Enhancing Merge Conveyor Utilization in a Line-Sorter Sortation System by Introducing a New Control Algorithm

A Case Study at Vanderlande

Revanth Sai Yayavaram

Delft University of Technology

TU Delft

VANDERLANDE

Enhancing Merge Conveyor Utilization in a Line-Sorter Sortation System by Introducing a New Control Algorithm

A Case Study at Vanderlande

by

Revanth Sai Yayavaram

Master Thesis

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department of Maritime and Transport Technology of the Faculty of Mechanical, Maritime and Materials Engineering of Delft University of Technology
To be defended on Wednesday, August 30, 2023, at 13:00 PM

Student Number : 5302307
MSc Track : Multi Machine Engineering
Report Number : 2023.MME.8848

Supervisor:	Dr Ir. Yusong Pang	
Thesis Committee:	Dr Jovana Jovanova,	TU Delft Committee Chair, Faculty of 3Me
	Dr Alessia Napolano,	TU Delft Committee Member, Faculty of 3Me
	Ir. Jaap Schouten,	Company Supervisor, Vanderlande
Date:	August 30, 2023	

An electronic version of this thesis is available at https://repository.tudelft.nl/

TU Delft

Preface

This thesis marks the completion of my journey as a master's student in Mechanical Engineering (Multi-machine track) at Delft University of Technology. My time in this program has been nothing short of a roller coaster ride, filled with highs and lows. Initially, the disappointment of not being able to come to Delft in 2020 due to the COVID-19 pandemic was disheartening. However, by reapplying for the following year, I was fortunate enough to join TU Delft. Reflecting on these two years, I take great pride in knowing I have successfully navigated through this roller coaster and emerged stronger.

I would like to express my sincere gratitude to the individuals who have been instrumental in the completion of this thesis. First and foremost, I am deeply thankful to my supervisor, Dr Yusong Pang. His invaluable feedback has always challenged me to surpass my limits and deliver my best work. Each meeting with him was an intense and enlightening experience that helped me gain knowledge and clarity about my research goals. Thanks to Dr Jovana Jovonava for accepting to be my Chair for this project.

I extend my appreciation to Jeroen Vennegoor op Nijhuis and Jaap Schouten for selecting me as a graduate intern at Vanderlande. Our regular meetings were always insightful and enjoyable. I am grateful for their continuous support and guidance, both personally and professionally. I am grateful to the entire OTMA-MBD team for making me feel at home and involving me in engaging lunchtime discussions.

I would like to express my heartfelt gratitude to my family and friends. Although it saddens me that my father cannot witness my graduation, I am certain that he would have been proud. I am forever indebted to my mother and sister for their unwavering belief in me and their unconditional support. Their sacrifices have brought me to this point, standing here before you, presenting this thesis. I would also like to thank Rishika, who has consistently motivated me throughout my master's phase, and Sriram, a close companion from India, for his unwavering support. Lastly, I want to express my appreciation to my close friends and cousins who were always just a phone call away when I needed someone to talk to.

As my time studying in Delft comes to an end, I am grateful for the opportunity to study at this esteemed university in a beautiful country. I am thankful for the friendships I have made and the valuable life lessons I have learned. Now, it is time for me to embark on the next phase of my life. Despite everything, I would like to sincerely thank you for taking the time to read my thesis. I hope you find it enjoyable and informative, and that it adds to your knowledge in some way.

Revanth Sai Yayavaram
Delft, August 2023

Summary

The parcel distribution industry, which encompasses prominent players like UPS, DHL, and FedEx, handles a substantial volume of parcels, often reaching millions per week. To ensure efficient operations, these companies heavily rely on automated material handling systems like conveyors and sorters. Among the critical components of these systems is the merging process, which involves combining parcels from multiple infeeds onto a main conveyor (*also called merge conveyor*) for further sorting. In their pursuit of continuous improvement, industry leaders, including Vanderlande, a renowned provider of material handling automation solutions, are actively seeking ways to enhance the throughput and utilization of their sorting systems. This research collaboration specifically addresses the challenge of low utilization of the merge conveyor in Vanderlande's line sorter sorting system.

This master's thesis aims to identify how the parcel merging process and the utilization of the merge conveyor can be improved by introducing a new control algorithm. To do so, the following research question is established:

"How can the parcel merging process and utilization of the merge conveyor in a line-sorter sortation system be improved by introducing a new control algorithm?"

To answer this research question, several sub-questions were explored. The first sub-question focused on identifying the key factors influencing the parcel merging process and utilization of the merge conveyor. Through a comprehensive examination of the current industry-level control algorithm and relevant literature, it was determined that factors such as parcel velocity profiles, announcement timing of the parcels on the infeed, merge conveyor speed, and parcel dimensions have a significant impact on both the merging process and system utilization. Achieving higher utilization while maintaining a balanced load among infeeds requires careful consideration of these factors.

The second sub-question revolved around identifying a suitable control algorithm for improving the merging process and utilization. A thorough review of the literature was conducted, encompassing control algorithms used in domains such as baggage handling systems and vehicle merging processes. Dynamic Programming (DP) emerged as a promising approach to optimize the parcel merging process. To evaluate the selected control algorithm, a set of key performance indicators (KPIs) were defined.

The third sub-question revolved around the development of the chosen control algorithm using Discrete Event Simulation (DES) implemented in Python. The model construction process involved creating a conceptual model that incorporated an imaginary segment preceding the merge conveyor. This segment, aligned with the merge conveyor axis, allowed parcels to be efficiently filled while maintaining the safety gaps between them using the control algorithm. The model encompassed three distinct processes: parcel generation, the DP-based control algorithm for assigning slices to parcels, and the delivery of parcels from the infeeds to the merge conveyor based on predetermined exit times. Rigorous verification and validation procedures were performed to ensure the reliability and accuracy of the model. Furthermore, a sensitivity analysis demonstrated that smaller segment sizes offer advantages when dealing with fly-through parcels.

The fourth sub-question involved comparing the developed control algorithm with Vanderlande's current industry-level algorithm. Through a series of scenario tests with consistent parameter settings, the KPIs obtained from both algorithms were compared to evaluate the new algorithm's superiority, as well as any potential disadvantages. Additionally, a comprehensive cost-benefit analysis was conducted to provide a holistic understanding of the algorithm's practical implications. This cost-benefit analysis has helped in answering the final sub-research question.

