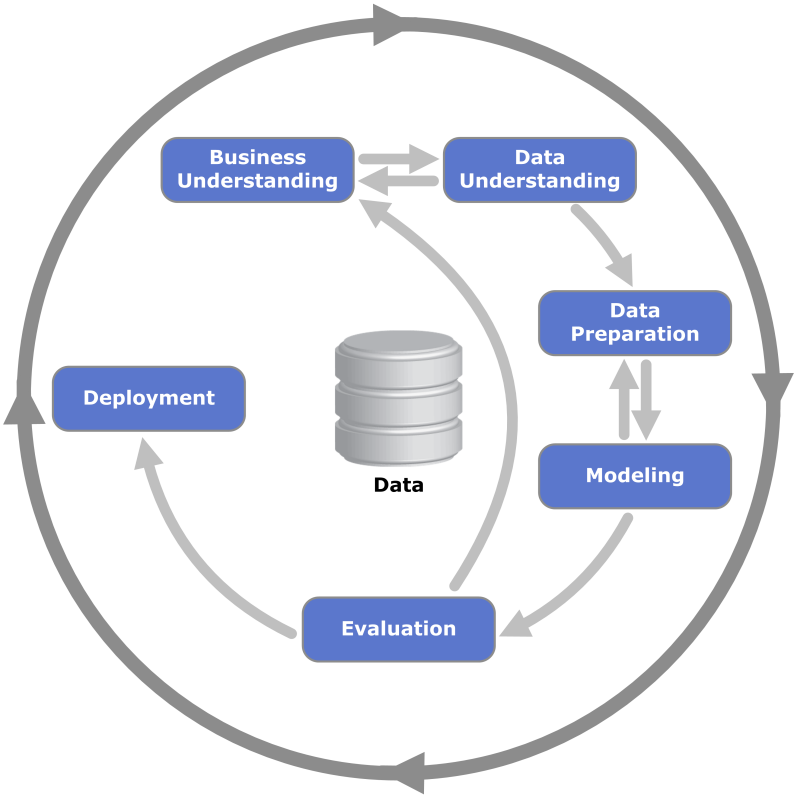

Figure 1: Diagram of CRISP-DM [3]

As seen in the diagram, the CRISP-DM methodology consists of 6 major steps. Each step is explained below.

Business understanding is the step in which we determine the business objectives. This involves assessing the situation and determining the score of the thesis. In the end, the goals of the thesis should be defined.

The next step is data understanding. In data understanding, data is collected, explored and the quality verified. In the diagram, there is an arrow going back to business understanding. This arrow is to illustrate that data understanding can affect and improve business understanding. These two steps (business understanding and data understanding) are covered in Chapter 3 of this thesis.

In the following step, data preparation is the focus. Here, data is cleaned and formatted to the desired shape. This is covered in Chapter 5.

After that is the modeling step. In modeling, the technique is designed and built. In the diagram, there is an arrow going back to data preparation. The reason being that, as a model is developed, new requirements for the data format can be defined. Thus, going back to the previous step could be necessary. In our case, the technique being created and designed will be our clustering technique. This is covered in chapters 6-8, which is the heart of this thesis.

The step after this is evaluation. Evaluation is the step where the results are analyzed and processed. From evaluation, it is possible to return to the business understanding step. The reason is that new insights gained from evaluation might lead to a better understanding of business objectives. And with the business objectives updated, we then we can repeat the whole process. This is why CRISP-DM is an iterative process and makes the methodology adaptable to new understanding. The final evaluation is covered in Chapter 9 and 10.

Finally, the last step is deployment. In deployment, the technique is finalized and a plan to deploy is created. This includes a way to monitor the technique and also maintain it if necessary. In this thesis we do not cover deployment.

CRISP-DM was chosen because it is an accessible and iterative technique. Starting with the business objectives allows participants with little data science experience but strong business domain experience to participate. With this thesis being a collaboration effort with Vanderlande, this aspect was particularly valuable. Further, because CRISP-DM is iterative, the business objectives can evolve and adapt as more understanding is uncovered by going through the process. This makes sure that the business objectives are constantly accurate.

1.5 Findings

To evaluate the resulting clusters of the clustering technique, we required the opinion of a domain expert. Since for clusters to be useful (whether to determine optimal/sub-optimal, or unusual routes), they need to be meaningful to a domain expert in order to have any value at all. Below we discuss the results we obtained and the interpretation given by the engineer.

1.5.1 Results

The domain expert we chose was an engineer familiar with the baggage handling system that was analyzed. An engineer was chosen because they would be the likely end-user of a tool containing our clustering technique. Thus, their opinion on what they would consider meaningful would carry the most weight with regards to creating business value within Vanderlande.

To obtain results, several of the most interesting clusters were presented to the engineer. Comments and interpretations from the engineer were recorded and noted.

1.5.2 Interpretation

The engineer was able to use the different clusters and come up with several theories as to why they could have been identified as interesting or why they could be causing problems. We consider this a success because this was the expected reaction if the clusters were meaningful. If clusters were not meaningful no explanation could be conceived.

However, none of these theories could be confirmed due to the need for additional context information. This additional information included data like equipment availability, load on a specific route and log information on operator presence. In short, root-cause analysis for these clusters was not possible due to missing additional context information and would be a desirable next step.

Further, it was discovered that the visualization developed to visualize the clusters only works well when looking at a specific route. In other words, it lacks the ability to visualize clusters of many different routes simultaneously. Thus, an improvement in this regard could lead to the identification of additional interesting clusters.

In the end, we have demonstrated how the clustering technique developed does in fact generate meaningful clusters. This has not only been confirmed by interpretations of the engineer but is

also evident by the engineer's desire for an expanded and enriched tool containing more context information on these clusters. The success of this technique demonstrates that we have not only created a novel route clustering technique in the discipline of process mining but also contributed to Vanderlande's ability to deliver value-added logistic process automation solutions.

1.6 Thesis Structure

The rest of the chapters in this thesis will explain all topics introduced here in more detail. Below we give a quick summary of what to expect in the following chapters:

- Chapter **??** will cover the current state-of-the-art techniques for log abstraction. This will introduce Process Mining as a discipline and go through the techniques that are most relevant to the work of this thesis.
- Chapter 2 will go over the external disciplines and concepts that need to be introduced. These will have to be familiar to the reader to easily follow the steps and choices described.
- Chapter 3 will go over the business motivation and how the identified usecases are driving the research questions. This includes an introduction into the system being analyzed along with a preliminary analysis and the problem statement being defined.
- Chapter 4 will give a summary of the method developed, going over the design decisions and the major steps involved in the process. The following 4 chapters will give detailed explanations of each of these steps.
- Chapter 5 will give a detailed explanation of what data pre-processing is necessary to apply the technique.
- Chapter 6 will give a detailed explanation of how the technique works and how routes are summarized to form clusters.
- Chapter 7 will give a detailed explanation of how the cluster validation is done to assure the quality of clusters.
- Chapter 8 will give a detailed explanation of how we apply this technique to the entire baggage handling system and how the results are visualized.
- Chapter 9 will go through our demonstration of answering the business use cases using our technique and visualizations. Some findings will also be presented here.
- Chapter 10 discusses the validation done with the engineer and the comments of the findings are presented.
- Chapter 11 will summarize everything by highlighting the contributions made, the capabilities of the technique and future work.

2 Preliminaries

In this chapter we cover the basic understanding of all external related disciplines and concepts used in this thesis. This involves covering the definition of process mining events in Section 2.1, introducing the concept of Principle Component Analysis in Section 2.2 and becoming familiar with the different aspects of clustering in Section 2.3.

2.1 Process Mining Events

Process Mining is the discipline of discovering, monitoring and improving processes by extracting knowledge from event logs [8]. An event log is a list of events that have occurred, containing the following information:

- Case ID: The identifier of the instance of a process
- Activity: The actions occurring in the event
- Timestamp: The time and date of when the event occurred

A case is the instance of a process, which consists of a sequence of events that have occurred. Note that 2 different cases can have the same sequence of events occur. We call this sequence of events a trace. Thus, it is possible for 2 difference cases to follow the same trace.

2.2 Principle Component Analysis

Principle Component Analysis (PCA) is a dimensional reduction technique that uses the variance-covariance structure of the data [10]. The type of data that PCA typically works the best on is data that contains variables that are interrelated. What PCA then does is identify a new set of variables (known as the principle components) that are uncorrelated and designed to retain as much of the variance as possible. The following images aim to provide an intuitive understanding.

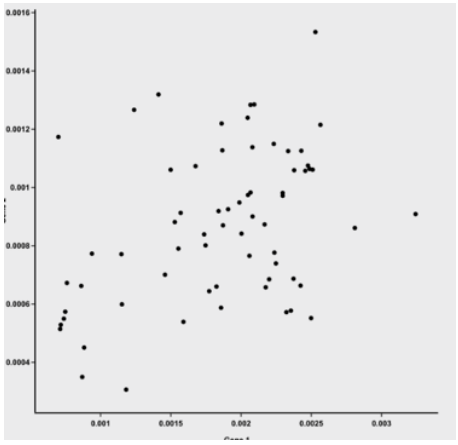

Figure 2: Regular group of points

We start our example with a group of points in a 2 dimensional space, shown here in Figure 2. What PCA does is calculate the eigen vectors and values of the covariance matrix. The details on how PCA does this and what eigen vectors are can be found in Jollife's book [10]. PCA then sorts these eigenvectors in descending order based on their eigenvalues and the top few are kept. These few that are kept are known as the Principle Components. In our example, we decide to keep the top 2 and they are highlighted in the following Figure 3.

Here in Figure 3 we can see the top 2 Principle components highlighted. Immediately it is clear in this image that these Principle Components are the dimensions that represent the most

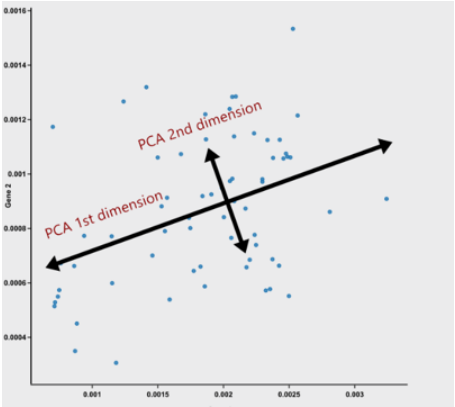

Figure 3: Principle Components Highlighted

variance in the data. This is no coincidence, and is the result of sorting the eigenvectors by their eigenvalues. The output of PCA that we are interested in is known as the loading matrix. The loading matrix is a matrix whose columns are the original variables or features of the data set and rows are the Principle Components. The values inside the loadings matrix indicate the correlation between the Principle Components and the original variables/dimensions or features. An example of this matrix is shown in Figure 4.

PCA Output

	Feature1	*Feature2*	*Feature3*	*Feature4*	*Feature5*	*Feature6*	…	*Feature n*
PC 1	.3	0	.1	.1	.1	.6	…	.3
PC 2	0	0.3	.1	0	.2	.1	…	.2
PC 3	0.1	.11	.1	.1	1	0	…	.1
…	…	…	…	…	…	…	…	…
PC m	0.2	.51	.3	.2	1	0	…	.2

Figure 4: Example output from PCA

And it is this matrix that we use in this thesis.

2.3 Clustering

Clustering is the activity of grouping unlabeled instances of data. Grouping is done using a distance metric. A distance metric is a function that gives a value to indicate how far apart two instances are. Once a distance metric is defined for a set of data then there are different strategies to use the distance metric. Below we identify the two that we use in this thesis.

2.3.1 K-means

K-means is an iterative clustering algorithm that finds K clusters (where K is a numeric parameter). A unique characteristic to K-means is that it uses the concept of a centroid to create clusters. The centroid of a cluster is the mathematical mean of all points in that cluster.

To initialize, K-means randomly picks K initial centroids. Then the following 2 steps are executed repeatedly.

- Assign points to clusters by minimizing the sum of the squared distance to the centroid of each cluster.

- Recalculate new centroids.

These steps are repeated until the centroids converge to a fixed point. More technical details on these steps can be found in Jain's paper [11].

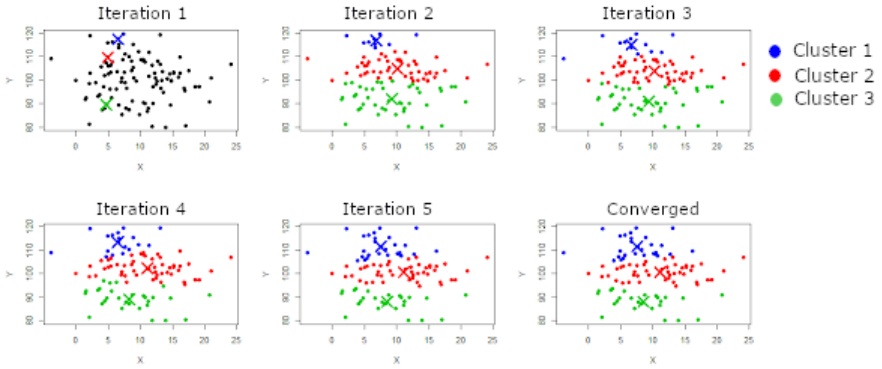

Figure 5: Example of the different iterations of K-means

An example of what the iterations could look like are shown in Figure 5. Here, 3 clusters are trying to be found. These 3 clusters are represented by the 3 different colors: blue, red and green. In the first iteration, 3 random centroids are picked. Then in the following iterations, clusters are assigned and the centroids are recalculated until convergence in the last iteration.

2.3.2 Agglomerative

Agglomerative clustering is a hierarchical clustering approach that can either take the number of desired clusters as parameter or a distance threshold. Agglomerative clustering is pleasant for two reasons: it is simple to understand and it possible to see what points the algorithm considers to be similar.

Agglomerative clustering starts by assigning every point to it's own cluster. It then repeatedly merges the two closest clusters until either the desired number of clusters is reached or the distance threshold is reached (all clusters have distance greater than the threshold). Exhaustive details on all the ways agglomerative can be used to determine which two clusters are the closest are covered in Müllner's paper [12] on agglomerative clustering algorithms.

The nice feature of agglomerative clustering is that it is possible to easily visualize the process in which points were clustered using dendograms.

In Figure 6 we see an example of a dendogram being used to visualize the order in which an agglomerative algorithm decided to merge clusters. This is incredibly useful because it allows us to understand within a cluster what point the algorithm considers to be the most similar.

2.3.3 Cluster Validity Indices

Cluster validation is an attempt at indicating how well data has been clustered (cluster quality). Since there is no known optimal clustering algorithm and many clustering algorithms are unable to naturally determine how many clusters should be formed (they must be given this as as parameter) it is a common problem to understand the quality of clusters.

External validation can be done when there are known correct clustering labels. This is where the proposed clustering by the algorithm is compared to the known correct cluster labels. However, in our situation we were not fortunate to have known correct cluster labels. Thus, we had to turn to internal validation.

In internal validation the approach is focused on the assumption that a good clustering will have members of the same cluster be similar to each other and members of different clusters be different from each other. In other words the intra-cluster distance (distance between members of

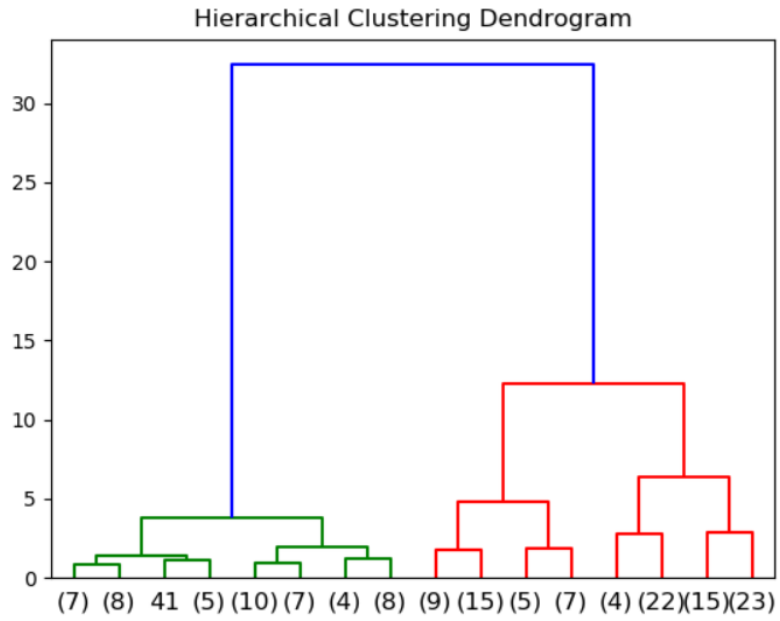

Figure 6: Example of a dendogram illustrating agglomerative clustering

the same cluster) should be minimized and the inter-cluster distance (distance between members of different clusters) maximized. A cluster validity index is a number to indicate the quality of the clustering based on this approach. Now there are different ways to calculate these and Arbelatiz paper [4] covers 30 different validity indecies. From these 30 we use 2: the Silhouette Coefficient and Dunn's Index. The reason why these two are discussed in great detail in Section 7.2.

Now that we have covered all the preliminaries, we are ready to discuss the business context which is the main motivator for this thesis.

3 Use Case

In this chapter we set the business context for this thesis. We start in Section 3.1 where we introduce the baggage handling system (BHS). We then go deeper by presenting the data associated with this system and its challenges in Section 3.2. After establishing the situation, in Section 3.3 we identify the business questions that Vanderlande finds relevant. To better understand the scope of these questions we do a quick preliminary analysis, which is explained in Section 3.4. Then armed with this understanding in Section 3.5 we finally establish a detailed description of the problem we are aiming to solve.

3.1 Baggage Handling System (BHS)

A baggage handling system is a network of conveyor belts, scanners, screening machines, storage units and additional types of equipment that transport baggage over distances. The purpose of this system is to not only transport luggage to the appropriate destination but also scan the contents (for security) and temporarily store luggage when necessary. To accomplish this the system must keep track of a variety of data for each bag: their locations, their destinations and their required processes. In the baggage handling system that was analyzed, these processes that bags complete are split into two different levels of granularity: *High-level Processes* and *Low-level Processes*. Each of these terms will be described in the next Sections.

3.1.1 High-level Process

The baggage handling system analyzed had over 2,000 individual locations and had been organized into a set of subsystems. These subsystems are smaller networks of equipment that together accomplish similar tasks. Accordingly, they also contain similar types of equipment such as a scanning subsystem containing many scanners with linear conveyor belts or a sorter containing circular conveyor belts (allowing bags to loop). Consequently, the behavior of different subsystems can vary greatly due to them containing such different equipment.

The High-level Process of a bag describes which subsystems a bag moves through in the baggage handling system. Now, we will denote these "subsystems" as *logistic steps*, since that is what they are in the context of the High-level Process. Logistic steps are the units of a High-level Process and it is the sequence of logistic steps that a bag passes through that illustrates which High-level process is being followed. To give some examples of this, let us first introduce the architecture of the baggage handling system that was analyzed in this project is shown in Figure 7.

In Figure 7:

- Rectangles represent logistic steps that contain only linear conveyor belts
- Circles indicate the presence of circular conveyor belts (looping is possible)
- Arrows represent the capability for bags to move between respective logistic steps
- Red filled rectangles represent the context of the baggage handling system we are looking at
- The input and output labels show where bags are first seen and where they are last seen in the tracking of the system

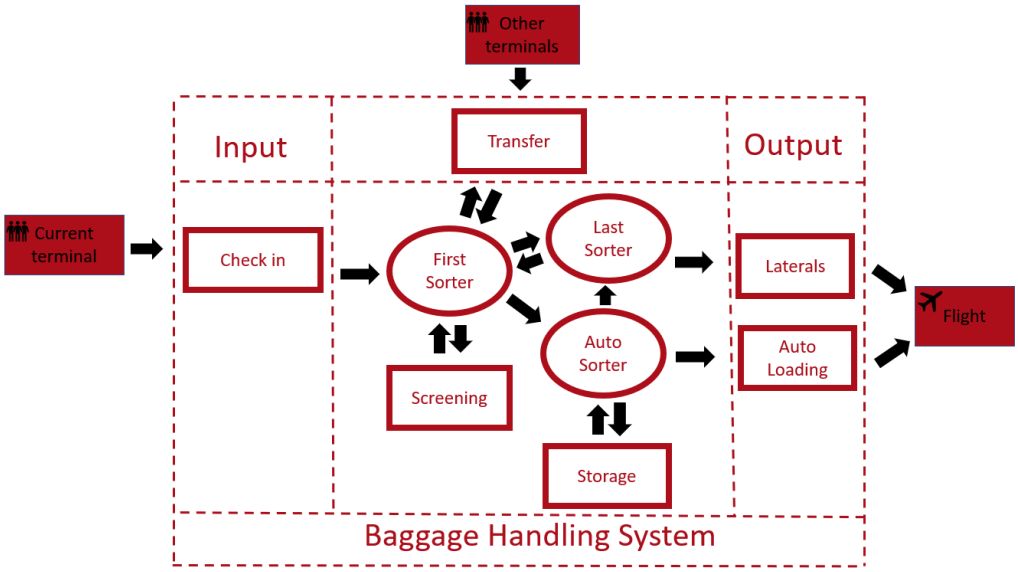

Figure 7: Architecture of baggage handling system

Now, looking at Figure 7, we can notice that the "Transfer" logistic step is the only logistic step that is both input and output. This is why it has been placed at the top, between the input and output labels. The Check-In logistic step is used just as input while the Laterals and Auto Loading logistic steps are just used as outputs.

Figure 8: Example of a High-level Bag trace going through screening

Figure 9: Example of a High-level bag trace going through screening and storage

Both Figures 8 and 9 show examples of High-level bag traces. In Figure 8 we can see a bag that started in the logistic step "Check In", went through screening and left the system at Laterals. In Figure 9, we have another variant where the bag goes through screening as well as storage. Thus, demonstrating that even though bags might start and end in the same logistic step, many different variants of that process are possible. Now, we see this is possible for how bags move through the system as a whole. However, this is also possible for how bags move within a Logistic Step. As mentioned earlier, Logistic Steps are subsystems in their own right and are complex enough to also have their own different subprocess variants. We denote this concept of subprocess within a Logistic Step as a "Low-level Process", and discuss them in detail in the next Section.

3.1.2 Low-level Process

As mentioned before, Low-level Process is a subprocess occurring within a particular Logistic step. In pragmatic terms, Low-level Processes are sequences of physical locations that a bag passes through. Thus, the units we use for a Low-level Process are *Locations*. Locations themselves are movements recorded into a log by sensors as a bag passes through the physical equipment.