The collective findings indicate that the utilization of the merge conveyor can be greatly enhanced by implementing DP as the control algorithm. By efficiently filling the imaginary segment, the algorithm optimizes the allocation of slices, utilizing available parcels instead of assigning a slice to each incoming parcel individually. Throughout the thesis, the importance of balancing utilization and load distribution among the infeeds is consistently highlighted. To achieve load balance, a max heap algorithm is employed, prioritizing the infeeds according to their maximum filled queue. The combination of DP and the max heap algorithm demonstrates the potential to achieve higher utilization with proper load balancing, ultimately leading to the desired outcome of increased utilization pursued in this research.

The developed algorithm in this thesis aimed to closely emulate real-world conditions, with certain assumptions playing a crucial role. Recommendations include conducting additional experiments to evaluate the algorithm's behaviour under realistic scenarios, considering dynamic variations in infeed capacity and parcel characteristics. It is also recommended to test the algorithm on different layouts and loop sorters. Considering potential additional costs and conducting further cost-benefit analyses is essential. Furthermore, future research can explore sustainability aspects and develop a control technique adaptable to different speed variations. Finally, it can be said that his thesis serves as an initial exploration of dynamic programming as a control algorithm to enhance merge conveyor utilization for a line sorter sorting system in the field of parcel handling.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
AGVs	Automated Guided Vehicles
AMHS	Automated Material Handling System
BHS	Baggage Handling System
CAV	Connected Autonomous Vehicle
CBA	Cost Benefit Analysis
CV	Computer Vision
DES	Discrete Event Simulation
DHL	Dalsey, Hillblom and Lynn
DP	Dynamic Programming
FedEx	Federal Express Corporation
FCFS	First Come First Serve
FIFO	First In First Out
HPF	Highest Priority First
ILP	Integer Linear Programming
KPI	Key Performance Index
LQF	Longest Queue First
MHS	Material Handling System
MILP	Mixed Integer Linear Programming
MW	Maximum Waiting Time
OTF	On-the-Fly
PBA	Priority Based Algorithm
PEC	Photoelectric Cell
QB-IM	Query Based Intersection Management
ROI	Return on Investment
RR	Round Robin
SC	Switching Condition
TMS	Traffic Management System
UPS	United Parcel Service
VS	Via-Stop

1

Introduction

Online orders and e-commerce have experienced exponential growth, reaching unprecedented levels in the present world scenario. Today, it is common for individuals to expect their orders or parcels to be delivered within a couple of days or no more than a week. However, behind the scene, there exists a vast and complex world of Material Handling Systems (MHS) that operates to make this possible. MHS can be observed in various aspects of modern economies, such as parcel and postal services, airports handling baggage, warehouses moving pallet loads, seaports managing shipping containers, manufacturing systems transporting parts, and many other applications (Haneyah et al. 2013).

By definition, a *MHS* is a system or a combination of methods, facilities, equipment, and labour employed to work for the sole objective of moving materials from one source to another (Cross et al. 1986). MHS can be customized according to the specific needs of the industry or application, allowing for efficient handling of a wide variety of products, from small parts to heavy loads. An *Automated Material Handling System* (AMHS) is a type of material handling system that uses advanced technologies to automate the movement, storage and retrieval of materials (Agrawal et al. 2006). Among the various applications of AMHS, parcel sorting systems play a crucial role in the parcel industry. These systems employ diverse techniques and technologies to sort parcels based on different criteria, such as destination, size, weight, shape, and other characteristics. Typically, parcel sorting systems consist of a series of conveyors, sorters, scanners, and other components, which work together to sort and route parcels to their designated destinations (Pfohl et al. 2020).

The parcel distribution industry, prominently dominated by companies like UPS, DHL, and FedEx, encounters the formidable task of managing an enormous influx of parcels, varying from a few thousand to millions every week (Cardenas et al. 2017). In order to handle such volumes, these industries heavily depend on conveying systems that possess robust load-carrying capacities. Furthermore, the integration of various sensors and sorting algorithms play a crucial role in augmenting their ability to handle parcels efficiently. These facilities are responsible for processing thousands of parcels on a daily basis, requiring optimal utilization of the system for effective operations (Drissi Elbouzidi et al. 2023). The issue of reduced efficiency in the sorting systems is a widespread concern in various industries involved in parcel handling. To address this concern, the current thesis is being carried out in collaboration with Vanderlande, a leading provider of material handling automation and logistics solutions. This research aims to identify and introduce a new control algorithm that can provide higher utilization and improved throughput for the industry. By leveraging advanced control algorithms and system optimization techniques, it is possible to enhance the efficiency of sorting systems and increase

the volume of parcels handled more effectively.

1.1. Company Background

Vanderlande is a leading provider of material handling automation and logistics solutions with a global presence in over 100 countries and a workforce of more than 7,500 employees (Vanderlande.com 2023). The company is known for its cutting-edge airport solutions and market-leading baggage handling systems, which are currently in use in over 600 airports worldwide. Their automated systems are designed to improve operational efficiency, reduce costs, and enhance customer service, all while prioritizing innovation and sustainability through the use of energy-efficient solutions and sustainable materials.

One of Vanderlande's key strengths is its customer-centric approach, which includes end-to-end support throughout the life cycle of its solutions. The company has a proven track record of delivering successful projects for some of the largest airports and e-commerce companies in the world, thanks in part to their flexible and customizable sorting systems (Vanderlande 2023a). These systems incorporate advanced technology such as high-speed conveyors, robotic arms, and computer vision, making them an excellent choice for those seeking state-of-the-art equipment.

At Vanderlande, transportation and sortation primarily rely on two technologies (Vanderlande 2023b): Conveyor systems and Automated Guided Vehicles (AGVs). While conveyor systems offer higher capacity and reliability, even minor disruptions can result in significant downtime. On the other hand, AGVs have a lower capacity within a fixed infrastructure of the same size as the conveyor systems, but they offer the advantage of individual breakdowns not affecting the entire system. Higher capacity requirements for the parcel industries encourage them to utilize conveyor-based sorting systems. As customer requirements for automation become stricter, systems have become more complex and highly automated, which has created opportunities for performance improvement to increase customer satisfaction. Currently, Vanderlande is looking for new possibilities to improve the utilization of its conveyor-based parcel sorting system. It is believed that the current algorithm used in the control of the parcel traffic in the sortation system has a scope for further improvement and there are numerous options that haven't been researched yet, that could potentially yield handling more capacities. This provides an opportunity for further research that can help in developing a control algorithm which can improve the utilization of a merge conveyor in their parcel sorting system.