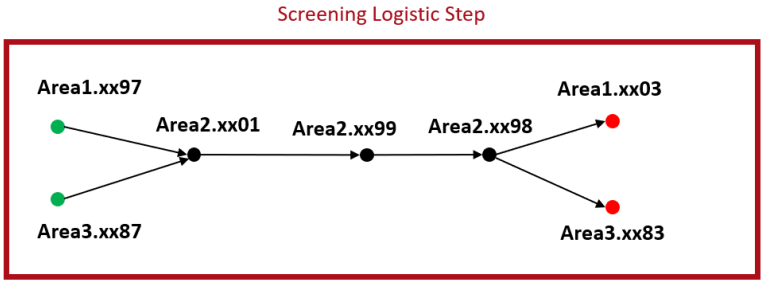

Figure 10: Example of Locations in the Screen logistic step

In Figure 10 we have an example of how the configuration of locations could be set up within the Screen logistic step. Each node in the graph represents a location. Green nodes are the starting locations (locations where bags can enter this logistic step) and red nodes are the exit locations (locations where bags can exit this logistic step). Further, each location identified and labeled by a sequence of numbers. However, in this thesis they have been anonymized.

The first several digits (before the dot) have a special meaning and are known as the "Area code". Area codes are identifiers for "Areas" which are a more fine-grained type of grouping, existing within a logistic step. Areas perform similar functions (have similar roles, equipment, etc) within a logistic step and vary greatly in granularity between logistic steps in the system. As an example, in the Screen logistic step there are 8 different areas, yet each of these areas contains only 4 locations each. While the First Sorter logistic step has only 2 areas, but each area contains 42 locations each. The rest of the digits (after the dot) indicate the zone id and equipment id. And together with the area code, they form a unique identifier for that location.

Area1.xx97 : Area2.xx01 : Area2.xx99 : Area2.xx98 : Area2.xx01 : Area1.xx03

Figure 11: Example of a Low-level Bag trace in the Screen logistic step

Area3.xx87 : Area2 .xx01 : Area 2.xx99 : Area2.xx98 : Area2.xx01 : Area3.xx83

Figure 12: Example of another Low-level bag trace variant In the Screen logistic step

Figures 11 and 12 both show examples of what a Low-level trace through the Screen logistic step would look like. Taking a close look at the examples, we can see that Figure 12 shows a different variant of a Low-level trace, that starts and ends in a different location than the trace in Figure 11. In this case, it may seem that bags move relatively similarly through the screen logistic step. However, depending on the logistic step, bags can move through very different Low-level variants in a logistic step. And for some logistic steps, the number of these Low-level variants can be in hundreds - allowing for a wide range of behavior. This is shown in detail in Section 3.4.

One important property that should be mentioned is that each Low-level sequence variant containing locations from the same High-level step (logistic step) can only be found in that logistic step. In other words, this Low-level variant is unique to that logistic step. And this is because each location belongs to just one logistic step. This might seem like a trivial property but it is important to point it out now as we will refer to it later when aggregating in Section 5.2.

Now, as mentioned before, these locations are logged as bags move past sensors. However, due to the large size of the system, only the critical locations are recorded in persistent memory. This has consequences, which are elaborated in the next section.

3.2 The Available Data

Due to the large size of the system, only certain locations are kept. Because of this, the logs of bag movements are incomplete. This is a problem since we need a log with complete bag movements in order to be able to understand what route each bag took. Thus, we need a technique that, from a partial log of bag movements, will be able to reconstruct a log containing all bag movements. To start, we introduce the fields in the incomplete log.

3.2.1 Incomplete Log

Below in Table 1, we can see an example of Locations being logged. The columns have the following meanings:

Bag id:
The unique identifier of the bag.

Location:
The unique identifier of a location in the baggage handling system.

Timestamp:
Day, Month, Year, Hour, Minute and Second that this movement occurred.

Logistic Step:
The high level process that the source node belongs to.

Bag id	Location	Timestamp	Logistic Step
36463	AREA1.xx01	27-9-2017 15:04:36	FIRST_SORT
36463	AREA2.xx01	27-9-2017 15:05:13	SCREEN
36463	AREA2.xx98	27-9-2017 15:05:40	SCREEN

Table 1: Incomplete log example

Now in this example, the incomplete log in Table 1 is the bag trace shown in Figure 11. If you look closely at Table 1 you will notice that only some of the locations of the Low-level trace shown in Figure 11 are logged. This is a problem because we do not have the full traces and have to reconstruct them.

A prior master thesis [2] reconstructed the bag movement of locations by applying Dijkstra's algorithm on the Node Segment Diagram (NSD). The NSD is a graph representation of the baggage handling system, where locations in the system are represented as nodes in the graph. Then by copying the timestamp of the previous event, the new events are inserted into the event log to obtain the complete event log. The result of that is shown in Table 2 the next section.

3.2.2 The Completed Log

Below in Table 2 we have the completed log, which shows bag movements between locations (rather than just locations with timestamps). This means our Location column has now been split into two: one for the source location and the other one for the destination location. Other than that, all other columns retain their same interpretation.

Here in Table 2 we can see the full trace from Figure 11 reconstructed. Now at this point, it was finally possible for Vanderlande to reconstruct bag traces for any particular bag and see how it traveled through the system. With the complete log forming a basis for deeper analysis, we are ready to examine Vanderlande's questions and usecases that we introduce in the next section.

Bag id	Source Location	Destination Location	Timestamp	Logistic Step
36463	AREA1.xx03	AREA1.xx97	27-9-2017 15:04:31	FIRST_SORT
36463	AREA1.xx97	AREA2.xx01	27-9-2017 15:04:36	FIRST_SORT
36463	AREA2.xx01	AREA2.xx99	27-9-2017 15:04:36	SCREEN
36463	AREA2.xx99	AREA2.xx98	27-9-2017 15:05:40	SCREEN
36463	AREA2.xx98	AREA2.xx01	27-9-2017 15:05:40	SCREEN
36463	AREA2.xx01	AREA1.xx03	27-9-2017 15:05:40	SCREEN

Table 2: Completed log example

3.3 Business Questions

Here we present business questions provided by Vanderlande. To recall from the introduction, these baggage handling systems are expensive and represent a serious investment for clients of Vanderlande. For this reason, extracting the maximal amount of value from these investments is a large point of interest. Below we articulate what the usecases for Vanderlande are, and how the research questions RQ1 and RQ2 stem from those.

Use case 0: What routes are being taken in the baggage handling system?
A critical part of understanding what is happening in the system is identifying what kinds of routes are being taken and at what frequency. Understanding the distribution of Low-level routes (the clusters to be formed by the technique) can help give insight into how the system is being used and provide a context for further findings. This idea naturally forms the question presented in RQ1.

Use case 1: What High-level routes are fast and slow?
Understanding which High-level routes are fast and slow will help engineers understand how effective their routing algorithm is and allow system analysts to diagnose how effective the existing High-level processes are. As this is dealing with fast and slow routes - this where RQ2 stems from. However, it is from the point of view of the High-level routes.

Use case 2: How do the routes per High-level step affect bag speed?
By understanding which Low-level routes per High-level step are fast and slow, it will be easy to see if it is the routing in that High-level step that is slowing bags down or something else. This again is where RQ2 comes from, however it is from the point of view of Low-level routes.

Now that we have established the potential value through usecases, a quick preliminary analysis was done to better understand the scope of the problem.

3.4 Preliminary Analysis

To understand how difficult these questions would be to answer, a small investigation had to be done. Due to the size of the data and memory restrictions, only a single day was able to be looked at a time. For a single day of data, 31 logistic steps (out of 44 possible) and 1751 locations were seen in traces. At the High-level 660 different permutations were seen for a single day. However, most bags follow only a hand full of variants. In Figure 13 we can see the distribution of the number of bags following in each variant, and can easily see why the top 10 variants contain $\frac{2}{3}$ of all bags.

Figure 13: Histogram showing distribution of number of bags per High-level variant

Now looking at a more fine-grained level, summed up there were a total of 2,463 different Low-level variants over all logistic steps. In Figure 14 we can see the distribution of the number of Low-level variants per logistic step.

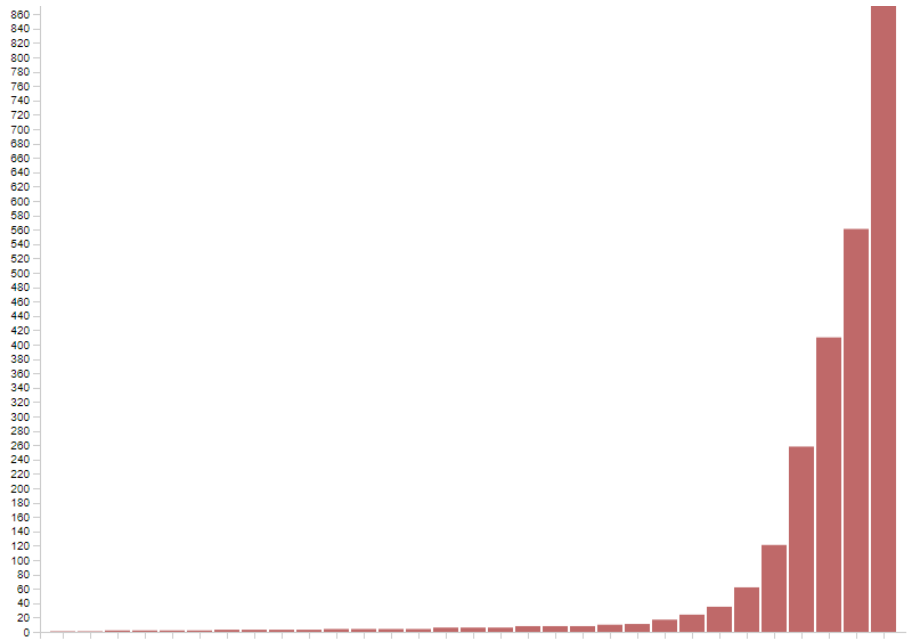

Figure 14: Histogram showing distribution of number of Low-level variants seen per logistic step

Now, taking just the top 3 logistic steps in terms of the number of variants seen on this one day, we see that theoretically, there are at least $873 * 561 * 410 = 200,798,730$ possible permutations for those 3 logistic steps in a fixed order. Since bags can loop in the system, at the logistic level we could have these logistic steps repeat, further increasing the possible variations. Due to not only

the huge number of possible variants but also the large frequency of uniquely occurring variants on both High-level and Low-level processes, it became immediately apparent that meaningful analysis could only be done when restricting the scope immensely. Even though some insights were able to be gained, clearly more had to be done in order to satisfactorily understand the system as a whole. We articulate the problem in greater detail in the next section.

3.5 Problem Description

Given the data, it is impossible to see how bags move through the system as a whole. As demonstrated in Section 3.4: there are simply too many possibilities for Low-level variants across the whole system to compare routes effectively. This makes understanding the routes a bag takes through the system an unmanageable task. Further, simple solutions such as reducing the bag trace to its High-level variant provides no desired insight. We already have seen from Figure 13 that the majority of bags follow only a handful of High-level variants. Hence although the bag traces are summarized, they provide no reason for why certain routes of bags are fast or slow. This means we need to create a summarization of routes that has a granularity in-between the High-level and Low-level variants.

Thus, we propose to solve the problem through data summarization. Once similar routes are clustered together and reduced to a suitable size for analysis, then we have the capability to answer the relevant business questions. However, the true challenge of this direction is: how do we determine what "similar" means? Now routes consist of sequences of locations, so intuitively one would expect that similar routes (in some form) share similar locations as well. Yet with over 1,500 locations in the system, almost 2,500 different kinds of Low-level variants and over 30,000 bags passing through the system every day, we do not have the computational power to compare all Low-level paths using all locations. However, if paths could be reduced to their essential locations, then comparisons could become feasible. Further, we hypothesize that if "similar" routes are effectively grouped, then fast and slow routes will also be grouped together. With these ideas in mind and data summarization becoming our main goal, we formalize this below in terms of objectives to make the direction very pragmatic.

Objective 1: Create a technique to cluster and summarize routes.
This will involve identifying what locations we can use to summarize routes and how we can use these to form clusters.

Objective 2: Create a tool to understand why routes get summarized similarly by the technique and what properties these summarized routes contain.
Once routes are clustered together we will need a way to not only understand what the clusters mean for ourselves but also to explain to Vanderlande.

Objective 3: Gain insights into the system, and answer the business use case questions described in Section 3.3.
Ultimately, the success of the whole project depends on how well the business questions can be answered and what kind of insights can be delivered.

With these objectives defined, we start immediately with a summary on the method developed as a solution in the next Chapter.

4 Method Used

In this chapter we aim to give a summary on how the method was developed. We start by defining optimal in Section 4.1. This is followed by a discussion on the design decision of supervised versus unsupervised in Section 4.2. Finally we finish with a succinct summary on the major steps in the method in Section 4.3. Since the method is quite technical and a high level detail is necessary, Section 4.3 is structured so that each sub-section refers to a later chapter that expands on the explanation of that step.

4.1 Defining Optimal, Sub-Optimal, Non-Optimal

First there is a practical ambiguity in RQ2 that needs to be made precise immediately. To recall, RQ2 is "Can we create a clustering technique that allows us to identify optimal, sub-optimal and non-optimal routes?". The ambiguity is that RQ2 uses the terms "Optimal", "Sub-optimal" and "Non-optimal". However, there is currently no generally accepted definition of what it means to be "optimal" or "non-optimal" with regards to a baggage handling system route. Thus, as a first attempt interpretation of this, we use the total time the bag was in the system (excluding time spend in storage). We compare that to a pre-existing time value known as the *SLA*. The SLA stands for System Level Agreement, which is an agreement on the expected time a bag will take to go through the system that Vanderlande makes with clients. For the baggage handling system that was analyzed, this SLA value is 15 minutes. Further, in the interest of categorizing using the SLA value, we define an margin of 30 seconds. 30 seconds was chosen arbitrarily and because it is the maximum amount of time that would be lost if rounding to the nearest minute. For example 14:30 being rounded to 15 minutes. With this SLA value define as 15 minutes and the margin being 30 seconds, we categorize bags as the following:

Optimal - less than 14:30
We say a bag has taken an optimal route when it takes less than the SLA value minus 30 seconds - or 14 minutes and 30 seconds.

Sub-Optimal - between 14:30 and 15:30
We say a bag has taken an sub-optimal route when it takes around the SLA value (within the margins) or between 14 minutes and 30 seconds and 15minutes and 30 seconds.

Non-Optimal - more than 15:30
We say a bag has taken an non-optimal route when it takes more than the SLA value plus 30seconds - or 15 minutes and 30 seconds.

Now, in the data we use alternative terminology when referring to a bag that has traveled an optimal, sub-optimal and non-optimal route. When following an optimal route, we say the bag is *below system time*. When following a sub-optimal route the bag is *at system time*. And when a non-optimal route has been taken, the bag is *above system time*. So with our terminology defined as such, we can move onto designing our method. In the next section we start at an abstract level, considering if we want a supervised or unsupervised method.

4.2 Type of Problem: Supervised vs Unsupervised

With designing our technique, there was a question of whether to use a technique that was supervised or unsupervised. Supervised learning could be done by choosing the total time a bag took in the system as the label, the locations of the route taken as features and then try to predict whether a bag would take an optimal, sub-optimal or non-optimal route based on the predicted time. And then categorize a route as such. The problem, as highlighted at the end of section 3, is that there are too many features (locations) and too many possible combinations. Thus any supervised approach would most likely overfit. Further, using just the logistic steps as features would be too coarse-grained (as discussed in Section 3.5). Therefore, some other features at the

correct granularity to capture the variation occurring in each logistic step that is affecting bag performance are necessary. For this reason, an unsupervised approach as opposed to supervised was chosen.

4.3 The Method Overview

So now that we have defined the scope of the method to an unsupervised technique, we still need to articulate what we want from the technique. We are looking for a summarization that is in-between High-level and Low-level and is done in a scalable way. Specifically, this means we want a way to summarize these Low-level variants that makes sense within a High-level process. Since there are so many locations and so many different types of Low-level variants - we need to do this in a scalable fashion.

Based on these requirements, the unsupervised technique that was developed is a 3-step technique based on PCA (Principle Component Analysis). PCA was chosen because the heart of the problem is data summarization. Already in the discussion about the technique needing to be unsupervised (in Section 4.3) we were thinking of Low-level routes in terms of locations as features of the route. However, with so many features we need to reduce them to the most critical or important ones. In other words, we want to keep the locations that will give us the most information gain. And PCA is the most well known scalable way to accomplish this. So, with our intent to use PCA set, we move onto the details how the method works to apply this 3-step technique.

4.3.1 Transforming the Data

The immediate problem is that our 3-step technique does not work on log data. So the data we have has to be transformed into a form that the 3-step technique can take as input. We give a brief summary on what that entails in the following paragraphs.

The very first thing to address is that the log is simply a collection of individual event records (bag movements). To be able to work with the traces, the events have to be stitched together to form sequences of locations per logistic step. During this, various features such as time category (over, at or under system time) and time deltas (time taken to move between locations) have to be extracted. More details on how this is done are explained in Section 5.1.

We know that the baggage handling system really is just a collection of smaller subsystems that have their own unique characteristics (introduced in Section 3.1). Because of this, we want to apply this PCA technique to the subsystems individually. This means we need all traces that belong to the same logistic step (each logistic step is a subsystem) grouped together. This is not a trivial task and more details on why are described in Section 5.2. The end result is that all traces are grouped by the same High-level step (logistic step). A visual of split traces by High-level step is shown in Figure 15 where we have 3 mock-up traces where each dot represents a location and each trace is identifiable by a color.

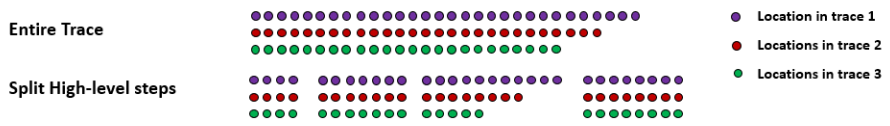

Figure 15: Splitting traces by Logistic Step

Further, now that we have sequences of locations rather than lists of events, we can make sure that the sequences of locations are free of errors. With a baggage handling system processing almost 30,000 bags a day, there are many chances for unexpected behavior to occur. And it has to be decided if this unexpected behavior is: an error in the log file, error in the reconstruction of the log file or if it is legitimate behavior but is simply unusual. This is discussed in more detail in Section 5.3. Now that we only have error-free sequences, we are ready to start thinking about how to apply PCA.

4.3.2 The 3-Step Technique: PCA for Feature Selection

The 3-step technique we summarize here is the heart of this thesis and is arguably the most important contribution made. The technique starts with sequences of locations and ends up with cluster assignments for these sequences of locations. The steps are:

1. Identifying the essential locations of a route
2. Condense routes using essential locations
3. Assigning clusters based on condensed form

We start by explaining the first step, where we look to identify the essential locations of a route. In this step, we use PCA to identify these essential locations. However, PCA is typically used to summarize high dimensional data and is done by creating a new set of dimensions. The problem is that we want to use PCA for dimensionality reduction of the original feature set - which is slightly different. This concept has been previously explored in a paper called "Feature Selection Using Principle Feature Analysis" [1] by Yijuan Lu and has been dubbed as "Principle Feature Analysis", or PFA. It is using PFA that we identify the set of our essential locations which we will refer to as the *principle set*. And for consistency, we will refer to the members of the principle set as the *principle locations*. Additional details on the intuitive understanding and the process to create the principle set are given in Sections 6.1, 6.2 and 6.3.

Now we continue by explaining the second step: condensing routes using the principle locations (essential locations). Using the principle set, we are able to summarize routes by the presence of the principle locations. We do this by projecting routes onto the principle locations. This means we keep only the principle locations and toss out all the other locations. More details on how this is done are given in Section 6.4. An example of this is shown in Figure 16 where we continue our example from Figure 15. In Figure 15 our last action was to group the traces by logistic step. In the first line of Figure 16 we highlight the principle locations in yellow. And then using projection, we are able to transform our original traces to a minimalist representation. This is what we see in the second line of Figure 16.

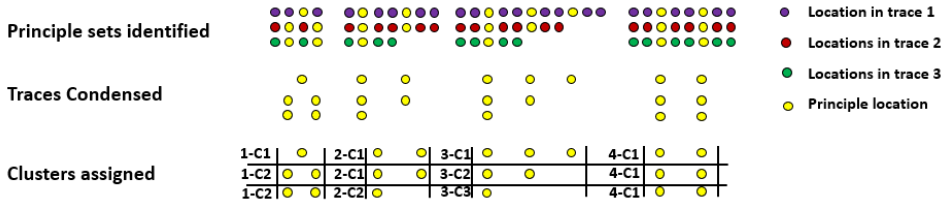

Figure 16: Projecting paths on the principle set

We now explain the last step of the 3-step technique: assigning clusters based on condensed form. Now to be clear on terminology, the term "clusters" are used to describe two very different types of clustering. In this thesis we will define two types of clusters:

1. **Route cluster**
   These are clusters whose objects are the routes themselves. These are referred to here in Section 4.3.2 and are discussed in more detail in Chapter 6.
2. **PCA clusters**
   These are clusters whose objects consist of points in the PCA space. The PCA space is defined and discussed in Section 7.1.

Now that we are clear on what we mean by "route clusters", we take the different variants of our condensed routes (the result of projecting routes on the principle locations) and these form

our route clusters from those variants. So, two routes that have the same condensed form, would belong to the same route cluster.

We see this in the last line of Figure 16 where the different clusters are given labels. These labels are in the form of "X-CX" where X is a number. The first number before the dash refers to the logistic step the cluster belongs to, and the second number after the C refers to the cluster within that logistic step. We can see that in Figure 16 that similar variants get the same label.

An example of similar variants getting the same label can be found by examining the 1st logistic step of the red and green trace. We see that they have similar locations highlighted in yellow, thus get a similar sequence after projection, and then both get labeled "1-C2". This is because they belong to the same cluster. To summarize, at this point we have demonstrated a 3-step technique that starts with sequences of locations, and uses PFA to end up with cluster assignments for these sequences of locations.

Now that we have this 3-step technique to create clusters, we turn our attention to being able to pick the best clusters. In the end, this boils down to picking the optimal size of the principle set. Details on why this is the case are explained in Section 6.5. So to do this, we need first a way to check the quality of this route clustering to know what the optimal size will be. And this problem is explained in the next section.

4.3.3 Checking the Clustering

So although our goal is to pick an optimal principle set size, it is actually a little more ambitious than that. Since we plan to apply this route clustering per logistic step (with 44 possible logistic steps), we require an "automatic optimal principle size picker technique" that can pick the optimal parameters automatically. The details for what these parameters are and the possible spaces we consider for measuring cluster quality are explained in Section 7.1.

As mentioned in Section 2.3.3, in order to determine cluster quality (since we have no known optimal clustering), a cluster index has to be calculated. Now since our clusters consist of routes, a critical part of calculating a cluster validity index is finding the appropriate distance measure for these routes. The distance measure chosen was a modified version of the Levenshtein distance. Levenshtein was chosen because the objects of our route clusters are really just sequences and Levenshtein is a well-known distance metric for sequences. However, it had to be modified to incorporate some domain knowledge to make the measure effective.

For the cluster validity index, the silhouette index was chosen. And now with both the distance measure defined and the cluster validity index chosen - this makes our "automatic optimal principle size picker technique" straight forward. We simply pick the parameters that give the highest score according to the silhouette index, using the modified Levenshtein distance. More details are explained in Section 7.2 and 7.3. So, with an automated method to optimize the parameters for PFA created, we were now ready to apply this to the whole baggage handling system.

4.3.4 Using the Clustering

Now that we are able to cluster routes, we have to apply it to the baggage handling system. Our strategy to apply it will be to do it one logistic step at a time. That is, cluster low-level routes per logistic step and form clusters per logistic step. Next, we have to figure out how to visualize the results. We need a visualization that is scalable with respect to the number of logistic steps and the number of route clusters able to visualize. This means the visualization should be capable of visualizing a large number of route clusters and a large number of logistic steps with no problem.

However, before we visualize, we need our traces in a more visualizable form. So the first step is to label our route clusters into a simpler form. And the second step is to express the traces as a sequence of clusters rather than a sequence of locations. An example of this is shown in Figure 17.

In the first line of Figure 17 you can see that the cluster labels are simplified, dropping the dash and the number before it. Although these are used internally, we do not visualize them in that format. Further, in the next line you can see that the principle locations are dropped, leaving

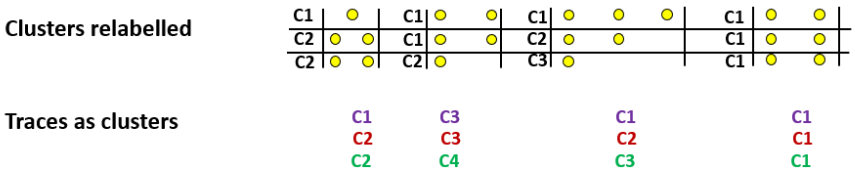

Figure 17: Traces in cluster form

just the clusters the traces pass through. We call this route cluster form. So now that the traces are in route cluster form, we can get to the visualization. A visualization in the form of a graph was chosen. This was chosen because it provided an excellent medium in which both route clusters and logistic steps could be scalable. More details on the requirements and how this was created are found in sections 8.2 and 8.3.

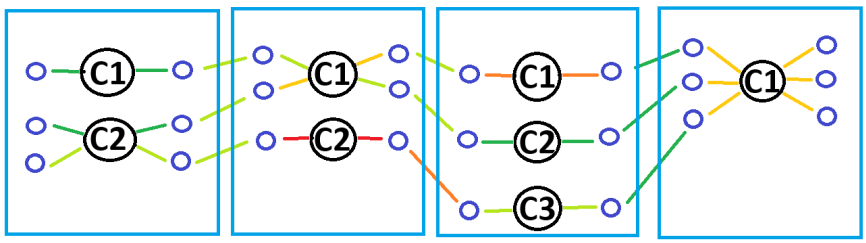

Figure 18: Traces visualized

Here in Figure 18 we see a small mockup of what the graph visualization would look like. We see that the route clusters are represented by nodes with the cluster names inside, and are connected by edges to blue nodes. The edges are colored based on the average time taken by bags that followed that route, where green is faster and red is slower. These blue nodes that are connected to each route cluster are the start and end locations of each respect trace. Further, the blue boxes seen are not part of the visualization but are there to highlight the fact that each logistic step is summarized by all the nodes in their respect blue box. And by having route clusters represented as nodes, and logistic steps represented by the collection of nodes in each blue box - we have effectively made a scalable visualization. More route clusters simply means that the visualization gets longer in the y-direction. And more logistic steps means the visualization gets longer in the x-direction. And with the start and end locations shown for each route cluster, we retain an acceptable level of detail to understand the context in which each route cluster is taken. More details of this are explained in Section 8.4.

5 Preprocessing and Execution

In this chapter we will explain the preprocessing step of the method in great detail. In Section 5.1 we start with an explanation of how we transform the event log into a format that we are able to work with. Then in Section 5.2 we go into detail of how we start grouping our newly transformed event log to a more compact format. Finally, the process of filtering incomplete bag traces is discussed in Section 5.3.

5.1 Getting Trace Features

As first introduced in Section 3.2.2 the data we have are event logs of bag movements. To recall, we present the completed log again an example shown in table 3:

Bag id	Source Location	Destination Location	Timestamp	Logistic Step
36463	AREA1.xx93	AREA1.xx92	27-9-2017 15:04:36	FIRST_SORT
36463	AREA1.xx92	AREA2.xx01	27-9-2017 15:04:52	FIRST_SORT
36463	AREA2.xx01	AREA2.xx99	27-9-2017 15:05:13	SCREEN
36463	AREA2.xx99	AREA2.xx98	27-9-2017 15:05:40	SCREEN

Table 3: Structure of the data

To give a better sense of the scale of the data we are dealing with - in our log we have 2,671,118 events (bag movements) of 28,048 different bags.

Now as table 3 shows, the data only contains events of bag movements. And the problem is that in this form it is difficult to work with. What we want is to not only transform these events into sequences of locations but also extract useful features. Useful features like the time taken between locations, the time category it belongs to and the High-level variant it belongs to. Further, we want our sequences of locations to be at the right correct granularity (between High-level steps and Low-level locations).

To accomplish the transformation from events to sequences of locations, we use grouping. And to get the location sequences at the right granularity, we choose to group by logistic step. Grouping by logistic step means that our sequences of locations will be paths through the logistic steps. And during grouping, we create following features:

- **Logistic Step**
  The Logistic Step that this row is describing.
- **Bag ID**
  The id of the bag this movement through a logistic step was about.
- **Location Sequence**
  The list of Low-level locations that this bag passed through in a particular Logistic Step.
- **Logistic Step Sequence**
  The list of all Logistic Steps that this bag passed through in the whole system.
- **Sequence ID, HL ID, Combo ID**
  These are synthetic ID's, based on the Location Sequence and Logistic Step Sequence - and are useful when looking at only part of the system in isolation. These will be explained in more detail below in Section 5.1.1.
- **Logistic Step Time Deltas**
  This uses the timestamps to calculate the amount of time it took to go from one location to another.

- **System Time Category**
  This is an indicator if the trace was above, at or below system time. This is calculated by taking the total time spent in the system, minus any time spent in any type of storage.

And below in table 4 we have an example of how this would look like as a csv file, with a simple example trace.

Logistic Step	Bag ID	Location Sequence	Logistic Step Sequence	Sequence ID	HL ID	Combo ID	Logistic Step Time Deltas	System Time Category
CHECK_IN	36463	AreaW.xx03:AreaW.xx01:AreaW.xx02:AreaW.13:AreaW.xx09:AreaW.xx02	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	0	1	0-1	00:00:45,00:00:34,00:00:15, 00:01:14	Under System Time
FIRST_SORT	36463	AreaW.xx04:AreaW.xx04:AreaW.xx96:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	0	2	0-2	00:00:25, 00:00:24,00:00:31, 00:00:14,00:00:03	Under System Time
SCREEN	36463	AreaW.xx97:AreaW.xx01:77350499:AreaW.xx98:AreaW.xx01:AreaW.xx03	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	0	3	0-3	00:00:15, 00:00:14,00:00:55, 00:00:34	Under System Time
FIRST_SORT	36463	AreaW.xx03:AreaW.xx78:AreaW.xx98:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	0	4	0-4	00:00:45, 00:00:34,00:00:15, 00:01:14.00:00:13	Under System Time
LAST _SORT	36463	AreaW.xx04:AreaW.xx04:AreaW.xx95:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	0	5	0-5	00:00:05, 00:00:04,00:00:15, 00:00:09	Under System Time
LATERAL	36463	AreaW.xx99	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	0	6	0-6	00:00:21	Under System Time
CHECK_IN	55463	AreaW.xx03:AreaW.xx01:AreaW.xx02:AreaW.13:AreaW.xx09:AreaW.xx02	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	1	1	1-1	00:00:45,00:00:34,00:00:15, 00:01:14	Under System Time
...	...	...	...	...	...	...	...	...
LATERAL	55463	AreaW.xx99	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	1	6	1-6	00:00:21	Under System Time
CHECK_IN	67463	AreaW.xx13:AreaW.xx11:AreaW.xx12:AreaW.23:AreaW.xx19:AreaW.xx12	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	2	2	2-1	00:00:45,00:00:34,00:00:15, 00:01:14	Above System Time
...	...	...	...	...	...	...	...	...
LATERAL	67463	AreaW.xx88	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	2	6	2-6	00:00:21	Above System Time
TRANSFER	96462	AreaW.xx83:AreaW.xx71:AreaW.xx12	TRANSFER:FIRST_SORT:LAST_SORT:LATERAL	3	1	3-1	00:00:12,00:00:04,00:00:15	Under System Time
...	...	...	...	...	...	...	...	...
LATERAL	96462	AreaW.xx99	TRANSFER:FIRST_SORT:LAST_SORT:LATERAL	3	4	3-4	00:00:15	Under System Time

Table 4: Example of grouped traces

From here we move onto explaining the Sequence ID, HL ID and Combo ID in more detail.

5.1.1 The Synthetic IDs

So now that we are creating these rows, grouping sequences by logistic steps, we run into the problem that the bag id no longer uniquely identifies a row. Being able to uniquely identify a row is incredibly useful, not only to visualize later on, but also when performing operations is it much simpler to refer to unique ids rather then the whole row. However, we require our identifiers to reflect how rows relate to each other. That is, since rows can contain information on the same bag - we want the identifiers to reflect this. And also we want this id to uniquely identify rows when looking at logistic steps in isolation.

A first thought of a solution to this is to extend Bag ID with a counter, one that increments every time we change a logistic step. We will call this counter the "HL ID" as it is incremented everytime we change a High-level step. And from this, we create an ID in the format of "Bag_ID - HL ID". We will call this concept of 2-part ID separated a "Combo ID". With a Combo ID, our criteria of an ID being able to indicate how rows relate to each other are met (rows containing information about same bag will have the same Bag ID number listed before the dash in the Combo ID). However, there is a problem with this when looking at logistic steps in isolation.

The issue is that bags are able to exit and re-enter a part of the system and we want to treat these re-entries differently. The idea being that when we look at a logistic step, we are looking at the behavior of routes in this step. And we consider "behavior" to be the route (sequence of locations) in a logistic step taken from the moment the bag entered that logistic step, to when it exited. So, when a bag comes around for a second time in that logistic step, it likely will take a different route - with different start and end locations. We have an example in the following Figures.

Here in Figure 19 we have an example of a logistic step with the entry locations highlighted in green, and exist locations highlighted in red. Entry/exit locations mean these are the locations where a bag may enter or exit this logistic step. Thus, all paths we see must start and end at these locations. And below, we have 2 examples paths going through this logistic step.

Here in Figure 20 we have two very different paths through this logistic step. And thus, are using the logistic step in a very different manner. This means, although maybe it is the same bag taking these routes, we want that second route to be treated as a separate distinct route.

Now normally, when looking at the whole system, it is impossible to re-enter a logistic step without going through another logistic step. For example, say in a High-level variant we have a bag

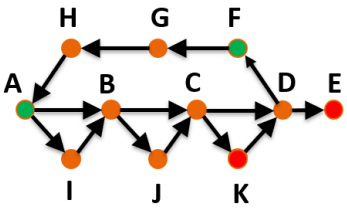

Figure 19: Example of a logistic step with highlighted entry/exit locations

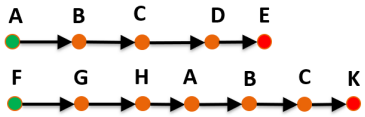

Figure 20: Example different routes taken

that goes through FIRST_SORT twice. It might enter FIRST_SORT after CHECKIN, then maybe leave to go to LAST_SORT, and then later re-enter FIRST_SORT. To re-enter FIRST_SORT the bag must go through another logistic step (in our case LAST_SORT). And because this, our previous counter (HL ID) will be able to identify this, since the counter is increased every time the High-level step changes.

However if we are looking at just FIRST_SORT in isolation, then LAST_SORT doesn't exist. Thus the HL ID counter will not be incremented because there is no change in High-level step. Thus, these two different routes through FIRST_SORT would be combined into one, since we have no way to distinguish them uniquely. Our solution to this problem is to introduce another counter.

We call this additional counter the "Sequence ID". The Sequence ID is a counter that is incremented every time there is a gap in a sequence (within in the same logistic step) or the Bag ID changes. Here, a "gap in a sequence" means that the next location that follows in the sequence is not the next expected location. Because we dealing with bag movements (that have a source and target location), identifying the expected location is relatively simple. When the target location of the previous movement doesn't match the source location of the next movement, we know that we have a gap.

And with the Sequence ID, we solve the problem of being able to distinguish these re-entries into the same logistic step as distinct routes when looking at subset of the system's logistic steps. So we redefine our Combo ID to "Sequence ID - HL ID". And now we have a working ID that uniquely identifies each row, maintains the ability to indicate which rows are related and works when looking at a logistic step in isolation. Even though in this paper we only look at the system as a whole, the Sequence ID remains, allowing the possibility to look at logistic steps in isolation.

Now that all attributes created from this grouping phase are familiar and clear, we move onto the next step that is done which is aggregation.

5.2 Aggregations

So at this point we have our bag location sequences that are nicely grouped by logistic step, and we have features for each of these groups. To be clear, we will now refer to these rows as *bag traces*, as we are no longer dealing with an event log, and these rows are now our units to describe how a bag moves through the system. Now our bag traces are not quite in the right shape to apply PFA or even to visualize. Since we plan to use this data to both PFA and visualize later, we need the aggregation to be at a fine enough level for both.

So to achieve the necessary level of granularity, we do aggregations on 4 different fields of table

4: Location Sequence, Logistic Step, Logistic Step Sequence (High-level variant) and System Time Category. And we explain the reasoning for each below.

As first mentioned in the second paragraph of Section 4.3.1 and explained in great detail in Section 8.1: we plan to apply the PFA technique to each logistic step individually. Now our traces are already split into the chunks based on on logistic step. In other words, although the first 6 rows of Table 4 all refer to the same bag, the sequences of locations (found in the "Location Sequence" column) are split based on the logistic step they belong to (found in the "Logistic Step column").

Thus, to apply PFA to a particular logistic step, we need all the location sequences going through that logistic step together. To do this, we group by location sequences. Now, as explicitly noted in Section 3.1.2 when all locations in a sequence of locations belong to the same logistic step, then this sequence can only belong to one logistic step. Since this is the case for us based on how we constructed Table 4, this means for every Logistic sequence the High-level step is already known. Thus, also grouping by High-level step is a little redundant. In other words, leaving the High-level step out of the grouping would still result in the same output. However, since we want a "Logistic Step" column in our resulting table, the easiest way to do this is to include it in the group by. An example of different bag traces being merged together can be seen in Table 4 with bags 36463 and 55463. These bags have exactly the same values for all columns and thus get combined in the result Table 5.

Now, in the context of being efficient, we also consider what kind of shape we need the data to be in for the visualization. As mentioned in the Section 8.2, we want a visualization per High-level variant. This means it is useful to have bag traces that are part of the same High-level variant together. An example of this is bag 96462 in Table 4. This bag has a different logistic step sequence than the rest of the bags in the example, and thus in Table 5 we see each bag trace getting its own row.

Further, the visualization has to be capable of uniquely identifying route clusters that are unique to bag traces that are under, at and over system time. To calculate that, we need our bag traces to be grouped by system time category. An example of this is bag 67463 in Table 4. Here, the bag traces are exactly the same as bags 36463 and 55463. However, because it has a different time category, each bag trace gets its own row in the resulting Table 5.

So, now that we are grouping by those 4 features, we now need to define how we plan to aggregate the rest of the features. Below, we explain for each feature, how its aggregated and what it's new feature name is. And for all features not mentioned, they are simply left out.

- **Bag ID → Number of Bags**
  Here we simply count. So this features gets reduced to a number indicated the number of bags present in the grouping.

- **Combo ID → Combo ID List**
  Here we form a list of Combo IDs. So, this will essential be a list of which parts of traces belong to that grouping.

- **Logistic Step Time Deltas → Logistic Step Time Deltas Averages**
  This will get reduced to an average. So, it will be a list of average time deltas for each movement between locations.

And below in table 5 we have an example of what the aggregated traces would look like.

So now our bag traces are sufficiently aggregated to not need repeated aggregation when applying PFA or visualizing. So we are ready to move onto how we apply PFA. Now, before applying PFA, the bag traces are sorted into different files based on logistic step. So no additional grouping or aggregation is being done - but the bag traces are split by logistic step. This way, we can feed each file in, one at a time into the PFA algorithm.

Logistic Step	Location Sequence	Logistic Step Sequence	System Time Category	Number of Bags	Combo ID List	Logistic Step Time Deltas Averages
CHECK_IN	AreaW.xx03:AreaW.xx01:AreaW.xx02:AreaW.13:AreaW.xx09:AreaW.xx02	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-1,1-1]	00:00:45,00:00:34,00:00:15, 00:01:14
FIRST_SORT	AreaW.xx04:AreaW.xx04:AreaW.xx96:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-2,1-2]	00:00:25, 00:00:24,00:00:31, 00:00:14,00:00:03
SCREEN	AreaW.xx97:AreaW.xx01:77350499:AreaW.xx98:AreaW.xx01:AreaW.xx036	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-3,1-2]	00:00:15, 00:00:14,00:00:55, 00:00:34
FIRST_SORT	AreaW.xx03:AreaW.xx78:AreaW.xx98:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-4,1-4]	00:00:45, 00:00:34,00:00:15, 00:01:14,00:00:13
LAST _SORT	AreaW.xx04:AreaW.xx04:AreaW.xx95:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-5,1-5]	00:00:05, 00:00:04,00:00:15, 00:00:09
LATERAL	AreaW.xx99	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-6,1-6]	00:00:21
CHECK_IN	AreaW.xx13:AreaW.xx11:AreaW.xx12:AreaW.23:AreaW.xx19:AreaW.xx12	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Above System Time	1	[2-1]	00:00:45,00:00:34,00:00:15, 00:01:14
FIRST_SORT	AreaW.xx04:AreaW.xx04:AreaW.xx96:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Above System Time	1	[2-2]	00:00:25, 00:00:24,00:00:31, 00:00:14,00:00:03
SCREEN	AreaW.xx97:AreaW.xx01:77350499:AreaW.xx98:AreaW.xx01:AreaW.xx036	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Above System Time	1	[2-3]	00:00:15, 00:00:14,00:00:55, 00:00:34
FIRST_SORT	AreaW.xx03:AreaW.xx78:AreaW.xx98:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Above System Time	1	[2-4]	00:00:45, 00:00:34,00:00:15, 00:01:14,00:00:13
LAST _SORT	AreaW.xx04:AreaW.xx04:AreaW.xx95:AreaW.xx97:AreaW.xx96:AreaW.xx95	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Above System Time	1	[2-5]	00:00:05, 00:00:04,00:00:15, 00:00:09
LATERAL	AreaW.xx88	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Above System Time	1	[2-6]	00:00:21
TRANSFER	AreaW.xx83:AreaW.xx71:AreaW.xx12	TRANSFER:FIRST SORT:LAST SORT:LATERAL	Under System Time	1	[3-1]	00:00:21, 00:00:24,00:00:25
FIRST_SORT	AreaW.xx03:AreaW.xx78:AreaW.xx98:AreaW.xx97:AreaW.xx96:AreaW.xx95	TRANSFER:FIRST SORT:LAST SORT:LATERAL	Under System Time	1	[3-2]	00:00:45, 00:00:34,00:00:15, 00:01:14,00:00:13
LAST _SORT	AreaW.xx04:AreaW.xx04:AreaW.xx95:AreaW.xx97:AreaW.xx96:AreaW.xx95	TRANSFER:FIRST SORT:LAST SORT:LATERAL	Under System Time	1	[3-3]	00:00:05, 00:00:04,00:00:15, 00:00:09
LATERAL	AreaW.xx99	TRANSFER:FIRST SORT:LAST SORT:LATERAL	Under System Time	4	[3-4]	00:00:21

Table 5: Example of aggregated traces

5.3 Filtering Traces

However, before we move onto the PFA algorithm there is some cleaning that has to be done. After having constructed these bag traces, and grouped them together based on High-level step (logistic step) - it became clear that some of these traces were incomplete. That is, many movements ended in the middle of logistic steps - where no bag would be expected to end. Clearly this is a problem since the traces no longer represent complete behavior. And so we need a way to remove these incomplete traces.

Now although there could be different reasons for this, the largest factor was that the time range of the log files started from midnight of the previous day and ended at midnight of the current day. And when the "day" ended, bags where still in the system - thus, leaving traces incomplete. So we aimed to remove all incomplete traces, and be left with a log that only contained complete bag movements.

To do this, a whole list of start and end logistic steps locations within each logistic step had to be created along with a list of logistic steps we expected traces to start and end at. We called such logistic steps (logistic steps where bags would appear for the first or last time in the whole system) "starting logistic step" and "ending logistic step" respectively.

So, not only did we have to know what logistic steps we expected bags to start and end, but within every logistic step, we also wanted to know where bags could start and end. Then, all traces had to be iterated through, and checked against these two lists. However, this also led to discovering several quirks of system and log files.

An example of this was that we found that within a starting logistic step, bags where starting in several different locations within these starting logistic step. In other words, there was a whole web of nodes that a bag could appear for the first time in the system. Now to illustrate why this is surprising, we use a small example in Figure 21.

In Figure 21 we have a small directed graph, with the start nodes highlighted in green. Now, in a direct graph, one would expect all objects that move through the graph to first be seen in the green nodes. This is because it is directed, and these are the only nodes with no incoming edges. However, what we were seeing in the data was that at any node in this graph, objects could appear for the first time. And this was unexpected.

Further, we observed the same behavior at the destination areas. Even though the destination area was in the shape of a line (3 nodes in a line) - the tracking of the bag could end at any of those 3. In the end, for the start and end areas, we decided that any bag starting anywhere in a start area or ending anywhere in an end area we could consider legitimate. Our reasoning was that the behavior we were most interested in observing was how the bag moved through the system,

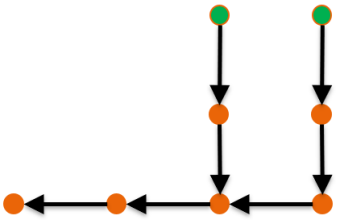

Figure 21: Example of a starting logistic step

rather than where it first appeared or disappeared.

So for the day that we processed, there were a total of 28,048 bags and ended up removing 2909 (little over 10%) via the process described above. Thus leaving, 25,139 bags remaining described by 284,548 rows of data, in the format of the table 5.

6 Using PFA

In this chapter we explain the 3-step technique of where we take a set of routes as input and end up with them clustered as output. We start this explanation by giving an intuitive explanation of what the "Principle Set" is in Section 6.1. We then move on to how we apply PCA to bag traces in Section 6.2. Now with the PCA output in hand, Section 6.3 explains how the principle set is found. Section 6.4 elaborates on how clusters are formed using the principle set. And then the chapter is wrapped in Section 6.5 where the 3-technique is summarized.