1.2. Research Overview

There are various potential configurations for a parcel sorting system, each tailored to the specific needs of the owning company. Several examples of parcel sorting system layouts can be found in the literature, such as those depicted in Figure 1.1, Figure 1.2, and Figure 2.1. Despite the availability of different layouts, the fundamental process remains consistent across all of them.

To begin with, parcels are initially placed on the infeed conveyors. These conveyors serve to transport the parcels along their lengths and merge them onto the main conveyor. The act of parcels from various infeeds converging into a single lane flow is referred to as merging (Gasperin et al. 2012). Consequently, the main conveyor is commonly referred to as a merge conveyor. In an autonomous environment, the timing and sequence of parcel merging are determined through information exchange between the infeed controllers and the merge controller. A comprehensive explanation of this communication process will be provided in section 2.3. The merge conveyor moves continuously in time and facilitates the

transportation of parcels from the merge zone to the sortation zone, where the parcels are sorted (Pfohl et al. 2020) at regular intervals without any interruptions. Further elaboration on the distinct zones will be presented in detail in chapter 2.

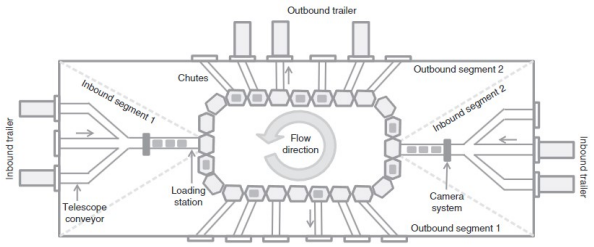


Figure 1.1: Parcel Distribution Center Layout , (Boysen et al. 2017)

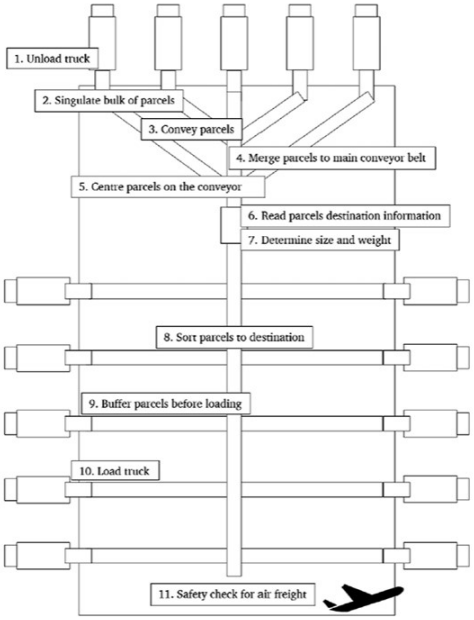


Figure 1.2: Parcel Distribution Center Layout; (Pfohl et al. 2020)

1.2.1. Research Problem

The parcel sorting system comprises multiple infeed lines through which parcels are inducted onto the merge conveyor. Parcels received from trucks are loaded onto the infeed belt by either operators or robots, with the number of infeeds, their length, and the spacing between them being variable in the entire parcel sorting system. To assess whether parcels fall within specified dimensional ranges, the infeed conveyors are equipped with Photoelectric Cells (PECs) positioned at the beginning. Once a parcel passes these initial PECs, it proceeds to another PEC. Subsequently, a request for reserving its designated space on the merge conveyor is initiated through communication between the controllers responsible for the infeed and merge conveyors. The reservation process entails several steps, which will be elaborated on in section 2.3. Eventually, parcels are transported along the length of the infeeds and inducted onto the merge conveyor based on their assigned slice and delivery time.

Once the parcels are merged, they are transported to the sortation zone, where they are either sorted or

directed for a subsequent trip if they were not sorted. In the latter case, the parcels are again guided through the merge zone. The current algorithm for reserving space to accommodate incoming parcels proves ineffective, resulting in an increased occurrence of empty spaces on the merge conveyor and a higher frequency of infeed reservations at the system's onset, leading to an imbalanced situation. This imbalance arises from the merge controller's reservation of space on a First Come First Serve (FCFS) basis. While there are techniques available to address this load imbalance, the current reservation system only achieves an approximate utilization rate of 80%, leaving ample room for improvement.

The problem of decreased utilization in parcel sorting systems is prevalent in many parcel handling industries and Distribution Centres, where thousands of parcels need to be handled efficiently every day (Drissi Elbouzidi et al. 2023). The consequences of ineffective parcel space reservations include lower utilization rates and an increased frequency of reservations, especially for the parcels on the infeeds located at the beginning of the merge conveyor, leading to load imbalances. This issue calls for identifying a new control algorithm to optimize the parcel merging process and improve system efficiency.

1.3. Research Objective and Scope

The potential enhancement of system output through increasing conveyor speed or modifying the layout is sub-optimal and is usually associated with huge costs. With multiple infeeds, the challenge is to improve space utilization while balancing the load among infeeds to enhance system throughput. A new control algorithm is required to address this scientific problem. The current control algorithm for the provided layout, Figure 4.2, requires improvement to meet the benchmark of having at least 80% utilization. While optimization techniques have demonstrated success in various fields such as Baggage Handling Systems (BHS) and Traffic Management Systems (TMS) - for instance, in highway on-ramp vehicle merging - their application in the domain of Parcel Handling remains relatively unexplored. Despite the potential benefits that these optimization techniques could offer, such as increased efficiency and throughput, they have not yet been fully explored in this area. Therefore, there may be opportunities to adapt and apply these techniques to parcel handling systems to improve performance and enhance the overall logistics process.

This study concentrates on developing a control algorithm to improve the utilization by improving the parcel merging process alongside considering the load balancing of infeeds for parcel merging in a line-sorter sortation system. The findings can be useful for similar sorting systems and applications. Since parcel sorting is impeded the most at the merge zone (Haneyah et al. 2013), this area is a focal point of this research. Additionally, since the arrival distribution of parcels affects system behaviour, this research also considers infeeds.

As the direction and relevance of this research is clear, it is furthermore necessary to set the boundaries in which the research takes place. This thesis focuses on a merge zone with multiple infeeds of a line sorter sortation system. The study will be conducted for a specific layout provided by the company. Other layout options are left out of scope for this research. A computer-based simulation model which allows for the testing of various scenarios and parameters in a controlled virtual environment will be developed. By using simulation, we can conduct experiments without the need for physical testing, which can save time and resources. Furthermore, the effect of different parameters can be easily observed for the selected control algorithm by modifying the inputs to the model. Nonetheless, the output of the research can be used for similar sorting applications or sorting systems due to their operational similarities.