6.1 Defining the Principle Set

At first, applying PCA to a graph might sound like a strange idea. So let us first intuitively understand the larger idea and why applying something like PCA to a graph would give any reasonable results at all. Doing this now will make the rest of the explanation much easier to follow. And in doing this we aim to formally introduce the notion of the "principle set".

First, consider a simple example such as the graph in Figure 22.

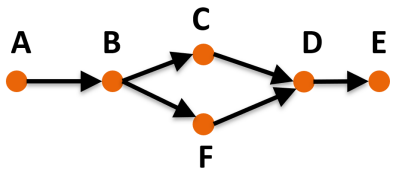

Figure 22: Simple graph

We can consider all the locations in this graph as features. This means a path in our graph is actually just a certain sequence of our set of features. Here only 2 paths are possible. Namely: A,B,C,D,E and A,B,F,D,E. Now we can encode these paths in a matrix, where the columns are locations on our graph, the rows are the different paths, and the values represent the frequency of that location appearing in the path. Encoding these will result in a matrix containing rows [1,1,1,1,1,0] and [1,1,0,1,1,1] respectively. This is shown in Figure 23.

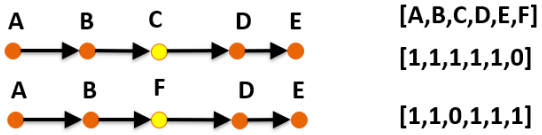

Figure 23: Simple example encoding of paths

And if we simply count, looking at the numbers, we see that the frequency in which location C and F are used changes the most - compared to the rest of the locations. In other words, the variance will be greater for these 2 points than for the others. And this is something that we not only want PCA to detect, but expect PCA to detect.

Now the reason this becomes useful to identify, is that these 2 paths can actually be identified solely by the presence of those 2 locations. If we know C is present, we know which path was taken. This means the other locations are unnecessary.

For future reference, we will call these locations we pick (locations like C and F) our principle set. Principle sets are the heart of the technique being described here. In fact, the different observed permutations of items within the principle set, end up become the route clusters as we will see later.

And so here, we have demonstrated intuition behind why applying a technique like PFA would make sense, how it can be useful. Now we will look at exactly how PFA identifies the principle set.

6.2 Getting the PCA Matrix

So now that we understand what the principle set is, and how it can be extracted from the PCA matrix, we have to first build the PCA matrix. The problem is that we have to go from a collection of bag traces, into a matrix.

We started with encoding all the bag traces as a matrix (as demonstrated in Figure 24). The first step to encoding was to generate a list of unique locations seen from all bag traces. This "header list" would form the header of our matrix.

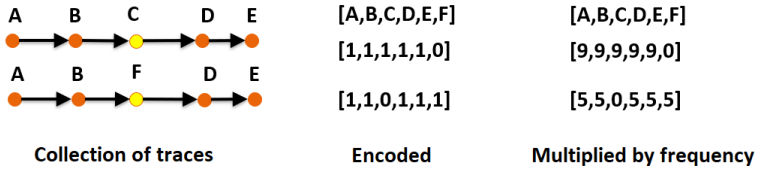

Figure 24: Simple example encoding of paths

Next, we converted all of our Location Sequences into an "encoded list" of the same length as the "header list", putting a 1 when the location occurs and a 0 otherwise. We then multiplied every value in the "encoded list" by the number of bags we have seen. We then combined the "header list" and all "encoded lists" to form a matrix, to which we applied a Python PCA library (the one used was from SciLearn), and stored the result.

The resulting matrix is like the one shown below in Figure 25.

PCA Output

	Location1	*Location2*	*Location3*	*Location4*	*Location5*	*Location6*	...	*Location n*
PC 1	.3	0	.1	.1	.1	.6	...	.3
PC 2	0	0.3	.1	0	.2	.1	...	.2
PC 3	0.1	.11	.1	.1	1	0	...	.1
...	...	...	...	...	...	...	...	...
PC m	0.2	.51	.3	.2	1	0	...	.2

Figure 25: PCA Output Example

Here you can see that the columns of the resulting matrix are the same as the input matrix. However, now the rows which were bag traces before, are now replaced with the Principle Components.

6.3 Finding the Principle Set

As first introduced in Chapter 4 we want to use PCA for feature selection. That is, using the output of PCA (a matrix, such as the one shown in Figure 25), extract a suitable principle set of a size that we choose. However, the problem is that there are several interpretations of the principle component matrix that would lead to different strategies for this. What we aim to do here is introduce all of these different strategies, and explain their pros and cons. And in the end

we want to pick the best strategy that allows us the most control over the size of the resulting principle set and contains the highest quality principle set.

First, we start with the output of PCA which is a matrix. The PCA matrix values represent each location's contribution to a principle component. In other words, a row together describes the direction of each principle component, in terms of the original dimensions (the original features). And since the principle components are specifically chosen to indicate the direction of the most amount of variance in the data, these values, effectively indicate how much each location contributes to that dimension of variance.

Now, from our PCA output matrix, we keep only the top 3 Principle Components. The main reason being that 3 is small enough to visualize in order to better understand the output, which was our main concern. This means our PCA matrix actually looks more like what shown in Figure 26.

PCA Output

	Location1	*Location2*	*Location3*	*Location4*	*Location5*	*Location6*	...	*Location n*
PC 1	.3	0	.1	.1	.1	.6	...	.3
PC 2	0	0.3	.1	0	.2	.1	...	.2
PC 3	0.1	.11	.1	.1	1	0	...	.1

Figure 26: PCA Output Example

Now there are several interpretations of the principle component matrix that can be used to extract a principle set of locations. The first is using the principle components themselves as representations of contribution to variance, and using thresholds to identify features that contribute the most. The second is using the principle components as point in Euclidean space, and using conventional clustering techniques to identify features that uniquely contribute variance. Each of these 2 interpretations and respective techniques are described in the following 2 sections.

6.3.1 PFA Threshold-Based Techniques

So if we interpret the principle component values as the amount variance contributions for each location, we have to come up with some techniques to use those values to identify the most essential locations. These ideas need to somehow link variance to essentialness or importance of a location in a route. And they have to allow us to choose the size of the resulting principle set.

Now since the principles components are sorted in descending order of the amount of variance they explain, one thought is to take locations that have a highest absolute value for one of the principle components and use those locations as our principle set. In other words, pick locations that are above a certain threshold of significance. And several possible variants of picking thresholds come to mind, as described below.

- **Idea 1 (T1)**
  Given a threshold x, keep all locations that have at least x for all 3 principle components.
- **Idea 2 (T2)**
  Given a threshold x, keep all locations that have at least one value for any principle component that is greater than x.
- **Idea 3 (T3)**
  Given a threshold x, keep all locations that have at least x for the first principle component (since the first principle component explains the most variance).
- **Idea 4 (T4)**
  Given a threshold x, create a weighted score for each location.

For each location l:
For each $PC\_num \in \{1, 2, 3\}$
Weighted score $= \sum_{PC\_num}$ (amount of variance explained by PC PC_num * PC PC_num value for location l).
Then keep all locations that have *Weighted score* greater than x.

Now the benefit to these ideas is that they are easy to implement, and they are intuitive to understand. However they suffer from 2 major drawbacks. The first is that it is difficult to control the size of the principle set. This is because the number of locations that meet a threshold jumps erratically. And the reason for this brings us to our second drawback: it is possible for several locations to contain the similar information content.

So, for example there might be 2 locations that have exactly the same values to the respective 3 principle components. And both would meet whatever the threshold was set to, since they contain the same, if not similar, values. This means locations in our principle set would contain redundant information, and reduce it's effectiveness.

Lu's paper [1] points this out, and thus provides a second interpretation that avoids these drawbacks.

6.3.2 PFA Clustering-Based Techniques

Now we move onto Lu's interpretation of the PCA matrix. This is where we look at each of the features (columns of the PCA matrix) as a point in space where the Principle Components (rows of the PCA matrix) form the axis of our new space. We refer to this space as the *PCA space* and it is discussed in more detail in Section 7.1. As mentioned in Lu's paper [1]: we can cluster locations together in this PCA space to form PCA clusters (first mentioned in Section 4.3.2). We then pick the location closest to the centroid of the cluster to represent the cluster. And then we finally construct our principle set from these locations closest to the centroid. However, we have to figure out what kind of clustering we are interested in and how this affects the effectiveness and understanding of the resulting principle set.

We consider just two different kinds of clustering: k-means and agglomerative clustering. We only consider these two due to limited time and computing resources.

- **K-means (C1)**
  K-means is the technique mentioned in Lu's paper[1]. It is the one they recommend and requires the number of clusters to form as input. However, K-means is not deterministic and requires repetitive runs to converge- where the speed of convergence depends on the initial seeds chosen. It is also difficult to see what locations k-means deems to be the most similar or different from each other. To try little more deterministic clustering approach, and also try to understand what locations are similar, we turned to agglomerative clustering.

- **Agglomerative (C2)**
  Now agglomerative clustering requires the distance at which to stop merging clusters together as input (although it can also be given a number of clusters to find as input). Other than being relatively deterministic, it is also easier to see what locations agglomerative considers to be similar. Because agglomerative works in a hierarchical fashion, the order in which it decides to merge clusters can be visualized in a dendrogram. And thus, can give a sort of hierarchical ranking as to which locations it deems to be similar. And this can give some insight into what locations are similar to each other, and how pca is capturing (or not capturing) the similarity we are expecting (from domain knowledge).

6.3.3 Evaluating the 6 techniques

Now all 6 of these techniques were implemented: the 4 different threshold ideas (T1-T4) along with the 2 clustering ideas (C1-C2). To be clear, we will refer to these two different categories

of techniques as the "Threshold-Based Techniques" and "Cluster-Based Techniques". Next we needed to determine which of these techniques we would use.

We ended up determining the best through a sort of trial-and-error approach. For a particular logistic step, all 6 techniques were run and where all told to find the same size of principle set. Then the quality of the resulting principle sets were judged using domain knowledge. The result was Cluster-Based Techniques preform better than Threshold-Based Techniques and we explain why below.

Again, Lu's paper [1] observation of thresholds picking locations that contained similar information was evident in the findings. Many times the Threshold-Based Techniques would pick several locations that would form a route cluster, when only 1 or two would be necessary. For example, if we consider Figure 27, we can see an illustration of a type of bad principle set typically found by the 4 threshold idea techniques.

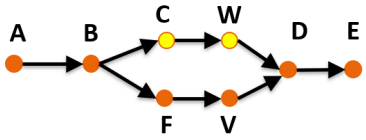

Figure 27: Example of bad principle set

Here in Figure 27 we see a slightly modified version of the graph first shown in Figure 22. Because locations C and W meet whatever threshold is set, and both contain exactly the same contributions of variance (since it is impossible to go through W without going through C), they will always be picked together.

So suppose that a threshold type technique is told to find a principle set of size 2, and C has a little more variance than F. The exact PCA values do not matter, just that there is some threshold set that C meets and F doesn't. The resulting principle set would be the highlighted yellow nodes shown in Figure 27. However, if exactly the same parameters were given to one of the 2 Cluster-Based Techniques, the Cluster-Based Techniques would choose a principle set similar to the example shown in Figure 28.

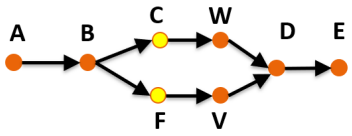

Figure 28: Example of good principle set

Here, since C and W contain exactly the same information, they would be considered similar and always be in the same cluster in the PCA space. The same with F and V. And since only 1 location that is closest to the centroid is chosen to represent the clustering in the PCA space - C will be chosen but not W (because of its proximity to C) and either F or V will be chosen instead (because of its larger distance to C and W). Thus, a higher quality principle set would be found.

During analysis both techniques from the Cluster-Based techniques (C1 and C2) gave similar and almost indistinguishable results. Since neither could be decisively declared better than the other, we chose C1 simply because it is the technique referred to in Lu's paper [1]. Now that we know which methods result in the highest quality principle set, we can move onto how we plan to use the principle set.

6.4 Using the Principle Set

Now that we have the principle set, we can go ahead and use them. From the principle set, we want to form route clusters.

If we recall from chapter 4, our last observation was that our paths can be effectively summarized by just the locations in the principle set. And this implies that the other locations in the path are no longer useful. Thus, to get rid of them, we can use the operation of projection. Projection is an operation in process mining that acts like a filter, leaving only items we are projecting on. And so projecting locations C and F respectively on the paths, would lead to a condensed form of the paths as we can see in Figure 29.

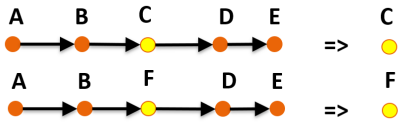

Figure 29: Projecting paths on C and F

The resulting variants of these newly condensed paths form our route clusters. So for our example in Figure 29, we have just 2 route clusters: namely Cluster "C" and cluster "F". And through this manner, we can assign every type of route we see to a route cluster.

So now that we know what the principle set is, we have the means to assign each route to a route cluster. We start by projecting on the "Location Sequence" column of table 5. We do this for each technique, and create a new column for the result of each projection. This means we add 2 new columns to our table 5, that contains the projected bag trace (the principle set variants). And essentially, the route clusters that each bag trace belongs to, depending on the technique used.

Below in table 6 we have an example of what we are talking about. In implementation, table 6 would be joined with table 5 in one long table, but to make things fit, we only show the "Location Sequence" column from table 5, and the new columns being generated.

Location Sequence	Method C1	Method C2
AreaW.xx03:AreaW.xx01:AreaW.xx02:AreaW.xx13:AreaW.xx09:AreaW.xx02	AreaW.xx09	AreaW.xx13
AreaW.xx04:AreaW.xx97:AreaW.xx96:AreaW.xx97:AreaW.xx96:AreaW.xx95		AreaW.xx97
AreaW.xx97 : AreaW.xx01 : AreaW.xx99 : AreaW.xx98 : AreaW.xx01 : AreaW.xx03	AreaW.xx01	AreaW.xx03
AreaW.xx07:AreaW.xx06:AreaW.xx02:AreaW.xx13:AreaW.xx09:AreaW.xx02	AreaW.xx09	AreaW.xx13

Table 6: Example of clustered bag traces

And just be absolutely clear and explicit - the values under the columns Method C1 and Method C2 are the route clusters that the bag traces are assigned to. So, if we look at the first row, we have sequence "AreaW.xx03:AreaW.xx01:AreaW.xx02:AreaW.xx13:AreaW.xx09:AreaW.xx02". This sequence is assigned to route cluster "AreaW.xx09" in method C1, route cluster "AreaW.xx01" in method C2.

6.5 The 3-Step Technique Summary

To summarize, at this point we have demonstrated a 3-step technique that start with sequences of locations, and ends with cluster assignments for these sequences of locations. The steps are:

1. Identifying the principle locations using PFA
2. Condensing routes using principle locations
3. Assigning clusters based on condensed form

However, we still need a way to check the quality of this route clustering. Now that we have a 3-step technique to cluster routes, we have to not only check that our clustering is effective, but try to get the best clustering possible. Since the only step the 3-step method that accepts a parameter is the first step, the problem is reduce figuring out the optimal parameters for this step. The first step is where PFA is used to identify the principle set, and accepts the size of the desired principle set as a parameter. Thus we are trying to optimize the size of the principle set in order to have the highest quality of route clusters as possible.

And this problem is explained in the next section.

7 Cluster Evaluation

In this chapter we explain how we optimize the 3-step technique to pick the best clusters possible with the one parameter available: the size of the principle set. Section 7.1 kicks off the explanation with a discussion on the different spaces in which we can evaluate the clusters. Once we know where we can evaluate, Section 7.2 dives into what measure we can use to evaluate. And then everything is tied together in Section 7.3 where how we use these measures on the different spaces is explained.

7.1 The Spaces

If you recall from chapter 4, we want to design an "automatic optimal principle size picker technique". However, what the summary glossed over is that we have to decide in what space to evaluate them in. Because of the way we constructed our route clusters (the clusters containing routes) we have 2 possibilities. And the problem is to understand the difference between the two, and how we can apply cluster validation scores to each to understand which leads to a more optimal principle set size.

Now there were 2 different spaces that we were considered when trying to determine how to automate the choice of the optimal size of the principle set. These 2 spaces are: the PCA space and the route space.

The PCA space is formed from the output of the PCA matrix. That is, it is looking at all locations as points in Euclidean space, where the dimensions of this space are 3 Principle Components that each location has been projected onto. In other words, the objects in our clusters are the locations themselves.

The route space is where the routes themselves are the objects in our clusters. Since routes are not points in Euclidean space, but rather sequences of locations, this is a different space.

The difference between these two spaces is that the PCA space is looking at the Principle set directly, to determine the optimal size of the principle set. This approach is reasonable since the question being asked is about the principle set. So looking at the properties of the principle set itself is a straight-forward, practical idea. Now the route space is looking at the quality of the resulting route clusters to determine if the size of the principle set is optimal. This approach also makes sense since the "optimal" part in finding the "optimal principle set size" is determined by the quality of the route clusters.

Now since the objects in the 1st space (PCA space) are simply points in Euclidean space, the Euclidean distance was an obvious choice for a distance measure. However, since the objects in the 2nd space (the route space) are routes themselves, a new distance measure had to be chosen.

7.1.1 Route Space Distance Measure

Since our clusters consist of routes, which are sequences of locations and not points in Euclidean space, a new distance measure had to be chosen.

Now routes are actually just sequences that can differ in length. And one of the most common ways to measure sequence distance is the Levenshtein distance measure. The Levenshtein distance is a measure that indicates the number of edits necessary between two sequences to make them the same. The formula used to calculate this is as follows:

Let a and b be strings, with length $|a|$ and $|b|$ respectively.
The Levenshtein distance between strings a and b is given by $\text{lev}_{a,b}(|a|, |b|)$ where

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j) + 1 \\ \text{lev}_{a,b}(i,j-1) + 1 \\ \text{lev}_{a,b}(i-1,j-) + 1_{a_i \neq b_j} \end{cases} & \text{otherwise} \end{cases} \quad (1)$$

Thus, the more edits necessary, the further apart the two sequences are.

However, based on domain knowledge, the number of locations in common was not the only indicator of how different two routes could be. The frequency of usage of a route (the number of bags passing through routes if you will) was also an indicator of how similar or dissimilar routes could be. This was because a closer inspection of routes revealed that there were routes that were more commonly used than other routes. In other words, there were "main" routes that bags would commonly take and "alternative" routes that bags would take less frequently. To detect this, we took the number of bags passing through a route as a measure. So even though two routes might contain similar locations, if they were used at a different frequency, we wanted them to be considered more distant.

Thus, an additional distance measure was created, called the digit distance. The digit distance is simply the difference in the number of digits between the number of bags passing through each route.

Let $x, y \in N$ indicate the number of bags passing through 2 arbitrary routes.
Then the digit distance is calculated as follows:

$$\text{digit\_dist}(x, y) = |\lfloor \log_{10}(x) \rfloor - \lfloor \log_{10}(y) \rfloor| \tag{2}$$

Thus, the final distance measure combines both these distance calculations together. The final distance measure used is as follows:

Given two routes a, b, where x, y are the number of bags passing through routes a, b respectively, the distance between these routes are calculated

$$\text{final\_dist}(a, b, x, y) = \text{digit\_dist}(x, y) + \text{lev}_{a,b}(|a|, |b|) \tag{3}$$

7.1.2 Route Space Modification

Now we have a suitable distance measure for the route space, but immediately run into a practical problem. This problem is that the route space is too large to apply a modified Levenshtein distance measure to all routes. This is actually not a new observation. One of the very reasons the 3-step technique had to be invented to cluster routes was that comparing all routes to each other was simply not possible. Thus, we needed a different procedure to be able to apply our distance measure to routes. This procedure had to not only be scalable but also reasonably represent the routes and their key characteristics.

To do this, we used domain knowledge to compress routes into a shorter form where it was feasible to apply our modified Levenshtein distance measure. Our compression relies on the fact that the "area" part of all locations (present in all Vanderlande systems) is an indicator of what locations belong together. What we do is transform our sequences of locations into sequences of areas. And the transformation is done using the following rules:

- After changing areas, keep all unique area edges
- For each loop, add an addition area edge

In Figure 30 we have an example of a location sequence that contains different areas, and how such a location sequence would get compressed.

In Figure 30 we can see that all unique edges are kept. In this compression, we can see two types of unique edges kept: transitions to different areas and an edge within the same area. Since there are no loops in this sequence, the 2nd bullet point does not need to be applied. However, for an example of applying this, we turn to the example shown in Figure 31.

Here in Figure 31 we see an example of a location sequence where one loop occurs, which means locations get repeated twice. We see that the first edge from Area5 to Area5 is in place because

Area1.xx97 : Area2.xx01 : Area2.xx99 : Area2.xx98 : Area2.xx01 : Area1.xx03

Area1 : Area2 : Area2 : Area1

Figure 30: Compressing a regular bag trace

Area5.xx09 : ... : Area5.xx50 : Area5.xx09 : ... : Area5.xx50

Area5 : Area5 : Area5

Figure 31: Compressing a looping bag trace

of the first bullet point. And since this is the only unique edge and there are no transitions to different areas, this is all the first bullet contributes. Then because there is an additional loop occurring (repeat of locations) we add a single additional edge. And this is how we end up with 2 edges in our compressed area sequence.

Area5.xx09 : ... : Area5 .xx50 :Area5.xx09 : ... : Area5.xx50 : Area5.xx09 : ... :Area5.xx50

Area5 : Area5 : Area5 : Area5

Figure 32: Compressing a double looping bag trace

In Figure 32 we have another looping example, where an additional loop is taken. And we can see in the compressed area sequence that due to this, an additional edge is added.

Now that we have an applicable distance measure for both spaces, we have the necessary tools available to apply cluster validation scoring to each.

7.2 Cluster Validation Scoring

To optimize any clustering the goal is to minimize the intra-cluster distance (distance within a cluster) and maximize the inter-cluster distance (distance between clusters). However, there are not only different ways to calculate the inter and intra cluster distances, but also different ways to use them to determine cluster quality. The problem here was to identify some of these different cluster validation scoring and understand them to generate some candidates for our "automatic optimal principle size picker technique".

An extensive study on different cluster validity techniques where done in Olatz's paper [4] covering over 30 different cluster validity indices. However, due to resource and time limitations, only 2 were considered. Namely Dunns Index and the Silhouette Index.

7.2.1 Dunns Index

Dunn's Index was chosen because it was simply, intuitive and easy to implement. Below we see the equation for Dunns Index.