1.3.1. Research Questions

Following from the research problem and research objective, the main research question that arises from this context is:

"How can the parcel merging process and utilization of the merge conveyor in a line-sorter sortation system be improved by introducing a new control algorithm?"

The following sub-questions can help in answering the main research question:

SQ1- Which factors significantly impact the parcel merging process and the utilization of the merge conveyor in a sorting system?

SQ2- What existing approaches for similar problems suggest a suitable control algorithm for the current issue of low utilization?

SQ3- How can the selected control algorithm be developed for the Line-Sorter sortation system?

SQ4- How does the selected control algorithm perform compared to the current algorithm that is being used in the industry?

SQ5- What are the implications of implementing the developed control algorithm in terms of their impact on the costs for Vanderlande?

To answer the research questions, the research approach and the methods need to be made clear.

1.4. Research Approach

A well-supported research methodology, as formulated by Peffers et al. (2014), can also be applied to research control algorithms in existing systems. In their paper, the authors outline a methodology for information systems, which can be adapted to address problems related to technology and organizations. This methodology is well-suited for the current research, which focuses on developing and introducing a new control algorithm to improve the utilization of the main conveyor of a parcel sorting system. This section illustrates how this methodology supports the research questions. A typical design research methodology consists of the following steps as represented in section 1.4.

Figure 1.3: Research Approach, inspired from Peffers et al. (2014)

1. **Problem Identification and Motivation**: This section pertains to identifying a clearly defined problem and the justification for its relevance. Upon defining the problem, it has become apparent that the parcel industry faces a significant challenge concerning the utilization of the main conveyor of the sorting systems. To cope with the escalating volume of parcels that require handling each day, there is a need to enhance the sorting systems further. However, system modifications, such as incorporating new conveyor systems with higher speeds or altering the system layout, entail undesirable costs for relatively low throughput. Therefore, there is a potential to improve the control system that manages the local parcel traffic. Since the parcel

merging process is a critical step in achieving higher utilization of the conveying system, the motivation is to enhance the control algorithm or implement a new one to accomplish higher utilization. The first research question aims to identify significant factors that potentially affect the parcel merging process and the main conveyor utilization.

2. **Definition of Objectives for the Solution**: After identifying and describing the problem, the objectives for addressing it must be defined. The present problem in this research is the low utilization of the merge conveyor, which can be resolved by developing an efficient control algorithm to improve the parcel merging process and increase utilization. Potential algorithms that have yielded promising results in similar applications must be identified to achieve this. The second research question aims to pinpoint such algorithms to identify a suitable solution for the present problem.

3. **Design, Development, and Demonstration**: This phase involves designing and developing a simulation model of the sorting system for a given layout. Additionally, the simulation model will demonstrate the use of a new control algorithm in the parcel merging process of the line-sorter sortation system. Conducting experiments using this model will determine whether this solution can enhance utilization and serve as a starting point for proving that implementing it can resolve the low utilization problem. The outcome of this step will answer the third research question.

4. **Evaluation**: This phase uses the experiments conducted in the previous step. Utilizing the key performance indicators (KPIs) such as utilization, throughput, and imbalance, an evaluation can be performed. These KPIs can help compare the developed algorithm with the current industry-level algorithm. Furthermore, the costs associated with implementing the current algorithm will be analyzed. The outcome of this step will answer the fourth and fifth research questions respectively.

5. **Communication**: As this research is carried out in partial fulfilment of the requirements for the degree of Master of Science, the communication step involves disseminating the research findings with stakeholders and individuals involved in the project and those who will work in similar fields in the future.

1.4.1. Research Methodology

The Figure 1.4 outlines the research methodology employed to address the sub-questions, culminating in the answer to the main research question. To answer the first question, the current control system of the line-sorter sortation system must be thoroughly studied. This is carried out by a combination of reviewing the literature and examining the current control system at Vanderlande. In a similar fashion, the investigation for the second research question to identify different control algorithms will be carried out. This can provide options from which the most suitable control algorithm can be adopted to solve the current problem.

To answer the third research question, a literature review will be performed to identify the appropriate simulation method. This method will then be used to simulate the parcel merging process of a line-sorter sortation system. The model will then be verified and validated. The fourth research question uses the KPIs (Throughput, Utilization and Imbalance) and performs two scenarios with several sub-cases to compare the developed control algorithm to the current control algorithm that is being used in the industry. This can help in evaluating and proving the feasibility of the control algorithm as a solution to the current problem. Finally, a cost-benefit analysis will be performed to provide the implications of implementing the developed control algorithm at Vanderlande.