We assume a distance function (for example the final_dist(a, b, x, y) function presented at the end of Section 7.1.1). And we denote it as such:

$$\mathrm{d}_e(x_a, x_b) = \text{Distance between points } x_a \text{ and } x_b \tag{4}$$

We define the function for calculating the intra-cluster distance as follows:

$$\Delta(c_k) = \max_{x_a, x_b \in c_k} \{\mathrm{d}_e(x_a, x_b)\} \tag{5}$$

We define the function for calculating the inter-cluster as follows:

$$\delta(c_k, c_i) = \min_{x_a \in c_k} \min_{x_b \in c_i} \{\mathrm{d}_e(x_a, x_b)\} \tag{6}$$

And then finally, Dunns Index is as follows:

$$\mathrm{D}(C) = \frac{\min_{c_k \in C}\{\min_{c_i \in C \setminus c_k}\{\delta(c_k, c_i)\}\}}{\max_{c_k \in C}\{\Delta(c_k)\}} \tag{7}$$

And put simply, Dunns Index is the ratio of nearest neighbour distance (the distance of the two closest clusters) to the cluster with the largest diameter (with the largest distance between cluster members). With Dunn's Index, a higher number is better. And intuitively, we can think of the nearest neighbour distance as the way to estimate inter-cluster distance and the cluster diameter as the intra-cluster distance. We already know that we want intra-cluster distance to be as low as possible, and inter-cluster as high as possible. In Dunn's index, this translates to be a high numerator and a low denominator. And so, the higher the numerator and the lower the denominator the larger the ratio will be. Because of this, a higher number is an indicator of better quality.

However, when applied, Dunn's index was not found to satisfactory, and so an additional validation index was tried.

7.2.2 Silhouette Index

The next one tried is known as the "Silhouette Index" and it approaches the inter/intra-cluster paradigm a little differently. Below we have the formal equation for the Silhouette Index:

Here we define an equation that serves as intra-cluster distance, but for a particular point rather than a whole cluster. And is as follows:

$$\mathrm{a}(x_i, c_k) = \frac{1}{c_k} \sum_{x_j \in c_k} \mathrm{d}_e(x_i, x_j) \tag{8}$$

Here we define an equation that serves as inter-cluster distance but for a particular point rather than a whole cluster. And it is as follows:

$$\mathrm{b}(x_i, c_k) = \min_{c_i \in C \setminus c_k} \left\{ \frac{1}{c_i} \sum_{x_j \in C_i} \mathrm{d}_e(x_i, x_j) \right\} \tag{9}$$

The finally, we define the Sillhoute Index as such:

$$\mathrm{Sil}(C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{\mathrm{b}(x_i, c_k) - \mathrm{a}(x_i, c_k)}{\max\{\mathrm{a}(x_i, c_k)\mathrm{b}(x_i, c_k)\}} \tag{10}$$

So while Dunn's index was using the extreme members in a cluster to represent a cluster, and then the extreme clusters to estimate the inter/intra-cluster; the Silhouette Index actually looks at every point individually. That is every single point gets a score based on it's inter/intra-cluster distance ratio. Then the score for the cluster is calculated by taking the average of all points in the cluster. Finally, the overall score of validity is an average of all clusters. And like Dunn's Index, a higher number if an indicator of higher quality.

Now that we have suitable candidates, we can move on to discover which performs the most accurate.

7.3 Using Scoring

So the problem here is that now that we have 2 different spaces and 2 different cluster validation indices. Both seem reasonable and have the potential to work. So we require a method to discover which combination results in the most optimal size of the principle set for as many logistic steps in the system as possible. Further, this method will have to be simple enough to be done manually, because the only way to verify optimal principle set size will have to be done manually.

So given these requirements, two different logistic steps that represented the rest of the system where chosen. The logistic steps chosen were screen and first sort. Their optimal principle set size was calculated manually and verified using domain knowledge. For screen, this ended up being a principle set size between 15 and 18 locations. And for First sort, this ended up being a principle set size of 2. Then each cluster validation index applied to each space was tried on each of these two logistic steps. These values are shown below.

Number Locations	Silhouette	Dunns Min	Dunns Max
2	0.0011	0.0	0.8571
3	-0.0795	0.0	0.8571
5	-0.0951	0.0	0.5
8	-0.1574	0.0	0.4285
11	-0.1889	0.0	0.75
15	0.1291	0.0	0.0
16	0.1291	0.0	0.0
18	0.2640	0.0	0.0

Table 7: Screen Route Space scores

Number Locations	Silhouette	Dunns Min	Dunns Max
2	0.3333	0.0	0.6666
3	0.0307	0.0	0.0
5	-0.0634	0.0	0.0
8	-0.0476	0.0	0.0
11	-0.0428	0.0	0.0
15	-0.0151	0.0	0.0
16	-0.0047	0.0	0.0
18	0.0	0.0	0.0

Table 8: First sort Route Space scores

Here in Tables 7 and 8 we can see the scoring values for the Route space. The reason we have 2 Dunn's Index columns, is that formula provided by the literature was performing horribly. The literature specifies that the inter-cluster distance should be calculated from the distance of the 2 points that are closest to each other. However, as seen in the "Dunns Min" column, the values are just 0. So several other versions of Dunn's Index were tried and the one that came the closest was using the furthest points to calculate the inter-cluster distance. And these values can be found under the "Dunns Max" column.

Number Locations	Silhouette	Dunns Min	Dunns Max
2	0.4507	0.0	0.0
3	0.5845	0.0	0.0
5	0.7485	0.0	0.0
8	0.7359	0.0	0.0
11	0.7577	0.0	0.0
15	0.7968	0.0	0.0
16	0.7777	0.0	0.0
18	0.7835	0.0	0.0

Table 9: First sort PCA Space scores

Number Locations	Silhouette	Dunns Min	Dunns Max
2	0.6795	0.0	0.0
3	0.7998	0.0	0.0
5	0.8424	0.0	0.0
8	0.9326	0.0	0.0
11	0.9415	0.0	0.0
15	0.9203	0.0	0.0
16	0.8852	0.0	0.0
18	0.8081	0.0	0.0

Table 10: Screen PCA Space scores

Looking at both Tables 7 and 8, we can start to interpret the score. To recall, a higher number is better for both Dunn's Index and the Silhouette Coefficient. So the highest number in each column is the size of the principle set that each validation index thinks is most optimal. We can see both the Dunn's Index fail to pick the optimal principle set size for both logistic steps. Dunn's Min simply has 0 for everything. And although Dunn's Max does pick correctly for First sort, it fails in Screen where it thinks 2 or 3 is the best. And we know we want something in between 15 and 18. However Silhouette gets everything correct, picking 18 for Screen, and 2 for First sort.

Now when we observe the scores in Tables 9 and 10 we immediately see that they are worse. Dunn's Index simply fails in all accounts. And Silhouette picks 15 for First sort and 11 for Screen, also missing the mark. Thus between the PCA space and the Route space and Dunn's Index and the Silhouette Index, we found that Silhouette Index on the Route space performed the most accurately.

Now that we found the more accurate combination, we can apply our "automatic optimal principle size picker as described in Chapter 4. To recall, we simply try a range of sizes for the principle set, and we keep the size that has the highest Silhouette Index applied on the Route space. And now that we can automatically determine what the optimal size of the principle set is, we are ready to apply the clustering technique to the whole system.

8 The Visualization

In this chapter we plan to introduce the visualization used to display the clustering. We start in Section 8.1 with an overview of how the method applies the 3-step technique to to the whole baggage handling system. From there, Section 8.2 moves onto defining what we require from our visualization, and how our visualization meets those requirements. However the data has to be further formatted before we can put our visualization into action- covered in great detail in Section 8.3. Finally, a little sample of the full visualization is shown in Figure 8.4.

8.1 Applying the Technique to a BHS

So for the visualization to make sense, it is important that we have a clear overview on how the technique is applied to the whole Baggage Handling System. So far, we have treated the idea of clustering routes as a technical problem, being applied locally to each logistic step. However, now that we know the details of how to do this, we are ready to discuss combining these clusters to form sequences of clusters. We will first discuss this before moving on to how to visualize it.

Having already gone through the whole technique in great detail, we currently have enough understanding to review it and understand the application of every step. This has been mentioned before, but to review: when we apply this to the baggage handling system, we actually don't apply it to the whole thing at once. The baggage handling system is a very complex system that is made up of much smaller subsystems, which earlier we mentioned we would call "logistic steps". These logistic steps have their own behavior, have their own unique topology and thus have different movement patterns. For this reason, when we apply this process to the baggage handling system, we apply it to these logistic steps one at a time. And the result is a sequence of clusters rather than a sequence of locations that a bag has taken, going through the system.

To illustrate this, lets go through a small example showing the concept of applying the technique from start to finish. We start with a bag going through the whole system.

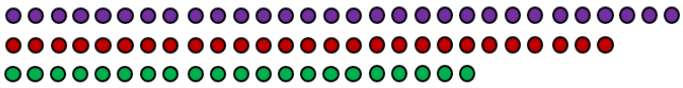

Figure 33: Three complete bags

Here in Figure 33 we have 3 different bags (colored purple, red and green respectively) where the sequence of dots represent the locations visited by the bag. These sequences are formed by simply going through the log and stitching together the locations. Already we can see that in this example that these location sequences are different lengths. Already this is an indicator that they went throug the system differently.

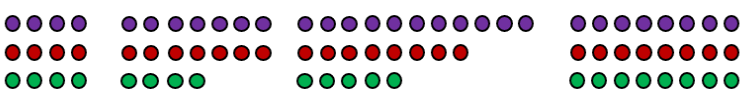

Figure 34: Location sequences split by logistic step

Here in Figure 34 we can see the same sequences, but now they have been split by the logistic steps they go through. For the simplicity of this example, we assume that they all go through the same logistic steps. And already we notice that the 2nd and 3rd logistic step are the cause of these location sequences being different in length. Likely, this is due to the possibility to loop in these logistic steps. Now that our bags are split by their logistic step, we have formed our "bag traces". Now we go ahead and apply our PFA technique to find the principle sets.

Figure 35: Principle locations highlighted in yellow

Here in Figure 35 we can see for each bag trace, the principle locations have been highlighted in yellow. We can see that for some logistic steps, the same sequence of principle locations seem to be found (such as in the 4th logistic step). While in others, there is some variety (such as in the rest of the logistic steps). Also, looking at the 3rd logistic steps, where looping seems to be possible, we see more looping is indicated by a longer variant of principle locations. Regardless, the purpose of this simple example is to demonstrate the process rather than get into analysis. And the next step we do is project the bag traces on these principle sets.

Figure 36: Bag traces projected on principle locations

Here in Figure 36 we can see the results of projecting on the principle locations. Our bag traces have been reduced to a much smaller sequence of locations. And we can notice now, that in certain logistic steps, similar sequences of locations (which are all principle locations) occurring. What we do is declare these different principle location sequences as clusters and label them as such.

Figure 37: Assigning Clusters

Here in Figure 37 we see an example of cluster names being assigned to each of the variants of sequences of principle locations. In this assignment of names, we have a convention where the name of the cluster is split into two different parts, separated by a dash. The first digit before the dash acting as a counter for the logistic step. And the second part after dash, being the letter "C" followed by another digit that acts as a counter for the variant of principle locations within that particular logistic step. Here, it becomes quite clear which bag traces belong to the same cluster. As mentioned before, we can see that in the last logistic step all Low-level variants get assigned to the same cluster. While in the 3rd logistic step, they all get assigned to a different cluster. Now that we have demonstrated the idea, we simplify the names with the intention of visualizing later on.

Figure 38: Relabeling Clusters

Here in Figure 38 we can see the relabeling of the clusters, where we simply drop the counter before the dash. This makes the labels more compact and easier to read. Now that is done, we

are ready to make the transition from our bag traces being sequences of locations to sequences of clusters.

Figure 39: Bags in their cluster form

Here in Figure 39 we see the result of that transformation, where bags are being described by the clusters they go through. So instead of a list of locations in each logistic step, we have a list of clusters that was taken in each logistic step. And essentially, that is the output of the whole process. That is the product that is being produced.

Now what we have is actually a description of what types of routes (because that is what the clusters essentially are) bags are taking when they go through the system. And this becomes powerful when we start grouping bags that have similar performance properties together. But this is something that will be demonstrated in Chapter 9.

Now that we have the idea of the final output in our minds, me move on to what we require from our visualization.

8.2 Visualization Requirements

So finally, we started with bags that were a list of locations going through the system, and now we have a list of clusters per logistic step that bags go through. Now the problem we face is the necessity to visualize this, that effectively summarizes the system, yet contains enough detail to understand the clusters and routes being taken in the system. There are two sides to understanding the cluster that are important to articulate.

On one hand, we want enough context information to be able to interpret how routes are affecting performance. This comes in the form of being able to see how quickly bags are moving through the system (their performance), how many bags are moving through clusters and also being able to see which clusters are unique to routes that contain only slow bags. Further, part of this is being able to see how a bag moves between clusters, what clusters and logistics steps precede and follow a particular cluster of interest. In other words, how bags flow through these clusters.

The second side of understanding the clusters is to be able to understand what this cluster is. What part of the system does it belong to and where do the Low-level sequences inside start and end (in terms of location). This is particularly critical when an interesting cluster is identified and we want to see what the routes are. Formally, we want a visualization that can:

- Show the different clusters being taken by bags
- Show performance (time taken in system) of bags through different parts of the system
- Show number of bags going through a cluster
- Show the context (start and end locations of a logistic step) in which clusters are being taken by bags
- Show clusters unique to routes that contain only: over, at and below system time bags
- Scalable to visualize thousands of bags simultaneously
- Scalable to visualize hundreds of clusters simultaneously

And with a visualization that can deliver these points, we would have the capability to understand how bags are flowing through the system. Particularly, see what clusters are being taken by bags, see how performance varies according to clusters being taken and logistic steps being taken. And also understand what kinds of routes bags are following in each logistic step.

The medium chosen to deliver a visualization meeting this criteria was a graph. A graph, where nodes represent both locations (start and end of logistic steps) and clusters. And edges between these nodes represent bags that have follow a particular sequence of start\end locations and clusters.

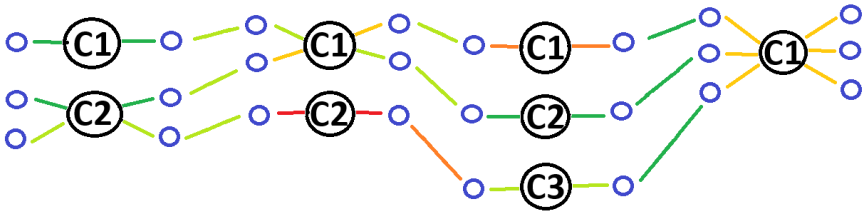

Figure 40: A mock-up of the visualization structure

Here in Figure 40 we see a mock up of the structure of the visualization with certain components of the visualization is highlighted. As first introduced in Section 4.3.4, the components in the visualization are interpreted as follows:

- Blue dots
  The blue dots are the cluster start and end locations for each Logistic step.
- Black Circles
  The black circles are the clusters for each Logistic step.
- Light Blue/Green/Yellow/Red edges
  These are edges that show which routes are assigned to what clusters. These will be colored in a gradient fashion, from light blue, to green, yellow, red and ending at brown. And they will be indicating time spent, going from faster (light blue/green), to slower (red/brown) respectively.

Further, having clusters represented as nodes rather than something like colors, was a design decision with the purpose to make the visualization scalable to the number of clusters that can be visualized. And this also allowed us to use colors as a way to encode performance behavior. Next we describe how the data was formatted to be able to extract the necessary attributes.

8.3 Formatting the Data

Now that we know what we want our visualization to look like and how to construct it, we need to extract necessary components from the data. If you recall 6, the data is in the form shown in Table 11 and has been augmented with the cluster that each row has been assigned to, illustrated in Table 12.

So from this point we have to get the data into a form that can be visualized. Specifically, we want to extract the necessary features that were highlighted in the visualization mockup in the previous section.

To do this, all the bags have to be iterated through one last time to put them into the final format. In this last run through we gather the following:

- Bags in cluster form
  Rather than a list of locations passed through, this is a list of clusters that a bag passed through.

Logistic Step	Location Sequence	Logistic Step Sequence	System Time Category	Number of Bags	Combo ID List	Logistic Step Time Deltas Averages
CHECK_IN	AreaW.xx04:AreaW.xx04:AreaW.xx34:AreaW.xx56:AreaW.xx76	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	3	[0-1,3-2,4-3]	00:00:45,00:00:34,00:00:15, 00:01:14
FIRST_SORT	AreaW.xx04:AreaW.xx04:AreaW.xx34:AreaW.xx56:AreaW.xx76	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-2,3-3,5-6]	00:00:25, 00:00:24,00:00:31, 00:00:14,00:00:03
SCREEN	AreaW.xx04:AreaW.xx04:AreaW.xx34:AreaW.xx56:AreaW.xx76	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	4	[0-3,2-1,4-4,5-6]	00:00:15, 00:00:14,00:00:55, 00:00:34
FIRST_SORT	AreaW.xx04:AreaW.xx04:AreaW.xx34:AreaW.xx56:AreaW.xx76	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	2	[0-4,1-1]	00:00:45, 00:00:34,00:00:15, 00:01:14,00:00:13
LAST _SORT	AreaW.xx04:AreaW.xx04:AreaW.xx34:AreaW.xx56:AreaW.xx76	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	3	[0-5,3-6,4-7]	00:00:05, 00:00:04,00:00:15, 00:00:09
LATERAL	77601399	CHECK_IN:FIRST_SORT:SCREEN:FIRST_SORT:LAST_SORT:LATERAL	Under System Time	4	[0-62-8,5-7]	00:00:21, 00:00:24,00:00:25

Table 11: Example of aggregated bag traces

Location Sequence	Method5	Method 6
AreaW.xx03:AreaW.xx01:AreaW.xx02:AreaW.xx13:AreaW.xx09:AreaW.xx02	AreaW.xx09	AreaW.xx01
AreaW.xx04:AreaW.xx97:AreaW.xx96:AreaW.xx97:AreaW.xx96:AreaW.xx95		AreaW.xx97
AreaW.xx97 : AreaW.xx01 : AreaW.xx99 : AreaW.xx98 : AreaW.xx01 : AreaW.xx03	AreaW.xx01	AreaW.xx03

Table 12: Example of clustered bag traces

- Cluster start locations for each Logistic step
  This is a simple list of all start locations of all bags traces that go through a particular logistic step. These will end up being the blue dots in the mock-up shown in Figure 40.
- Cluster end locations for each Logistic step
  This is a simple list of all end locations of all bag traces that got through to a particular logistic step. These will end up being the green dots in the mock-up shown in Figure 40.
- Clusters for each Logistic step
  This is simply a list of clusters for each logistic step. This will end up being the orange dots in the mock-up shown in Figure 40.
- Bag traces in start\end location form
  Again, instead of a list of locations, this is a list of pairs of the start and end locations for each cluster
- Combo ID to time taken. This will end up being used to determine which of the edges (show as red edges in Figure 40) to create, how thick to make them and the count of bags passing through that route.
  This is a mapping from Combo ID to the average time a bag takes going through that Combo ID's Location sequence.This will end up being used to determine the color of the edges (show as red edges in Figure 40) to indicate the average time taken along that route.

Each of these are gathered for a particular method number and High-level variant. That is because for each method number and High-level variant combination, an individual visualization is created.

Finally, the end product is made in Javascript using a visualization library. This means the visualizations themselves are interactive, they contain a limited form of filtering. This filtering was implemented with the intention of being able to restrict the scope of analysis since there can be an incredible amount of information in the visualization.

8.4 Visualization Sample

Now that we have seen mockups of the visualization, understand how it is constructed and how it represents the data - we are finally ready to see it.

Figure 41 shows a close up of the visualization for a particular logistic steps. Each column of lighter blue dots represents the different possible clusters a bag can go through in a logistic step, and the columns of darker blue dots on each side of a column of orange dots represent the start and end locations, respectively, of that logistic steps. Moreover, if a cluster dot is colored red,

yellow or green - this is an indicator that the cluster is unique to bags under, at and over system time, respectively.

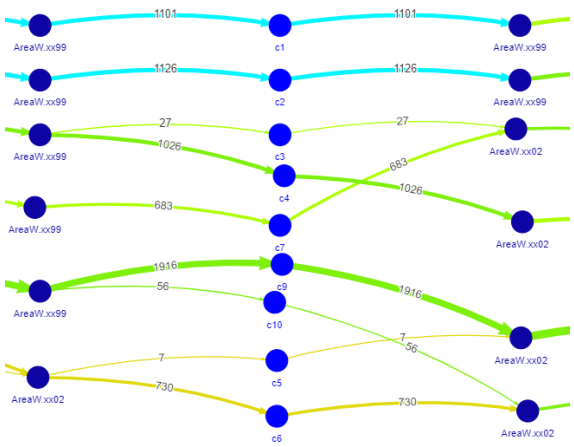

Figure 41: Visualization of a logistic step

The edges between dots show the different routes that bags took when moving through this part of the system. And the colors of the edges represent the average amount of time (in units of seconds) a group of bags took going through that logistic steps. Where lighter green/blue is less time, and darker red/brown is more time. Further each edge has a number, which indicates the number of bags that took that route.

Now, the "whole" visualization is essentially these visualized logistic steps, stitched together into one long graph. However, due to the structure of the visualization, we only visualize bags that have the same High-level variant. This is necessary because we need our bags to have the same length (with respect to the number of logistic steps in the High-level variant) and order of logistic steps. Consequently, bags with longer High-level variants will have longer visualizations, and bags with shorter High-level variants will have shorter visualizations. But now that we understand all the components and the structure, we move on to the whole visualization.

Here in Figure 42 we see an example of the whole visualization for a particular sequence variant of logistic steps.

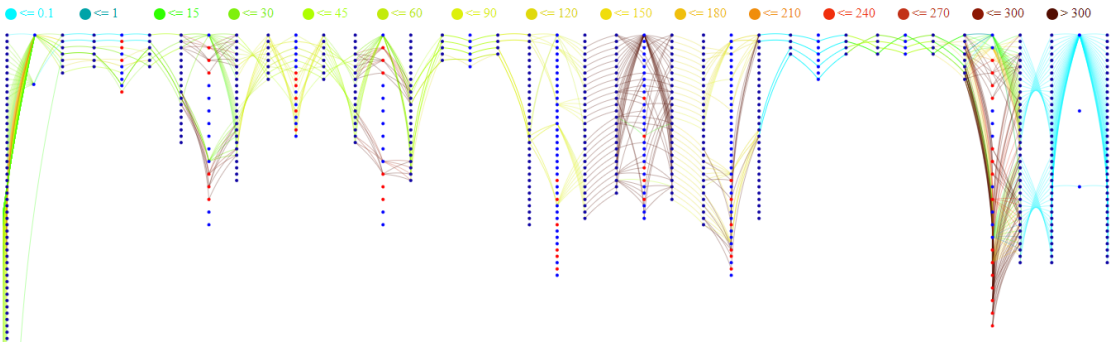

Figure 42: Example Visualization

It is easy to see due to the scale, why it is desirable for the visualization to be interactive. This is type of visualization that we use in the next section to understand not only what kind of routes the PCA technique is clustering together, but also how these clusters affect the performance of bags. And in the next section we will use this when trying to answer our usecases.