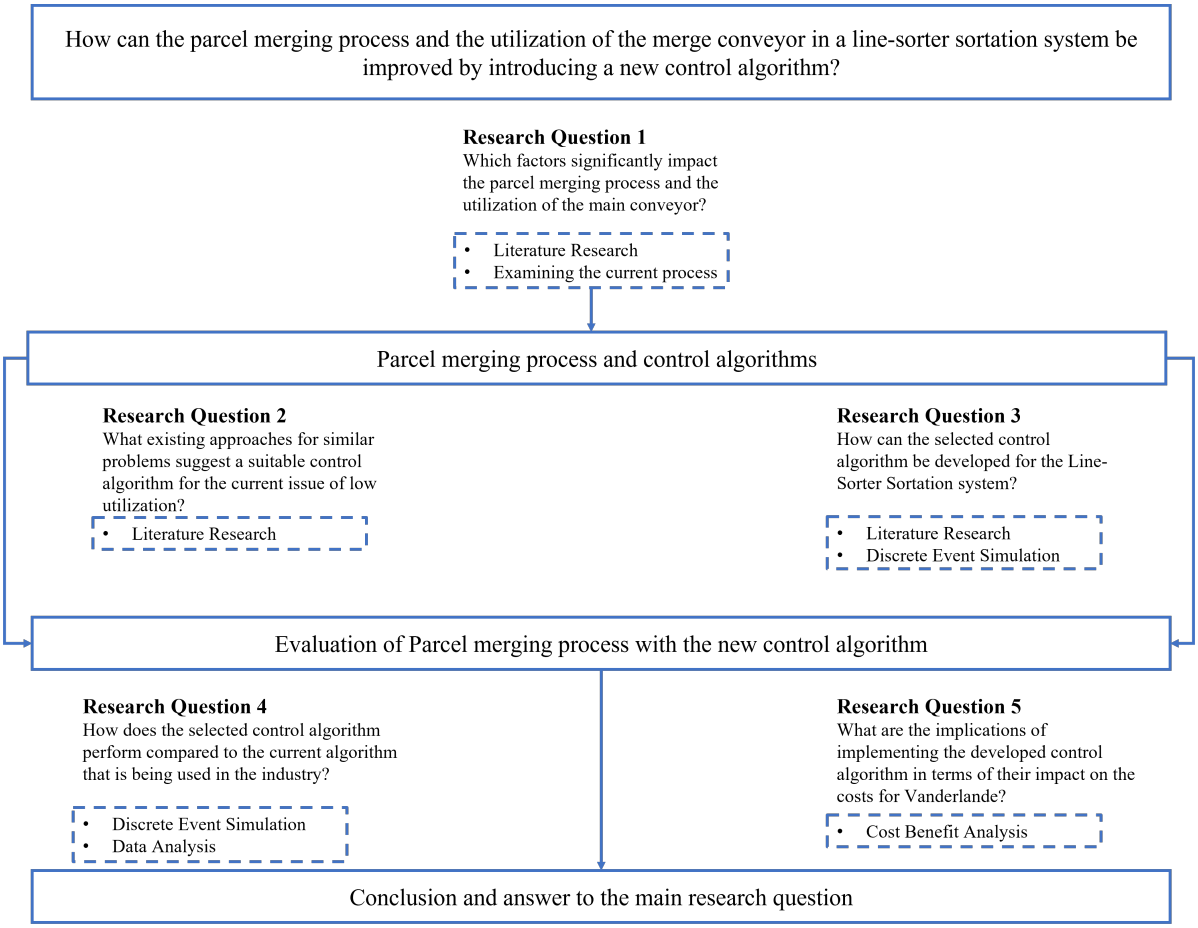


Figure 1.4: Research Flow Diagram

1.5. Structure of the report

The structure of this master thesis is organized as follows: In chapter 2, an examination of the control system in the parcel industry is conducted, with a specific focus on Vanderlande's control algorithm. The objective is to identify the factors influencing the parcel merging process and low utilization, alongside learning more about the process in which the reservations for the parcels are carried out by the control algorithm. In chapter 3, a comprehensive review of relevant literature is presented, narrowing down the focus based on the findings from the previous chapter. Subsequently, chapter 4 expands on the third chapter by detailing the development of a simulation model, corresponding to the methodology's design and development phase.

Once the simulation model is constructed and validated, chapter 5 utilizes the model to evaluate and validate the developed algorithm in comparison to the existing algorithm that is being used currently in the industry. The intention is to showcase the potential of the developed algorithm as a viable solution. Finally, chapter 6 serves as the concluding chapter of this thesis, addressing the research questions, providing recommendations, and suggesting future research possibilities after describing certain limitations based on the obtained results.

2

Parcel Sorting Systems in Practice

This chapter provides an overview of the sortation system in a parcel distribution centre. It discusses the different zones of the system, including the infeed zone, merge zone, and sortation zone. The chapter also explains important terms related to the parcel merging process and provides details about the merge zone's functionality and the communication between the infeed and merge controllers to handle the parcel merging process. Furthermore, it discusses different techniques to ensure load balancing and concludes with identifying important parameters that affect the parcel merging process. An understanding of these operational zones and the utilization of load-balancing techniques are instrumental in discerning the factors that impact the parcel merging process and the efficiency of the merge conveyor within a sorting system.

The entire sortation system is composed of several zones, each of which has a unique operation as can be seen in Figure 2.1. For instance, the *infeed zone* is responsible for accepting parcels into the system, whereas the *merge zone* combines parcels from various infeeds onto a single conveyor. In the *sortation zone*, the parcels are separated based on their destination or weight. Finally, the *overflow zone* collects any unsorted parcels and returns them to the merge zone (Ven 2022).

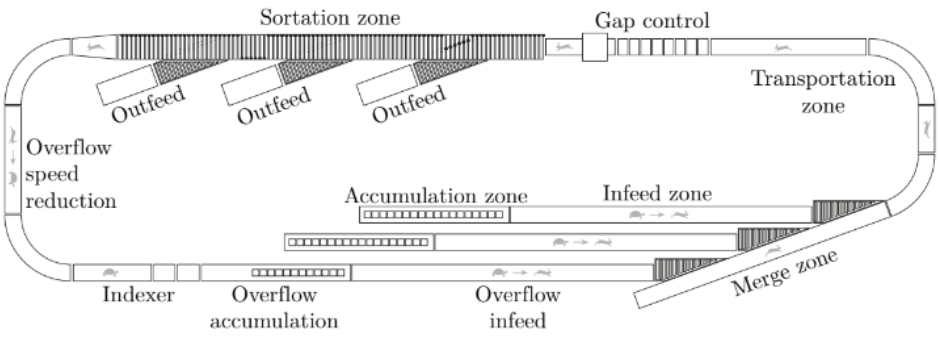


Figure 2.1: Parcel Distribution Center Layout II, (Ven 2022)

2.1. Infeed Zone

The infeed zone of the system can consist of either a single or multiple infeed lines. These infeeds can be in-line or at a certain angle to the merge conveyor. Once the assigned inbound truck occupies the designated dock door, the parcels are loaded onto the infeed conveyor in various ways such as an operator, an extendable conveyor, or even a robot. It's important to note that the infeeds are not limited to the start of the sortation system and can be located in other parts of the system as well. In the former case, an employee unloads the parcels from the truck and transfers them to the infeed conveyor if sufficient space is available Figure 2.2. This area is known as the un-loading station from the perspective of the sorting systems (Chen et al. 2023). For each incoming parcel, there are typically several consecutive steps that it goes through: parcel tracking, length and orientation measurement, parcel announcement and allocation request by the infeed controller, and acceleration of the parcel onto the main conveyor.

Figure 2.2: Parcel being placed on the infeed by an employee, (Handling 2023)

2.1.1. Infeed Parameters

In the context of any given layout, it is possible to have single or multiple belts or even roller conveyors based on the type of material moving on it, for an infeed. An example of a two-belt infeed involves the utilization of two short belts, capable of accelerating, decelerating, and transporting parcels, or stopping them at the infeed location. In a case where the infeeds are angled to the merge conveyor, there exists a junction belt that operates at a constant speed which is higher than the speed of the merge conveyor. A visual representation, as shown in Figure 2.3, illustrates the angled alignment of the infeed conveyor with the merge conveyor at an angle denoted as α. The infeed velocity is selected in such a way that the resulting speed in the direction of the merge conveyor's transport is equivalent to the merge speed. However, the speed in the perpendicular direction differs, as the merge belt does not possess any speed in that specific direction. This velocity also ensures smooth induction of parcels from the infeed onto the merge conveyor, preventing any hindrance or slipping of parcels. Consequently, the velocity can be calculated using the formula presented in Equation 2.1.

$$v_{infeed} = \frac{v_{merge}}{cos(\alpha)} \ (m/s) \tag{2.1}$$

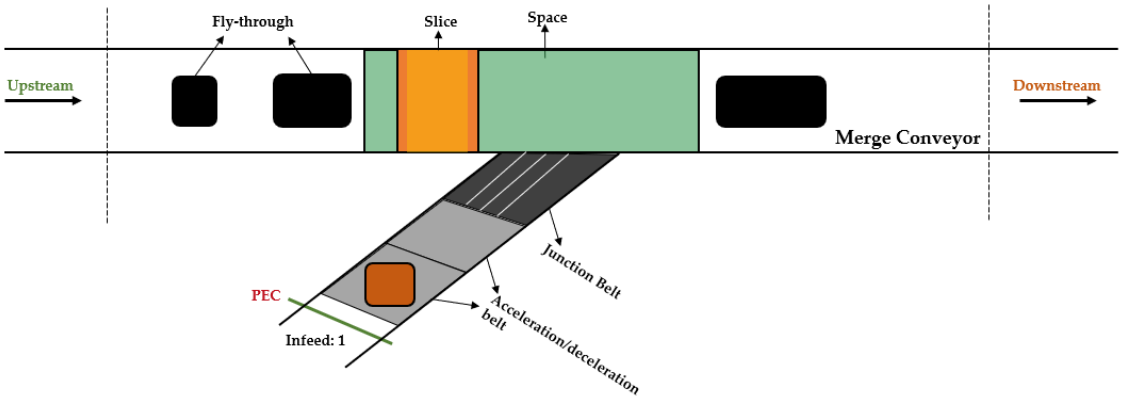


Figure 2.3: A single infeed and merge system

Currently, at Vanderlande, there are more than two parcel trajectories based on their kinematic restrictions. Kinematic restrictions refer to the specific capabilities and limitations of the parcels being transported. These restrictions can include factors such as maximum acceleration, deceleration, and velocity that the parcels can handle without experiencing issues like slipping, tipping over, or colliding with other parcels. The utilization of distinct trajectory profiles allows for the optimization of the merging operation by considering the capabilities and limitations of each parcel. By tailoring the movement parameters, such as acceleration and deceleration rates, to match the respective trajectory profiles, the merging process can be carried out smoothly and efficiently. The trajectory profiles that are relevant to the current project are given below and can be seen in Figure 2.4.

- **On-The-Fly (OTF)**: The parcel arrives with a constant arrival velocity until it is measured by the sensors. After the sensors, the parcel accelerates to the maximum velocity that it can achieve, which is the velocity at which the parcels leave the infeed. The acceleration takes place between the sensor location and the last possible point where the parcels can still accelerate. The parcels then exit the infeed. This trajectory is used for parcels that do not need to be stopped or slowed down. However, by choosing the time of acceleration, the arrival times can be adjusted for the parcels to merge with the merge conveyor.

- **Via-Stop (VS)**: The parcel arrives with a constant arrival velocity until it is measured by the sensors. However, instead of accelerating immediately after the sensors, the parcel travels with a constant arrival velocity to a location where it is decelerated. The parcel is then stopped for a minimum stop time before it is accelerated to the velocity on which the parcels leave the infeed. This stop duration can be adjusted to alter the arrival time of the parcel from the infeed on the merge conveyor. After that, the parcel exits the infeed. This trajectory is used for parcels that need to be stopped or slowed down before they can be merged.

The T_1, T_2, T_3 delivery profiles for each parcel are dependent on the length of the parcel, belt lengths, merge speed and acceleration capabilities of the belt. This is because the parcel first needs to be fully measured by the sensors on the infeed before it can start its trajectory. The delivery profiles take into account the time it takes for the parcel to accelerate and reach the maximum velocity allowed by the system, as well as the distance it needs to travel on the infeed before it can be inducted onto the merge conveyor.

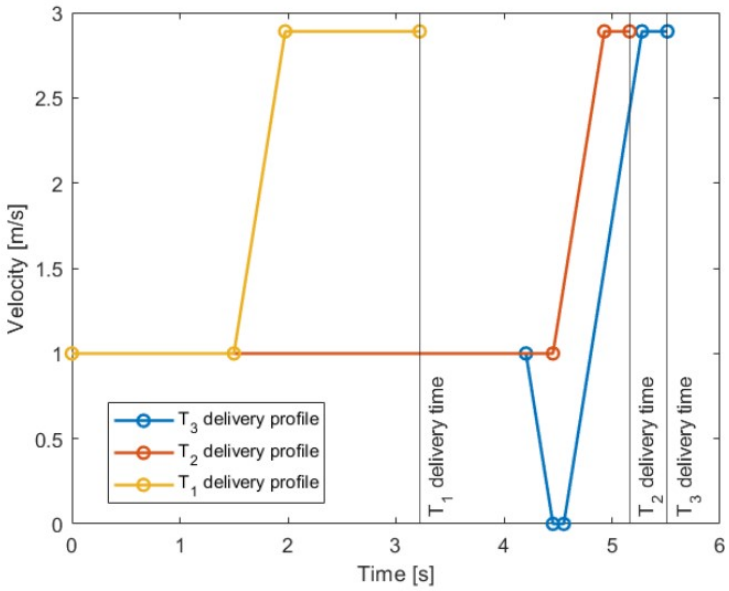


Figure 2.4: Trajectory Profile, (Bals 2021)

Before delving into the complexities of the merge zone and the process of parcel merging, it is crucial to bear in mind specific terminology that will be frequently employed in the subsequent sections.

2.2. Important terms related to parcel merging process

The planning and control of the parcel merging process are very complex. Figure 2.5 depicts a situation with one infeed and fly-through parcels travelling downstream in the system. In an autonomous environment, the arrival of a parcel is detected by using a sensor called a *Photoelectric cell* (PEC). This sensor helps in determining the length and position of the parcel.