9 Results

In this chapter we demonstrate how the method and visualization can be used to gain insights into the baggage handling system. We start in Section 9.1 with a concrete interpretation of the usecases. Next are Sections 9.2, 9.3, 9.4 where a very detailed explanation is given on how each of these usecases are implemented. This is naturally followed by Sections 9.5,9.6, 9.7 where insights found are demonstrated. Finally in Section 11.4 we mention some problems encountered, and reflect on them.

9.1 Usecases Interpretation

At this point we have all the tools available to begin to answer the usecases. However, the problem is that the usecases require some interpretation. So we want an interpretation that is in terms of High-level process, Low-level processes rather than "routes".

If you recall from Section 3.3, the 2 usecases listed below are questions we are aiming to answer.

Use case 0: What routes exist?

Use case 1: What routes are slow and fast?

Use case 2: How does the system affect bag speed?

In usecase 0 we interpret "routes" as Low-level routes. The reason being that, we already know that from the preliminary analysis, that the granularity of High-level variants is not fine enough to be insightful. However, at the Low-level granularity, it is too fine. So with our new clusters, which is a summarized form of the Low-level granularity - we would like to know what the distribution looks like.

We interpret the term "routes" as High-level variants in usecase 1. This means we see usecase 1 as identifying High-level variants that are fast and slow (only contain bags that are fast or slow), and looking for clusters that could potentially explain this.

Complementary, we interpret usecase 2 as identifying High-level variants that have a variety of performance behavior(containing fast and slow bags). And looking for clusters that could potentially explain this deviation. Again, this would be in the form of identifying clusters that are unique to a specific deviation in performance (at, above or below system time).

9.2 Implementing Usecase 0

So now that we know how we want to interpret usecase 0, we need to look for a way to realize this. What we want to see is the distribution of our route cluster among the High-level variants. Further, we want to be able to see this in a compact form.

The solution to achieve this came in the form of a heatmap. In this heatmap we have the y-axis be the different HL-variants we are looking at. And on the x-axis, the different route clusters. Through a heat map, we will have a condensed form to visualize the distribution of clusters across the High-level variants.

9.3 Implementing Usecase 1

So now we have an interpretation of usecase 1 and know that we are dealing High-level variants and clusters unique to these High-level variants. Our next problem is to implement it. For usecase 1, we want to know not only which High-level variants contain only bags that are at, over or below system time but also which clusters are unique to these variants.

Calculating those High-level variants is already a small challenge. Table 11 has to be iterated through to identify which High-level variants contain what types of bags (with regards to system

time category). Once these were calculated, then some additional computation was necessary to find the High-level variants that only contained bags that were over, at, or under system time.

To describe how we computed these, we turn to set theory.

Let L = set of all Low-level traces.
Define $H'(\sigma)$ = High-level trace/High-level variant of σ where $\sigma \in L$
Define $H(L') = \{H'(\sigma) | \sigma \in L' \text{ and } L' \subseteq L\}$
Let $L^+/L^0/L^-$ = The set of complete Low-level traces that are above/at/below in system time.

Note then $L = L^+ \cup L^0 \cup L^-$.
Also note $H(L^+)$ has High-level variants that all contain at-least some Low-level traces above in system time (but also contain Low-level traces that are below and at system time).

Let $U1^+ = H(L') \setminus (H(L^0) \cup H(L^-))$
Let $U1^0 = H(L) \setminus (H(L^+) \cup H(L^-))$
Let $U1^- = H(L) \setminus (H(L^+) \cup H(L^0))$

Our resulting sets ($U1^+$,$U1^0$,$U1^-$) are the sets that contain High-level variants that only contain traces that are above, at or under system time respectively. We will call these the U1 sets. Once the U1 sets have been calculated, we move onto the clusters. We need to extract clusters unique to the variants in the U1 sets. The way to we do this is illustrated in the following set theory logic.

Note, for each $\sigma \in L'$, there exists $\sigma_1, \sigma_2, ...\sigma_n$ where $\forall \sigma_i$ with $a < i <= n$ are the subtraces for each logistic step in the High-level variant $H(\sigma)$

Define $Split(\sigma) = \sigma_1, \sigma_2, ...\sigma_n$ where $\sigma \in L$.

For a subtrace $\sigma_i \in Split(\sigma)$ where $\sigma \in L$,
Define $CLM(\sigma_i)$ = the cluster that σ is assigned to.

Define $CL(\sigma) = \{CLM(\sigma_i) \forall \sigma_i \in Split(\sigma)\}$ where $\sigma \in L$

Define $C(H') = \{CL(\sigma) | \text{ for all } \sigma \in L \text{ and } H(\sigma) \in H'\}$

Note that the sets $C(U1^+), C(U1^0), C(U1^-)$ now are all route clusters occurring in each of High-level variants that only have traces that are over,at or below system time. However, it is possible for there to be overlap of route clusters. So an extra calculate has to be made to get rid of these.
Let $U1_{C+} = (C(U1^+) \setminus C(U1^0)) \setminus C(U1^-))$
Let $U1_{C0} = (C(U1^0) \setminus C(U1^+)) \setminus C(U1^-))$
Let $U1_{C-} = (C(U1^-) \setminus C(U1^+)) \setminus C(U1^0))$

Now, what we are finally left with in the sets $U1_{C+}, U1_{C0}, U1_{C-}$ are the clusters that are unique to High-level variants that are over, at and below system time.

9.4 Implementing Usecase 2

So now we move onto usecase 2, which we interpreted as identifying High-level variants that have a variety of behavior. This means High-level variants that contain bags in all 3 system time categories (over,at and under system time). So the problem now is how to calculate this. We not only want to know which High-level variants contain this variety of behavior, but also the clusters that are unique to traces at are at, over and below system time, within a particular High-level variant.

Again, we refer to set theory logic to describe these calculations.

Let $U2 = H(L+) \cap (H(L^0) \cap H(L^-))$

So borrowing off of previously defined sets, we illustrate here how we calculate our set of High-level variants that contain traces from all three time categories (above, at and below system time). However, before we move onto the clusters, we have to make 3 additional sets of Low-level variants which is shown below.
For each $v \in U2$:
Let $L +_v (v) = \{x | x \in L^+ \text{ and } H(L^+) = v\}$
Let $L0_v(v) = \{x | x \in L^0 \text{ and } H(L^0) = v\}$
Let $L -_v (v) = \{x | x \in L^- \text{ and } H(L^-) = v\}$

Here, for each High-level variant that has traces of all 3 time categories, we create 3 different sets of Low-level variants - one for each time category. In other words, each of these 3 sets has Low-level variants belonging to the same category, and all share the same High-level variant. Now that we have these sets, we are prepared to move to the clusters.

For each $v \in U2$:
Let $C_v^+(v) = \{CL(\sigma) | \sigma \in L +_v (v)\}$
Let $C_v^0(v) = \{CL(\sigma) | \sigma \in L0_v(v)\}$
Let $C_v^-(v) = \{CL(\sigma) | \sigma \in L -_v (v)\}$

Now for each variant in $U2$, these sets (C_v^+, C_v^0, C_v^-) are all route clusters occurring in each of the Low-level variants that all belong to the same High-level variant, but different system time category. And again,it is possible (in fact very likely) for there to be overlap of route clusters between these sets. So more calculation has to be done get rid of these.

For each $v \in U2$:
Let $U2_v^+ = (C_v^+ \setminus C_v^0) \setminus C_v^-)$
Let $U2_v^0 = (C_v^0 \setminus C_v^+) \setminus C_v^-)$
Let $U2_v^- = (C_v^- \setminus C_v^+) \setminus C_v^0)$

Now, what we are finally left with in the sets $U2_v^+, U2_v^0, U2_v^-$ are the clusters that are unique to the different time categories of Low-level variants for each High-level variant that has traces of all 3 system time categories.

9.5 Usecase 0 Results

So we start with usecase 0. Here we aim to get an understanding of how our route clusters are distributed across the top High-level variants. We start with figuring which High-level variants to include in our heatmap.

A quick investigation reveals that the top 40 High-level variants contains 23,325 traces. Out of a total of 28,048 traces in the system, that is 83.2% of all bags in the system - thus capturing the majority of the behavior in the system. Thus, we choose the top 40 High-level variants. And in the same grain of thought, we choose the top 50 route clusters to display on the x-axis. The resulting heatmap can be seen below in Figure 43.

In this heatmap we can see the distribution of occurrences, which turns out to be nothing interesting. Since the first row is the most common variant, and the first column is the most common cluster, naturally it has the most number of occurrences. And we see the number of occurrences of the first cluster in the most common variant dwarfs the occurrences of the other clusters in other High-level variants. Thus, in the next heatmap we scale the values by the number traces in each High-level variant. To do this, the values each row are modified by dividing the value by the total number of traces in that High-level variant. Thus the values now represent a

Figure 43: Heatmap showing number of cluster occurrences per High-level variant

more proportional value for frequency of occurrence of clusters. The result can be seen below in Figure 44.

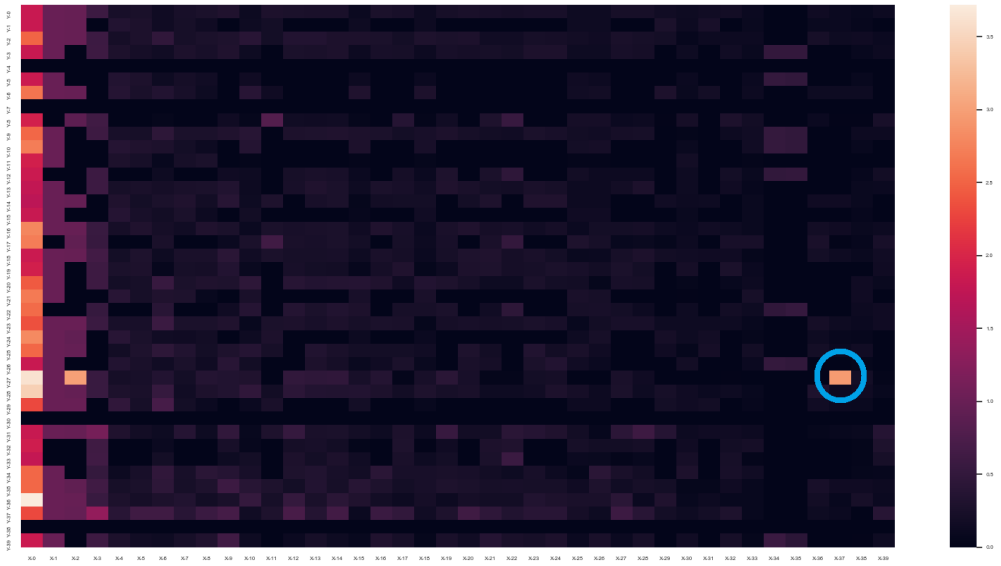

Figure 44: Heatmap scaled by number of bags in each High-level variant

And immediately we can spot an outlier cluster X-37 on the far right highlighted in light blue. Closer inspection of the High-level variant revealed that this was a variant that contained 3 instances of the Check-In logistic step. Clearly an interesting finding, and one that we couldn't explain. So this was something we plan to ask the engineer.

9.6 Usecase 1 Results

Next, we move to usecase 1. What we hope to find is that our technique and visualizations are useful, and are able to provide relevant insights to usecase 1. We start with taking a quick look at what kind of High-level variants we have in our usecase 1.

To recall, in usecase 1 we consider High-level variants that contain bags of only 1 type of time category (at, over or below system time). A quick investigation revealed that of the total 660 High-level variants seen in the system, 580 fell into usecase 1. And further, the majority of these turned out to be over system time. That is, out of 580 High-level variants in usecase 1, 547 of them contain only slow bags. And recalling the histogram previously shown in Chapter 3 (Figure 13) which showed the distribution of bags across High-level variants. Below in Figure 45 we shown a similar histogram. However, now each of the time categories are highlighted to form a stacked bar chart. In Figure 45 the legend uses symbols to indicate the time category, where "+" indicates over system time, "-" indicates under system time, and "0" indicates at system time.

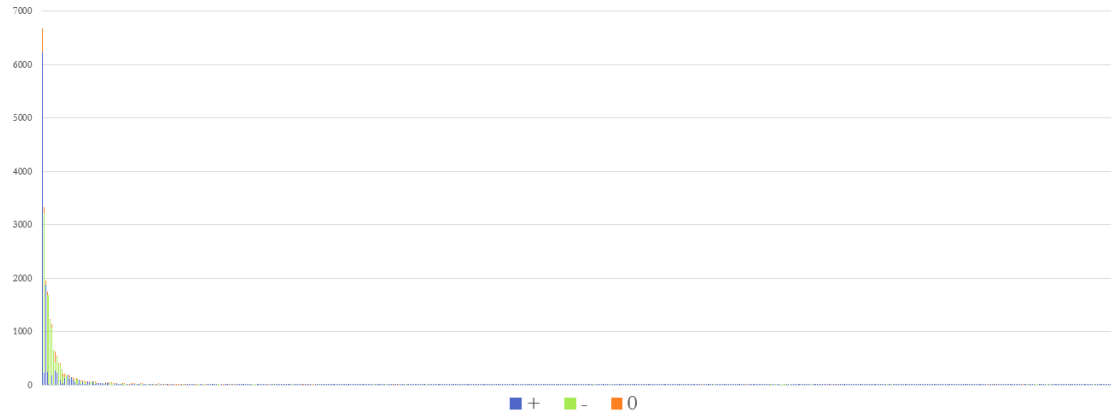

Figure 45: Stacked histogram showing distribution of number of bags per High-level variant

Because of the scale and difficulty to see the histogram in 45, we have a split it into two parts: the top 40 shown in Figure 46 and the bottom 620 in Figure 47.

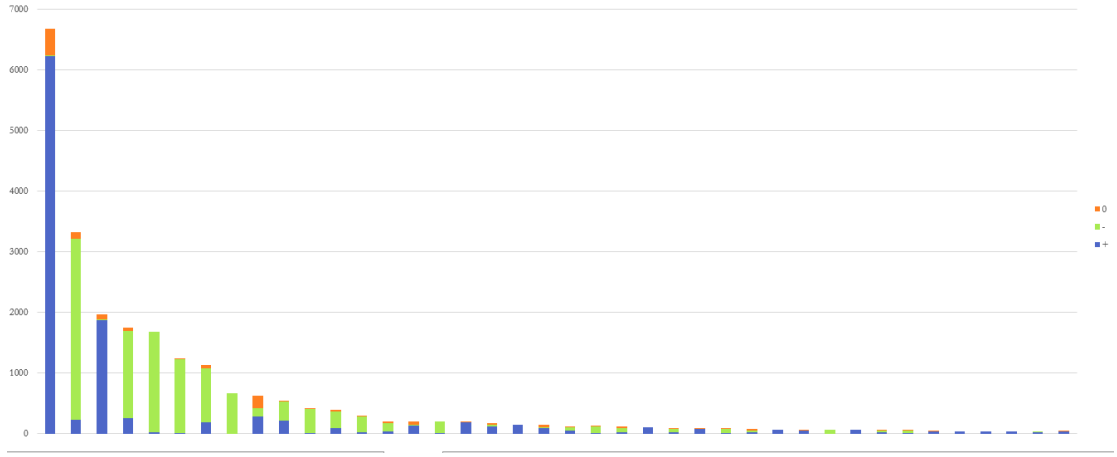

Figure 46: Stacked histogram of top 40 High-level variant

Here in Figure 47 we can see that the long tail of the distribution is just blue, thus indicating that these variants contain only bags that are over system time. And with so few bags present in these High-level variants that are in the tail of the histogram, it seems logical that it is easy for these High-level variants to contain bags that all belong to the same time category.

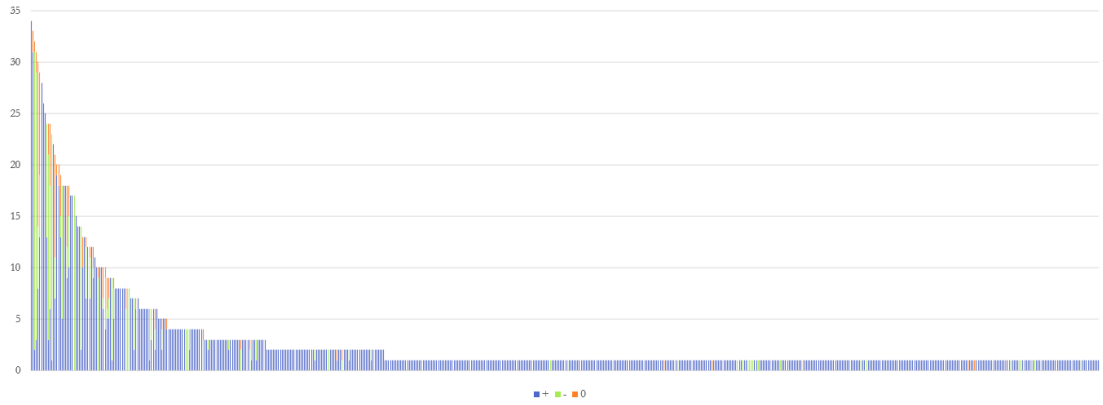

Figure 47: Stacked histogram showing bottom 620 High-level variant

And because of the huge disproportionate distribution of High-level variants containing only bags over system time, it was impossible to do effective analysis using the visualization and technique. The visualization is designed to look at one High-level variant at a time, and not over 500 simultaneously.

Further, the other High-level variants containing bags that only belonged to the other time categories were inconsequential due to the disproportionate distribution. There were only 27 High-level variants in the under system time category, containing a total of 793 traces. However, the majority of these (666) belonged to a single High-level variant, that revealed nothing interesting when investigated. Thus leaving only 127 traces. And in the At system time category, only 6 High-level variants were in that set, with only 6 traces total.

So unfortunately, this usecase 1 was not ideal to apply the visualization and technique. This is a discovered limitation and is an important observation for the future. However, in an attempt to gather some type of insight, we did apply the heatmap to the set High-level variants that only contained bags that were overtime. The other High-level variants of other time categories were also considered, however they simply did not not contain enough bags to be significant enough. This heat map for the top 40 High-level variants that only contain bags over system time is shown below in Figure 48

Again, the heatmap shown in Figure 48 Shows a similar distribution as Figure 43 where we don't normalize the values. However, despite that, there was a cluster in the heatmap that did slightly stand out - highlighted in blue. Again, closer inspection of this cluster revealed nothing to us, and required domain knowledge to interpret further.

Then once again, the scaling of values was applied to check if any additional findings where possible. The result of this is shown below in Figure 49.

Here, the most interested identified cluster is highlighted in blue. Interestingly, this cluster was not the same, but belonged to the same logistic step and was similar to the cluster identified in the previous heatmap (Figure 48). However, just like before, closer inspection of the cluster required a deeper level of domain knowledge in order to interpret.

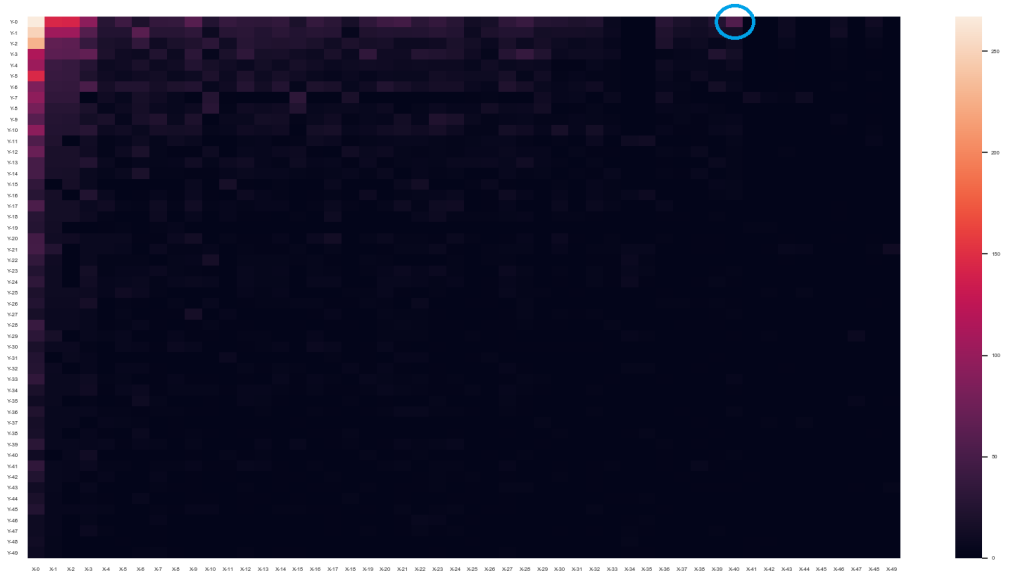

Figure 48: Heatmap showing number of cluster occurrences per High-level variant

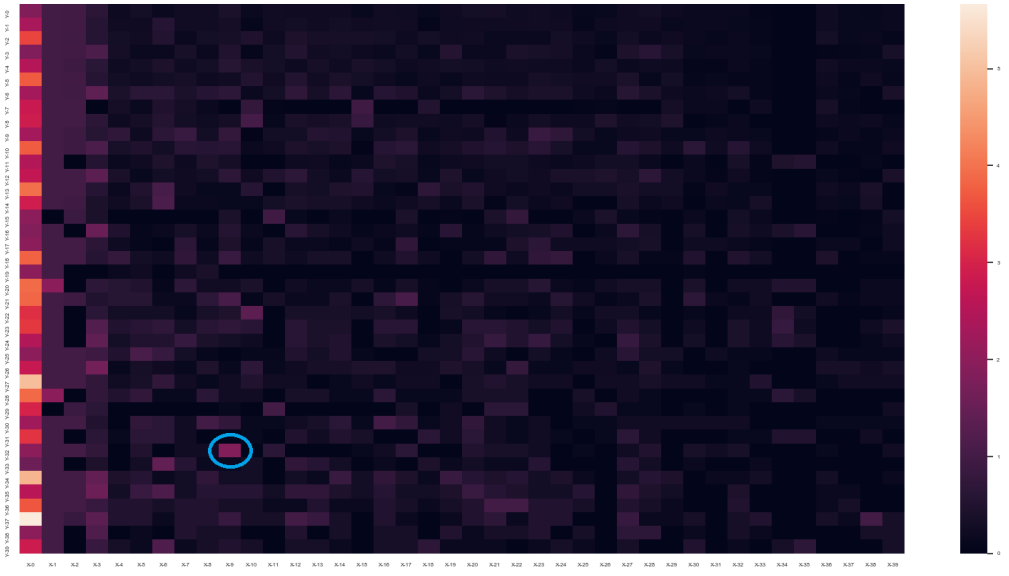

Figure 49: Heatmap scaled by number of bags per High-level variant

9.7 Usecase 2 Results

So now we move onto usecase 2 where we also aim to demonstrate what kind of findings relevant to usecase 2 are possible using both the technique and the visualization. To start, we begin with discussing what kind of High-level variants fell into usecase 2.