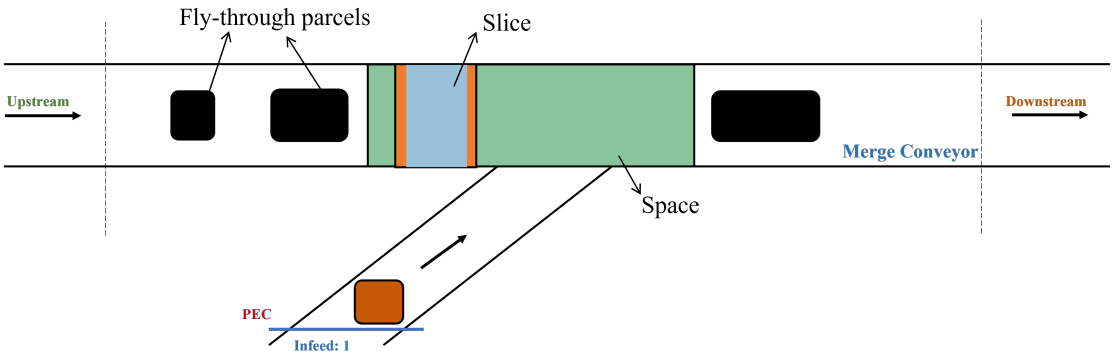


Figure 2.5: Single infeed and merge system

According Haneyah et al. (2013),

- **Downstream and Upstream**: The direction of the flow of material is referred to as downstream while the opposite flow direction is called upstream as can be seen in Figure 2.5.

- **Fly-through parcels**: These refer to parcels that were not sorted in the sortation zone before and are reintroduced into the system at a later stage for the purpose of sorting. Furthermore, fly-through parcels can also include parcels originating from various other sections of the system. Since these parcels arrive on the merge conveyor directly, it is not possible to alter their position on the conveyor.
- **Transport length**: Parcels commonly exhibit misalignment or lack of parallelism with the axis of the infeed conveyor, leading to disparities between their original lengths and the measured lengths. Transport length denotes the dimension of the parcel in the direction of the conveyor's flow, as measured by the PEC. Depending on the orientation of the parcel, the transport length can vary from at least the parcel's length to a maximum corresponding to the length of its diagonal.
- **Trailing & Leading gap**: These gaps are an addition to the transport length of the parcel, necessary to avoid parcel overlapping or collisions. Furthermore, it is also necessary for other processes that happen further downstream in the system such as weighing the parcel or scanning the parcel barcode.
- **Space**: This is the empty area available in between the slices.
- **Slice**: It is a reserved space on the merge conveyor that is based on the transport length of the parcel along with the trailing and leading gap.
- **Maximum Head Position**: This is the minimum time and the downstream position at which the infeed can deliver the parcel to the merge.

2.3. Merge Zone

The merge zone denotes the specific region where the merge operation takes place. Within this zone, the merge controller carries the responsibility of determining the appropriate timing for each infeed to release a parcel. As the number of infeeds increases, the algorithm used to control the merge zone can become more complex. When considering a single infeed and a merge conveyor in the merge zone, as depicted in Figure 2.5, the location of the parcel (1,1) - corresponding to parcel 1 of infeed 1 - on the merge conveyor is determined based on available space and time of induction. Typically, this merging process is controlled by the communication between two separate controllers: an infeed controller and a merge controller. At Vanderlande, the communication between the controllers is as discussed below:

1. Once the parcel passes the PEC located at the beginning of every infeed, the infeed controller announces the arrival of the parcel to the merge controller and requests an allocation of space for the announced parcel on the merge conveyor. This announcement consists of the possible delivery times to the merge conveyor which is calculated using the possible speed profiles (subsection 2.1.1), and the dimension of the parcel.

2. The merge controller has an algorithm called the search algorithm. Each time an allocation request is raised by an infeed, the search algorithm is executed. For every execution, the search algorithm will do the following:

   (a) Get the list of available empty spaces on the merge conveyor.

   (b) Determine the maximum and minimum head position of the parcel.

   (c) Determine the length of the slice that needs to be allocated. This length is completely dependent on the parcel dimensions.

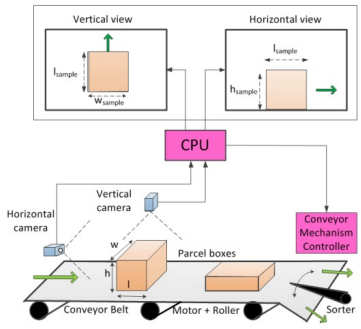
   (d) Place the allocated slice in the downstream space of the merge conveyor. This space must be sufficient not only to accommodate the parcel but also the leading and trailing gaps associated with it. The leading and trailing gaps are essential to ensure that parcels do not collide or overlap during the merge process, which can result in jamming and damage to the parcels.

(e) The allocated slice is placed such that the resultant flow is balanced among all the infeeds.

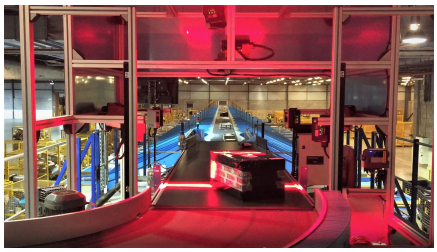
3. Based on the position of the slice, a delivery time is communicated to the infeed controller. This is also referred to as the confirmed time. The infeed then performs the delivery profile that ensures that the parcels are delivered at the confirmed time.

The search algorithm used in this case is comparable to a linear search, also known as sequential search with a complexity of orders of n, leading to increased search time as parcel size increases, as noted by Gibney 2022. It involves searching each element in an array to locate the target element, which in this context refers to space.

Upon exiting the infeed and entering the main conveyor, the parcels undergo orientation and centring through the utilization of a product turner. Subsequently, they proceed to an area where they are scanned from multiple sides to verify their dimensions and barcode information, while their weight is measured while on the conveyor Figure 2.6b. Notably, this scanning process typically occurs without halting the conveyor's motion. One approach to calculating the volume, as discussed by Yunardi et al. (2016), involves the implementation of a contour-based object identification technique. In their study, a Computer Vision (CV) system was designed, incorporating two webcams as illustrated in Figure 2.6a, to acquire dimensional information of the parcels by analyzing the captured 2D pixels. These pixel measurements serve as inputs to a multiplication program, enabling the calculation of the parcel's volume by multiplying its three dimensions.