In usecase two, we were looking at High-level variants that had a range of behavior occurring (fast and slow traces). Out of the top 30 High-level variants (ordered ascendingly on number of traces contained with in each variant) 25 fell into usecase 2. This came out to a total of 21,779 traces that fell into usecase 1. Out of a total of 28,048 traces in the system, that is 77.6% of the traces covered by usecase 2. This indicates that usecase 2 is the usecase that captures more of

the typical behavior seen in the BHS, and so any findings would be applicable to the majority of cases.

To illustrate our findings, we looked at the visualization of the most common High-level variant. And specifically, we look at the First Sorter logistic step.

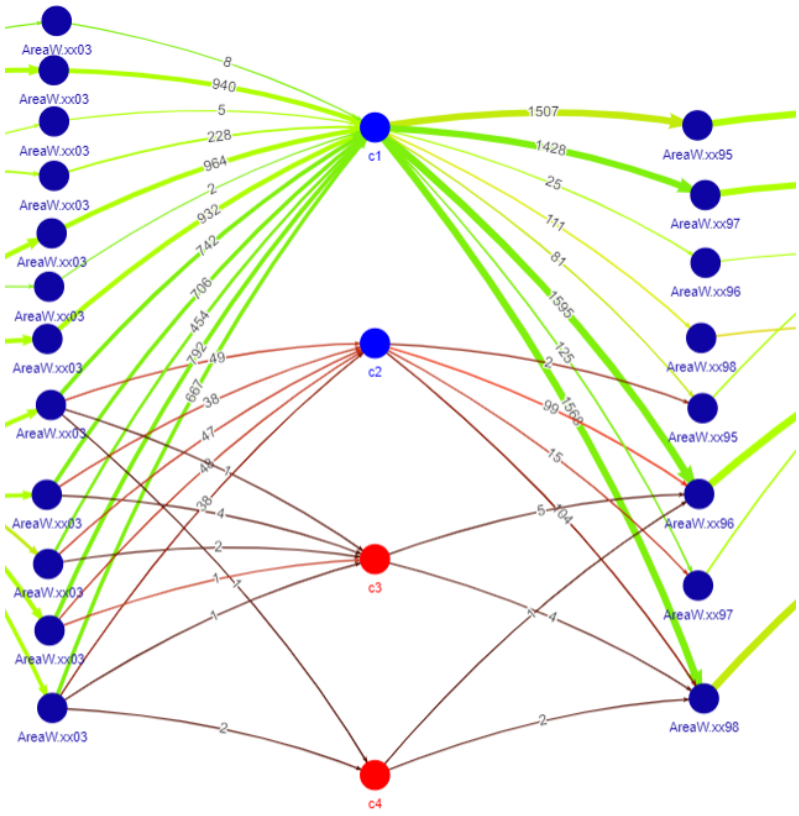

Figure 50: Results seen in First Sorter

Here in Figure 50 we see the visualization, and notice immediately that both route clusters 3 and 4 are colored red and all edges going to and from these clusters are red as well. Red edges indicate that the average time taken by bags going through this route is higher, making bags slower. And red clusters indicate that these are clusters used only by traces that are over system time in this High-level variant. Clearly whatever routes this route cluster is capturing, bags are slow when taking that route.

Closer inspection of route clusters 3 and 4 reveal that these are summarizing looping behavior. If you recall Figure 7 from Chapter 3, First sorter is one of only 3 logistic steps in the system where looping behavior can occur. This explains the red edges nicely - when a bag loops, undoubtedly it will take longer. And what we are seeing in this variant is that when bags loop, they not only take longer in this part of the system, but they also become slow overall (that is they become a route that is above system time). The red edges is evidence that bags are slowing down in this part, and the red clusters indicate that only late bags follow this route.

To dig a little deeper into the potential effect route clusters 3 and 4 could have on performance, we take all 25 of our High-level variants into account.

Starting with Route cluster 3, a quick survey of the other 25 High-level variants reveal that route cluster 3 occurs in 21 other High-level variants. And 12 of these times the cluster is colored red. Meaning that over half the time when cluster 3 is taken in a High-level variant, it is unique to a bags that are over system time. This does not mean that when taken, half the time the bag will be slow. However, it most definitely is a factor.

Now with route cluster 4, we see it occurring only 6 times in the other 25 High-level variants. However, every time it occurs, the cluster is colored red. This means the only bags that ever follow this route cluster are bags that are under system time. In other words, there is no bag we have seen that is at system time or below system time, and has followed cluster 4. Now although this doesn't necessarily prove that cluster 4 causes those bags to be above system time, it is certainly a strong indicator.

Next, we looked at another logistic step, serving as a link between the Check-In logistic step and the first sorter logistic step.

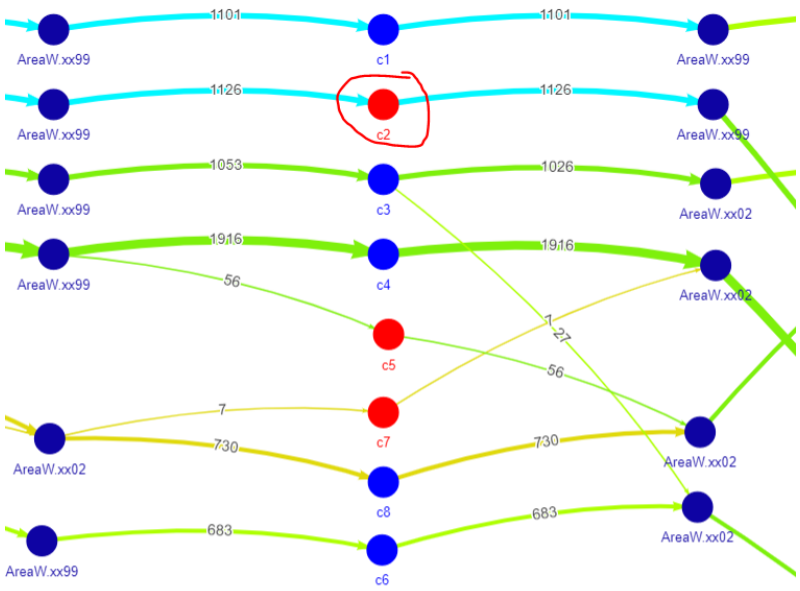

Figure 51: Results seen in link logistic step

Here, the interesting cluster is cluster 2, highlighted with the red circle in Figure 51. The curious fact about this cluster is that it contains a significant number of bags (over a thousand) that all ended up being slow. From the top 30 variants, half of them contain this logistic step. And out of these 15, only 2 of the variants had a significant number of bags in cluster 2 and had the cluster colored as red. This other variant can be seen in Figure 52.

For both these variants, which are colored red and have a significant number of bags passing through, it was not possible to determine what was slowing these bags down. Thus, it was determined that possible someone with more domain knowledge could shed more light on a possible explanation.

So these types of findings are possible to discover with the visualization and the technique. Although it doesn't necessarily provide root-cause analysis (why a bag is slow) it narrows the area and the bags that have to be further investigated to discover that. Since the main idea of the technique was to summarize data with the intent of just being able to understand what was happening, we find this to be acceptable.

In the next section we discuss an evaluation that was done with a system engineer, and the reactions and thoughts captured when presenting the visualization and technique.

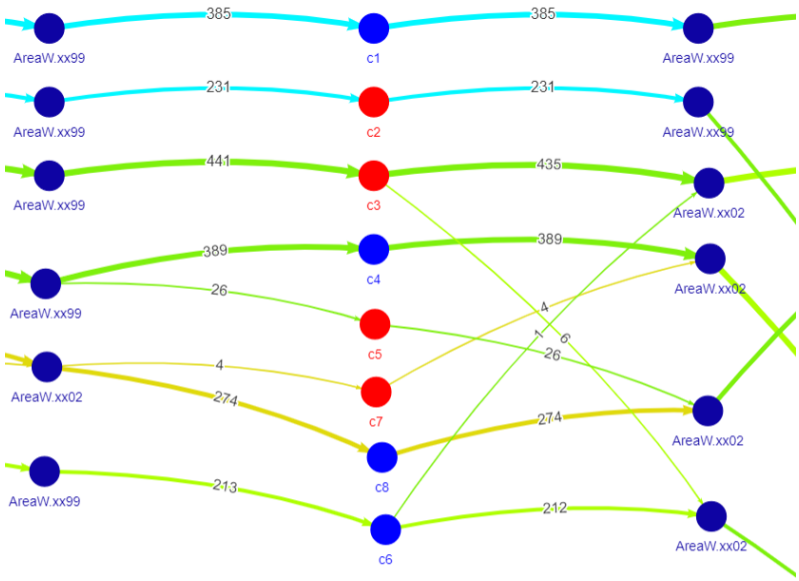

Figure 52: Results seen in link logistic step

10 Validation

In this chapter we go over the validation done with an engineer familiar with the baggage handling system analyzed. We start with a statement in Section 10.1 of what the objective of this validation is. We then move onto a summary of the necessary practicalities to setup our validation in Section 10.2. This is followed by a quick explanation of how we executed our validation in Section 10.3. Finally, we present our results from the validation in Section 10.4.

10.1 Objective

The purpose of this evaluation was to present the different usecases to an engineer familiar with the baggage handling system analyzed and gauge the meaningfulness of the findings. An engineer was required for this because determining how meaningful a cluster demands extensive domain knowledge. Further, the baggage handling system is so complex that it would have not been possible to acquire the necessary domain knowledge within the span of the project.

10.2 Setup

The data we used was from a single day in September. September was chosen because it was considered an average day (not high season or low season). For this single day of data we pushed it through the pipeline described in the previous chapters (summarized in Chapter 4 and described in more detail in Chapters 5 - 8).

The 3 usecases were then prepared as described in Chapter 9. And the results that were demonstrated in Chapter 9 were chosen to show the engineer. We considered these to be the most interesting and limited our presentation to show just these findings to the engineer due to limited time availability.

To prepare for the meeting, a presentation with an explanation of the technique and relevant findings was created. Although it would be possible to demo the findings in the tool directly, it was considered to be more efficient to have them as images in slides (similar to how they have been presented in this thesis). As there was a lot of material that we wanted to cover, this was important to consider. However, if the situation did arise where the engineer did want to dive deeper into a finding, all visualizations were pre-loaded in the browsers and ready to be used.

10.3 Execution

The discussion with the engineer was split into two parts. In the first part, the technique was introduced to the engineer. This included an intuitive explanation of the technique and some examples to illustrate the mechanics involved. We thought that the engineer being aware of how and why the technique was clustering routes together could be helpful for two reasons. One, it would be possible for him to mention improvements since he is much more knowledgeable than we are about the system. Two, he would be able to better explain the findings we would be showing, knowing how it works on the back-end.

In the second part, all the 3 usecases findings (explained in Chapter 9) were shown to the engineer. Here each finding was presented and discussed one at a time. A lot of the discussion involved consulting the Node Segment Diagram (the diagram of the physical layout of the baggage handling system) and understanding what kinds of routes were in the cluster being shown. During this discussion, the engineer's interaction with the finds and tools were all noted (his comments on usability, features he found useful and his thought process for diagnosing possible interpretations).

10.4 Results

Here we describe the comments and insights mentioned by the engineer for each usecase and demonstration of results.

10.4.1 Usecase 0

To recall the results from Usecase 0 in Section 9.5, the most interesting finding was in the second heatmap shown below in Figure 53.

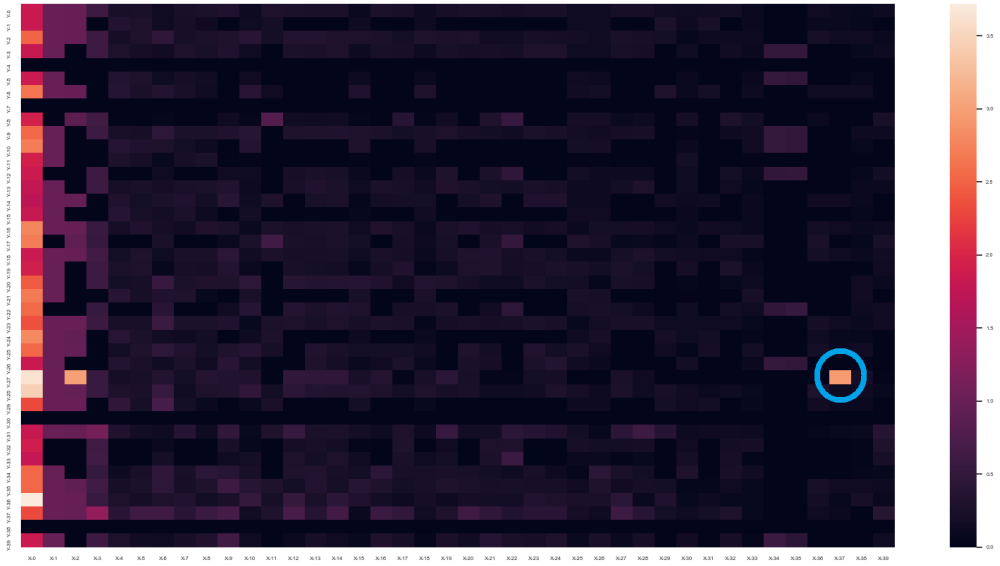

Figure 53: Heatmap scaled by number of bags in each High-level variant

In the heatmap, the High-level variant of the cluster highlighted in the blue circle has the Check-In logistic step occurring 3 times, which is quite odd. The engineer explained this could be caused for several reasons. It could be that the bag missed the flight, lost tracking or the passengers were not allowed to fly. When a passenger is not allowed to board a plane their luggage is also not allowed to be loaded. However, inspecting the cluster, which had 73 bags passing through, revealed that even though they came from several different Check-In counters - they took the same route through the link to first sorter. In fact, all 73 bags took that exact same route all 3 times that they got checked in. Furthermore, inspections of the Check-In counters they came from indicated that this was not the route they normally would take.

A closer inspection of these lines with the engineer revealed that due to the configuration of these check-in counters 2 different conveyor lines merged into a single line, where a priority hierarchy was in place. Thus if there was sufficient load already on the line, potentially the bags could be forced to a different route that they normally wouldn't take. Alternatively, the conveyor equipment could have been broken. Nevertheless, additional log information would be necessary to confirm either of these scenarios.

Regardless, even though more information would increase the value, this doesn't take away from the fact this cluster is highlighting unusual behavior. This was our goal and this confirmation from the engineer was the best we could have hoped for.

10.4.2 Usecase 1

As reminder, in Usecase 1 shown in Section 9.6 there were 2 heatmaps with results. The first is the one shown below in Figure 54

And the second one below in Figure 55.

Both of these highlighted similar clusters belonging to the same logistic step. Inspection of this cluster with the engineer revealed that this was a route that could lead from the auto sorter to the last sorter. However, in this part of the system bags are not allowed to be transported right after each other. In other words, there is a reduced capacity. Moreover, this could cause

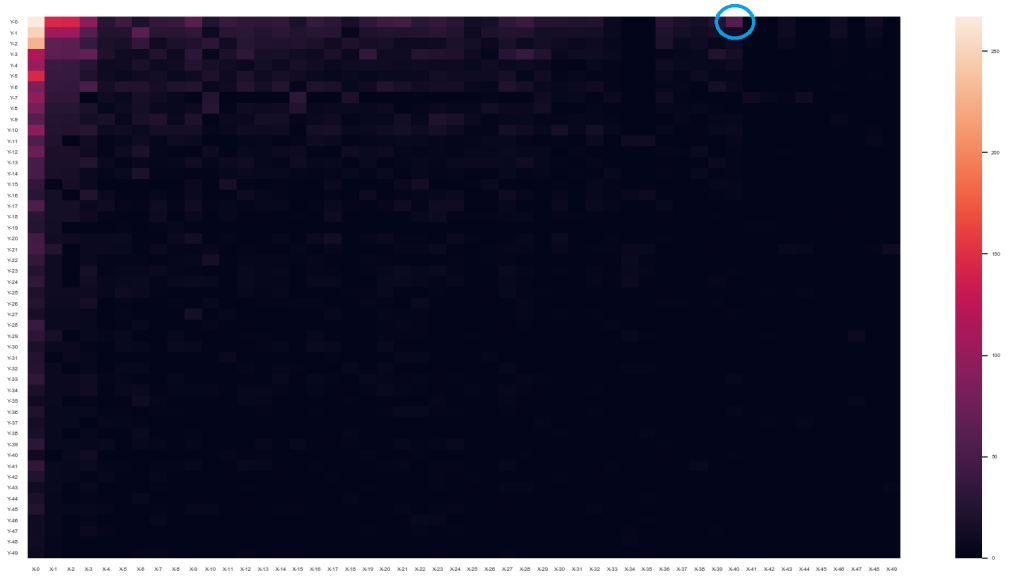

Figure 54: Heatmap showing number of cluster occurrences per High-level variant

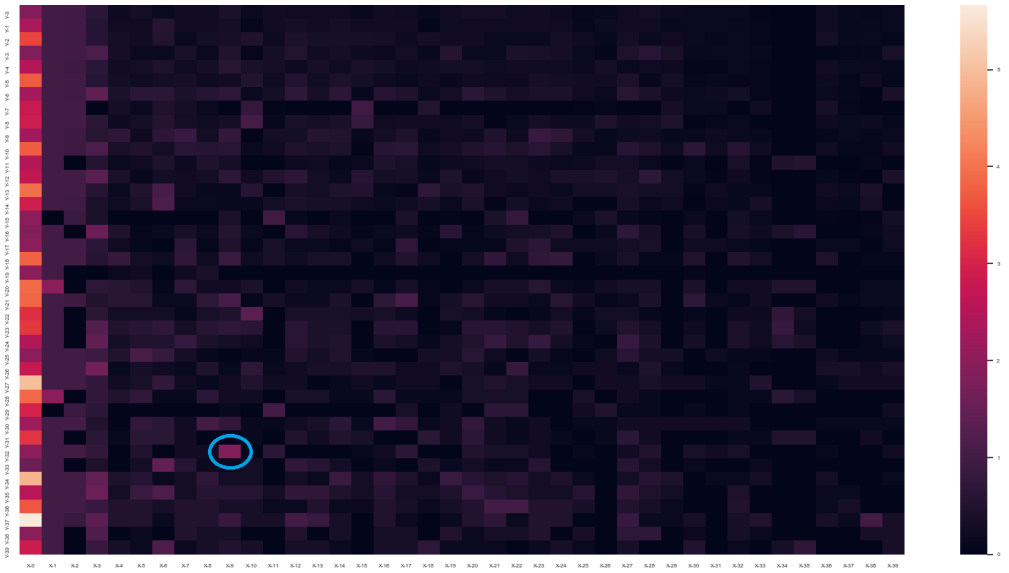

Figure 55: Heatmap scaled by number of bags per High-level variant

congestion. However, this would have to be seen over several days and the load in this part of the system would have to be taken into account to properly diagnose this.

Once again, the necessity for additional information does not take away from the fact that the method is in fact identifying clusters that are meaningful. And the fact that it is meaningful we interpret as a success.

10.4.3 Usecase 2

Finally, to recollect to Section 9.7, we had one significant finding in usecase 2 where looping behavior identified in the first sorter. This is seen by the red clusters in the following Figure 56.

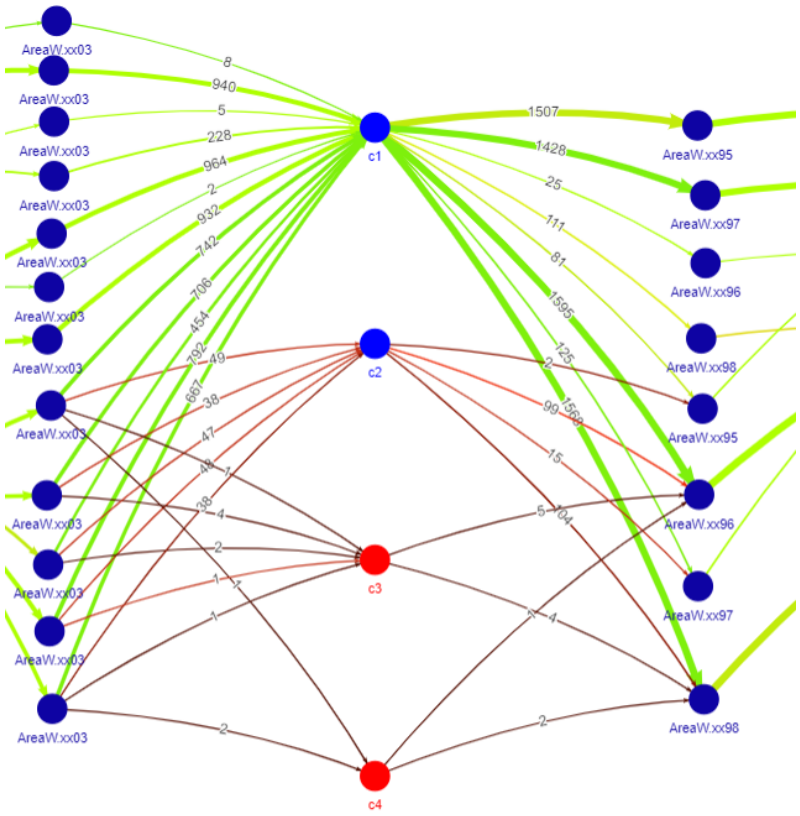

Figure 56: Results seen in First Sorter

The comment here was that although a bag may loop, it does not mean that a bag will miss its flight. Hence, although it might non-optimal use of the system, the true consequences depend on whether or not the bag misses its flight. Furthermore, 2 possible causes were identified as to why a bag may loop in the First Sorter. These two causes are the Screen logistic step and the bag information readability which we explain next.

First is the Screen logistic step. Although the Screen step is a different logistic step than First Sort, the two are related and affect each other. In the Screen logistic step, there are different screening machines that a bag must pass through to clear the security requirements. However, all machines might not necessarily be active. That is because a screen operator is required, and if no operator is present, then that screening line is not active. And with less screening lines active, then a bag might be required to loop in First Sort before it can enter a screening line. However, confirming this would require the logs of the screening machines to verify.

Second is the ability for the bag's information to be read. If a bag's barcode information can't be read or no flight information can be found, then the bag gets routed to different areas. This can cause the bag to loop in order to reach its new destination and could be a possible explanation for the looping behavior. However, verifying this also would require additional log information.

The next finding in usecase 2 was the mystery cluster 2 in the link logistic step. To recall, the cluster is shown below in Figure 57.

Here, this cluster was more closely investigated. The engineer identified that this route cluster 2 was only taken when bags would be checked in from certain Check-In counters. Additionally,

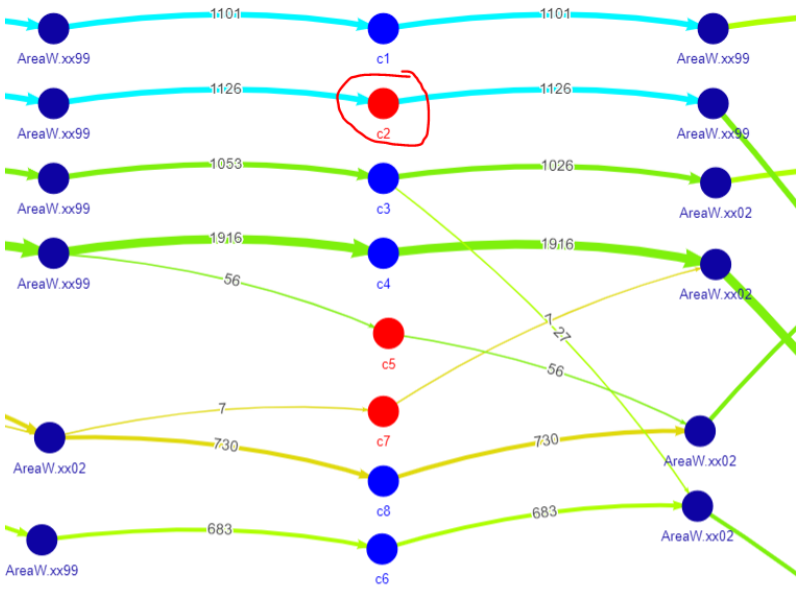

Figure 57: Results seen in link logistic step

due to the configuration of the equipment, it was possible that alternative and slower configuration equipment was being used. However, to confirm this, equipment availability logs would have to be checked.

Just like the previous findings, we were very happy with the fact that these clusters were meaningful. Even though more information would be needed to confirm the findings, these validations of meaningfulness are a necessary and critical step to turning these into actionable insights.

Now that we have validated our results successfully with the engineer and are happy with the findings, we are ready to move onto the next chapter where we conclude the thesis.

11 Conclusion

In this chapter we conclude the thesis. We start by articulating the contributions of the thesis in Section 11.1. Then in Section 11.2 we revisit the research questions and explain how they were answered. This is followed by Section 11.3 where we illustrate the found limitations of our method. We then begin to wrap up by mentioning the many possibilities for future work in Section 11.5 and we finally conclude with some last words in Section 11.6.

11.1 Contributions

What we have demonstrated in this thesis is that PFA can be used to create a route clustering technique that works on logistic route data and clusters routes meaningfully. This involved using PCA to identify the most critical locations of a route. Then using clustering and distance measures to create the optimal principle set. With the optimal principle set identified, form clusters of bag traces. Finally, to visualize these clusters, we created a new graph visualization capable of showing the proper context of these clusters. We list the contributions of our work more specifically below:

1. **Novel event log abstraction using PFA**
   This is a contribution because it is a novel approach to log abstraction. Using PFA to identify the critical events in a trace and then creating clusters from these critical events, and summarizing the log by replacing the events with these event clusters has never been done in process mining. Although there are some papers on applying PCA to graphs, clustering traces for process discovery and some other papers on log abstraction - specifically using PFA to cluster events to abstract the event log has not been done.

2. **Creating meaningful clusters of Low-level routes**
   With the event log abstracted using clusters, the missing granularity for routes in Vanderlande baggage handling systems now can be automatically generated with this method. These clusters group similar routes together and summarize how bags move through each logistic step in the system. This summary is important because it is not only a demonstration of how the novel technique can be used but also provides a strong basis for future capabilities for Vanderlande such as prediction and actionable insights. And both of these capabilities directly result in gained value.

3. **Portability to all Vanderlande baggage handling systems**
   Due to the selection of domain knowledge used (the use of areas mentioned in Section 7.1.2), this method is applicable to all Vanderlande baggage handling systems. Although the effectiveness might vary (there is no guarantee on the consistency of how locations are grouped by area) since every Vanderlande baggage handling system does have areas the method should have a reasonable degree of effectiveness. This means applying the method on another system should require little if any adaptation. Consequently, this makes the method capable of extracting that much more value.

4. **Novel trace visualization**
   This visualization is capable of summarizing the clustering present in a High-level variant, yet contains enough context information to get into the details if necessary. This is a concept that could potentially not only be applied to other domains (other than process mining) but also it is an interactive tool that a user can use. If extended with sufficient features, it could become a prototype of a tool that Vanderlande could either sell to clients or use themselves to offer additional consulting services.

11.2 Research Questions

Here we recall our research questions and discuss how we fulfilled or why we were not able to fulfill them.

1. **RQ1: Can we create a clustering technique that allows us to identify unusual routes, or abnormal behavior?**
   The results of usecases 0 and 1 (explained in Sections 9.5 and 9.6) and the validation by the engineer (explained in Sections 10.4.1 and 10.4.2) demonstrated that we were successful. By analyzing the heatmaps presenting in these usecases we were able to identify clusters that were occurring at an unusual frequency. This led to the identification of some interesting clusters, that could be caused by abnormal behavior. This was verified by the engineer as abnormal behavior, which meant that it was a meaningful finding. And thus we consider this question entirely fulfilled.

2. **RQ2: Can we create a clustering technique allows us to identify optimal, sub-optimal and non-optimal routes?**
   The result of usecase 2 (explained in Section 9.7) showed that we could successfully answer this to a degree. As verified by the engineer (more details can be found in Section 10.4.3) the looping behavior detected is certainly a clear indicator of non-optimal behavior. Further, (as initially discussed in Section 9.7) it was noted that simply looping does not guarantee that a bag is categorized as an over system time bag. Although, looping does seem to make it more likely for a bag to be over system time. Additionally, when certain looping behavior is observed (more than twice) the bag is always observed to be over system time. Thus, it could be argued that non-optimal routes are ones where bags loop twice or more, sub-optimal routes are routes that contain loops, and optimal routes are ones that contain no loops.

   However, it is needless to say that this is incomplete. It is incomplete because we can't confirm that all bags that are over system time loop and all bags that don't loop are above system time. Furthermore, we believe the cause of the inability to answer this question stems from the way we applied the technique rather than the technique itself. This point is discussed in more detail in Section 11.3.1. Thus, we consider this question partially fulfilled.

11.3 Limitations of Method

11.3.1 False Positives

The strategy of identifying clusters unique to bags that were above, at or below system time is not without drawbacks. To illustrate this, we look at the screening logistic step. Here, contradictory colorings were observed.

In Figure 58 we see an example of the clusters in the Screen logistic step. Immediately it is noticeable that there are a large number of clusters colored red here. To explain these red clusters, we have to first recall the fact that the Screen logistic step sits within the first sorter logistic step. That is, the only way to enter the screen logistic step is from the first sorter. And the only logistic step Screen can exit to is the first sorter. Now the first sorter consists of 2 loops - each with different area codes. This means the Screen logistic step can receive input from either of these loops and likewise output to either loop. Now we will call the phenomenon of receiving input from one loop and outputting to the other (different) loop as *Switching Behavior*. We call it Switching Behavior because the bag switches loops.

Closer inspection of these red clusters revealed that most of these clusters (5/6) are highlighting Switching Behavior in this logistic step, which seemed promising at first. However, an inspection of the following High-level variants revealed the following Figure 59.

Here we see that rather than red the clusters are colored green. Although they are not exactly the same clusters, all previously red clusters are now either green or not colored at all. Now switching behavior is generally not common meaning not too many bags follow Switching Behavior routes. And this is most likely the cause of the inconsistency of the coloring. In the previous variant (Figure 58) 93% of the bags in that variant were over system time. In the variant shown in Figure 59, 82% of the bags were under system time. Thus, we actually see that these clusters are simply reflecting what the majority time categorization is. If most bags in the High-level variants are

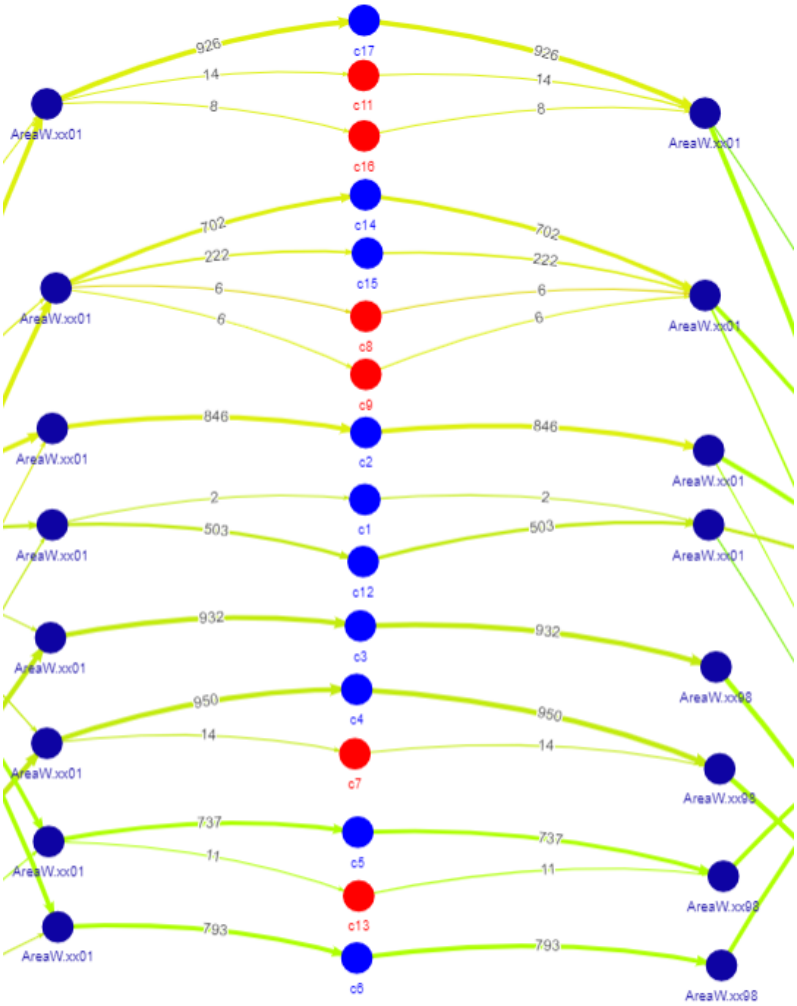

Figure 58: Results seen in screen logistic step

below system time, then the clusters are colored green. And if most of the bags are over system time, then the clusters are colored red.

What this illustrates is that rather than thinking in terms of absolutes (clusters that only appear in a certain time category) it would be a better idea to think in terms of percentages. Thus, the percentage of bags under system time in a cluster or percentage over system time. And then highlight clusters using some type of percentage criteria relative to the percentage of bags in the whole High-level variant. For example, let us take a High-level variant that contains 90% of bags over system time If there is a cluster that contains 85% of bags under system time, this is a cluster that should be highlighted. This cluster is drastically different likely will reveal something interesting.

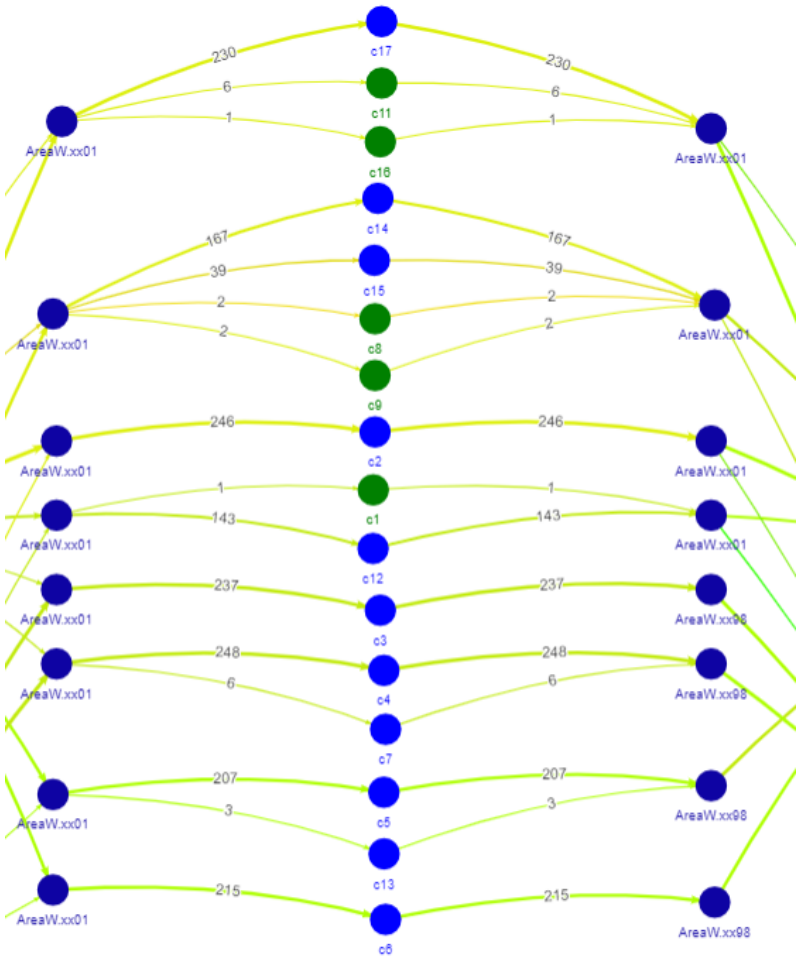

Figure 59: Results seen in screen logistic step

11.3.2 Lack of Root-Cause Analysis

Although we can identify interesting routes, either because they appear at a higher frequency than is expected or because they only contain slow bags, with the current method it is not possible to explain why. That is, the visualizations and heatmaps do not contain enough information to come up with a cause for why only slow bags take those routes or why a set of routes are appearing more frequently than normal.

This was particularly noticeable during the validation discussion of the findings with the engineer. A cluster, or set of routes, would be identified as "interesting", some theories as to why that could be the case were identified but nothing definitive could be established. More context information (equipment availability, system load information, routing preferences) was necessary to confirm a theory. Thus although the technique was effective in identifying where something interesting was happening, not enough information was available to explain why.

11.3.3 Inability to Visualization Multiple High-level Variants

As discovered in usecase 1, the visualisation is not capable of visualization a large number of High-level variants simultaneously. Due to the amount of detail in the visualization and the average length of High-level variants, it is an incredibly tedious task to compare visualizations. Based on the experience of working with the visualizations, when focusing on just 1 logistic step, the most

that can be compared reasonably is around 30. And when looking at the whole system, maybe 3 together. Thus when there are several hundred High-level variants that have to be compared, this becomes a problem.

11.4 Problems Encountered

11.4.1 Memory

The largest problem encountered when implementing was a lack of main memory, which led to batch file processing and slow run-times. As mentioned at the beginning of this section, the machine used was a Windows 64-bit operating system laptop with an Intel i-5 CPU and only 16gb ram. This meant loading, even just 1 day of data, and working with it in the main memory was not possible. Thus everything had to be done reading and writing from files. Now, this had 2 main negative effects: slower run-time and increased complexity.

Reading and writing files had a negative impact on run-time since the project was implemented in Python. Python was chosen since it is excellent for very quick and efficient prototyping which led to rapid development of the project. Since a large part of this project was exploratory, it seemed appropriate. However, it was discovered that frequently reading and writing from files is a run-time bottleneck for Python. Since reading and writing files were necessary due to memory constraints, the run-time efficiency of the whole pipeline suffered greatly.

Further, batch processing of files introduced a layer of complexity to the project. Everything had to be done in a scalable, and sometimes unintuitive, manner. This lead to slower development and thus a slower rate of progress. This also compounded the fact that bugs became an issue, which is elaborated on in the next section.

11.4.2 Bugs

Through out this project many kinds of bugs were encountered. These included:

- Syntax errors (were not picked up by the system because they were legitimate code, but absolutely not the intention)
- Accidental hard-coding
- Wrong logic
- Incorrect assumptions
  - Strange edge cases
  - Or simply, assumption did not hold

Most of these bugs were caused by human errors - arguably due to the size of the project. But the last one listed was a combination of not having enough domain knowledge and being unable to easily verify assumptions. Both of these aspects improved with time, which seems to imply that in some sense, it was unavoidable. However, they still caused delays and confusion.

Regardless, this project was a very large one. Nearly 10 thousand lines of code were created, and gigabytes of files read and written. With such a large codebase, bugs seemed to be unavoidable and delays inescapable. However, everything turned out to work well in the end.

11.5 Future Work

Below we identify the different areas in the method where additional research could be done and hint towards possible solution directions.

11.5.1 Different Dimensionality Reduction

In chapter 6 PCA is mentioned as the technique used in PFA to reduce the number of dimensions. However, there are alternative algorithms for dimensionality reduction. An example of this is ZIFA (Zero Inflated Factor Analysis) which is a dimensionality reduction method designed for zero-inflated data. Although the paper that explains ZIFA [5] did so using single-cell gene data, the technique could still be used. And possibly, this alternative could result in higher quality dimensions (which in our case are locations).

11.5.2 PFA Clustering Algorithms

In Section 6.3.2 2 different clustering algorithms were discussed: namely k-means and agglomerative. However, dozens of other possible clustering algorithms could be tried. Examples include mean-shift clustering (an adaptation on k-means) or density-based clustering (such as DBSCAN). It is unclear which would be best, and thus this would be a point where plenty of additional research could be done.

11.5.3 Cluster Evaluation

In chapter 7.2 a paper [4] describing 30 different cluster validity indices was mentioned. However, only 2 of these indices were tried. Dunn's Index was chosen because it was simple, quick to implement and potentially could have been the answer we needed. Then the Silhouette Coefficient was also picked because according to that paper [4], the Silhouette Coefficient tends to perform the best on most data. However, that leaves 28 others that were not tested or carefully studied. So from such a wide selection, one of them could be studied closer and tried, leading to a possibly more accurate scoring.

Along these lines of cluster evaluation, a more extensive range of principle set sizes could also be developed. The range from 2-18 was based on domain knowledge of the system as there was not enough time to create a more robust method. But trying a larger range of sizes and using techniques like Bayesian Optimization would result in a more complete search to optimize the principle set size.

11.5.4 Different Definitions for Optimal, Sub-optimal and Non-optimal

These definitions of optimality are critical because changing these would change the distribution of fast/slow bags in each High-level step, and thus would change what clusters we identify as belonging to just slow/fast bags. Thus changing these definitions would have a significant impact. Two ideas for additional research are described below.

First, a closer look into how bags are distributed across these categories could reveal ideas for more accurate definitions. For example, if bag system times cluster around 4 different durations, then maybe 4 categories of optimality are better.

Second, these definitions could also be enriched by considering different definitions depending on key characteristics of the High-level route. For example, what we consider optimal for a bag that starts at check-in might be different than a bag that comes from transfer. Or a bag that goes to storage will certainly take longer on average versus a bag that does not.

11.5.5 Expand Visualization Features

As mentioned in the limitations, the visualization is not capable of proper root-cause analysis. More context features of the related clusters would allow the tool to further deepen the analysis. This would be an increase in the value of the insights and findings and its overall usefulness. However, this is very difficult to do properly. It is not easy to reduce so much information to just the essentials and a user-friendly fashion.

11.5.6 More Than One Day of Data

Understanding how clusters change over days of high load, low load and same load in the system is a very interesting question. Also understanding if the clusters stay stable, (or if they change, how much they change) would deepen the understanding of the technique. However, figuring out how to compare clusters across days would be the main challenge here.

11.6 Final Words

With our novel method of clustering logistic routes, we have accomplished some exciting results. However, there is plenty more to be discovered, as mentioned in the section of future work and we look forward to future developments in clustering of logistic routes.

12 List of References

References

[1] Lu, Y., Cohen, I., Zhou, X., Tian,Q. *Feature Selection Using Principle Feature Analysis*. ACM Multimedia, 29 Sept. 2007. 18, 30, 31

[2] Anna Turu Pi *Performance-aware Conformance Checking on Material Handling Systems* Master Thesis, Technical University of Eindhoven July 2019. 12

[3] Harper, G., Pickett, S.D., *Methods for mining HTS data*. Drug Discovery Today. 11 (15–16): 694–699. August 2006 viii, 2

[4] Arbelaitz, O., Ibai, G., Muguerza, J., Perez,J. M., Ingo, P. *An extensive compartive study of cluster validty indices*. University of the Basque Country, Spain, 3 Feb. 2012. 8, 36, 66

[5] Pierson, E., Yau, C. *ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis*. Genome Biology. 2015;16:241. 66

[6] Bose, R. P., van der Aalst, W. *Context Aware Trace Clustering: Towards Improving Process Mining Results*. Technical University of Eindhoven, 1 April. 2009. 1

[7] Song, M. Gunther, C.W., van der Aalst, W. *Trace Clustering in Process Mining*. Technical University of Eindhoven, 2008. 1

[8] van der Aalst, W. *Process Mining Manifesto*. F. Daniel et al. (Eds.): BPM 2011 Workshops, Part I, LNBIP 99, pp. 169–194,2012. 1, 5

[9] van der Aalst,W.M.P. *Process Mining: Data Science in Action*. Springer Berlin Heidelberg, 2016. 1

[10] Jolliffe, I., *Principal component analysis*. Springer Science and Business Media, New York. 2002. 5

[11] Jain, A.K. *Data clustering: 50 years beyond K-means*, Pattern Recognition Letters, Volume 31, Issue 8,Pages 651-666,ISSN 0167-8655. 2010. 7

[12] Müllner, D. *Modern hierarchical, agglomerative clustering algorithms*, arXiv:1109.2378v1. 2011. 7

[13] Song, M., Gunther, C.W., van der Aalst, W.M.P., *Trace Clustering in Process Mining*, In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops, LNBIP 17, pp. 109-120. Springer, Berlin 2008. 1

[14] de Leoni, M., Dundar, S., *Event-Log Abstraction using Batch Session Identification and Clustering* In Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC '20). Association for Computing Machinery, New York, NY, USA, 36–44. 2020 2

[15] *From Low-Level Events to Activities - A Pattern-based* Mannhardt F., de Leoni M., Reijers H.A., van der Aalst W.M.P., Toussaint P.J. *From Low-Level Events to Activities - A Pattern-Based Approach* In: La Rosa M., Loos P., Pastor O. (eds) Business Process Management. BPM 2016. Lecture Notes in Computer Science, vol 9850. Springer, Cham. 2016. 2

[16] Ferreira, D., Szimanski, F., Ralha, C. *Mining the Low-Level Behavior of Agents in High-Level Business Processes*. International Journal of Business Process Integration and Management. 6. 146-166. 10.1504/IJBPIM.2013.054678. 2013.

[17] Montani, S., Leonardi, G., Striani, M., Quaglini, S., Cavallini,A. *Multi-level abstraction for trace comparison and process discovery*, Expert Systems with Applications, Volume 81,2017, Pages 398-409, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2017.03.063.

[18] van Dongen B.F., Adriansyah A. *Process Mining: Fuzzy Clustering and Performance Visualization*. In: Rinderle-Ma S., Sadiq S., Leymann F. (eds) Business Process Management Workshops. BPM 2009. Lecture Notes in Business Information Processing, vol 43. Springer, Berlin, Heidelberg. 2010.

[19] Günther C.W., Rozinat A., van der Aalst W.M.P. *Activity Mining by Global Trace Segmentation*. In: Rinderle-Ma S., Sadiq S., Leymann F. (eds) Business Process Management Workshops. BPM 2009. Lecture Notes in Business Information Processing, vol 43. Springer, Berlin, Heidelberg. 2010.