(a) Hardware schematic and illustration for volume calculation, (Yunardi et al. 2016)



(b) Weighing and scanning, (Mothership 2015)

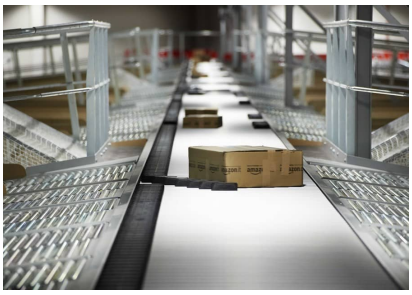
Figure 2.6: Scanning and Weighing

2.4. Sortation Zone

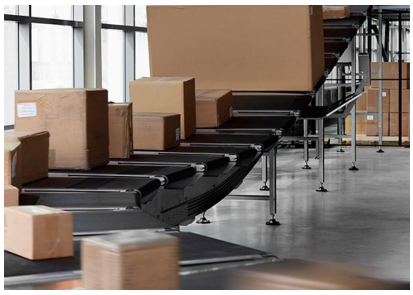
This phase entails the actual sorting of parcels. Once the merge controller assigns a slice for the incoming parcel, the parcel is inducted by the infeed and is transported to the sortation zone. During this sorting process, various types of sorters can be employed, tailored to meet the specific requirements of the company. In the domain of sortation systems, these systems are commonly referred to by the name of the sorters themselves, as these sorters are considered the central component of the entire sortation system (Landschützer et al. 2012). The sorters used in these systems can be broadly classified into two main types: Line sorters and Loop sorters (McGuire 2009). A line sorter is arranged in a linear fashion, encompassing a defined starting point (merge) and an endpoint (the sorter). The presence of a re-circulation loop in a line sorter system is contingent, as it is an optional element that can introduce complexity to the overall system. An illustration of a line sorter called the Posisorter, manufactured by Vanderlande, can be seen in Figure 2.7a. This sorter contains shoes, which help in directing the parcel

towards the respective outfeeds based on their designated destination.

Conversely, a loop sorter is purposefully designed to adopt a loop configuration. One example of a loop sorter system is the Vanderlande crossorter (Figure 2.7b). In a loop sorter system like the crossorter, parcels are merged onto the sorter, and the sorter's carriages continuously circulate within the loop. The ability of the parcel to be sorted on a given carriage does not impact the functioning of the loop. Among the two types of sorters, line sorters are cheaper when compared to loop sorters. However, loop sorters generally offer a higher capacity for processing and sorting items compared to line sorters. Loop sorters, due to their continuous loop configuration and optimized design, are capable of achieving higher throughput rates and handling larger volumes of parcels or items (Vanderlande 2023c).



(a) Posisorter (Line Sorter), (Vanderlande 2023a)



(b) Crossorter (Loop Sorter), (Vanderlande 2023d)

Figure 2.7: Line Sorter and Loop Sorter

2.5. Overflow Zone

In the sorting system, any parcels that remain unsorted are gathered in the overflow zone. Within this zone, these parcels undergo a deceleration process and subsequently enter the merge zone as fly-through parcels (Ven 2022). The unsorted status of parcels can occur due to several possible reasons. Firstly, it could be because the outfeed area is already full, and unable to accommodate additional parcels. Secondly, an unsorted parcel may have an unreadable barcode, which prevents the system from identifying its destination accurately. Lastly, the parcels are too close to be sorted individually among many more.

The overflow zone serves as a temporary holding area for unsorted parcels, allowing the system to manage the flow of parcels efficiently. By slowing down and accumulating these unsorted parcels, the system maintains order and prevents congestion in the sorting process. Ultimately, in the merge zone, these parcels are reintegrated into the sorting operation, where further attempts can be made to properly identify and allocate their destinations.

The merge zone, among the zones mentioned earlier, holds significant importance in this research due to the interaction of parcels originating from various infeeds and merging onto the main conveyor. The efficient management of the merge zone during the merging process is crucial as any disruptions in the parcel flow can have a substantial impact on the overall system throughput. Ensuring smooth operations and unhindered movement of parcels within the merge zone is vital to maintain optimal system performance. The next section corresponds to understanding more about the issues in the merge zone.

2.6. Problem Diagnosis

The complexity of the parcel merging process intensifies as the number of infeeds increases, as demonstrated in Figure 2.8 and Figure 2.9. The latter figure illustrates a scenario where parcels from multiple infeeds are merging onto the merge conveyor. In cases where there is no balancing mechanism in place, the upstream infeeds tend to deliver a larger number of parcels to the merge conveyor compared to the downstream infeeds. Consequently, this leads to longer waiting times for parcels in the downstream infeeds and creates an imbalance in the load distribution among the infeeds.

Existing studies have primarily focused on improving load balance among the infeeds. However, an issue persists regarding the suboptimal utilization of space on the main conveyor, as depicted in Figure 2.9 with red arrows. The current algorithm results in inefficient spacing between allocated parcels, which leaves empty spaces that could have been utilized more effectively. By allocating parcels slightly closer to each other without neglecting the minimum gap requirements, it becomes possible to merge the parcels from the downstream infeeds with the preceding ones, thereby reducing empty spaces and enhancing system efficiency. This optimization approach has the potential to increase the number of parcels processed within a specified timeframe, resulting in improved overall system performance.

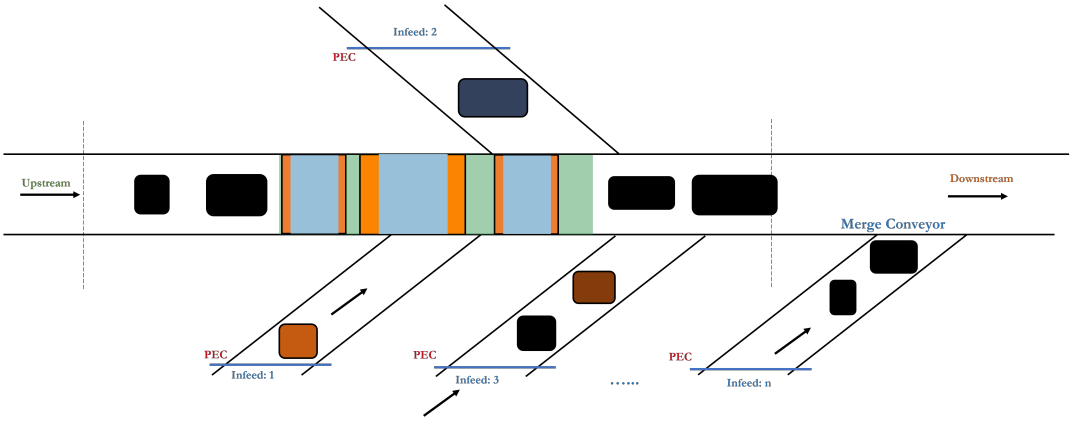


Figure 2.8: Merge Conveyor with *n* infeeds before merging

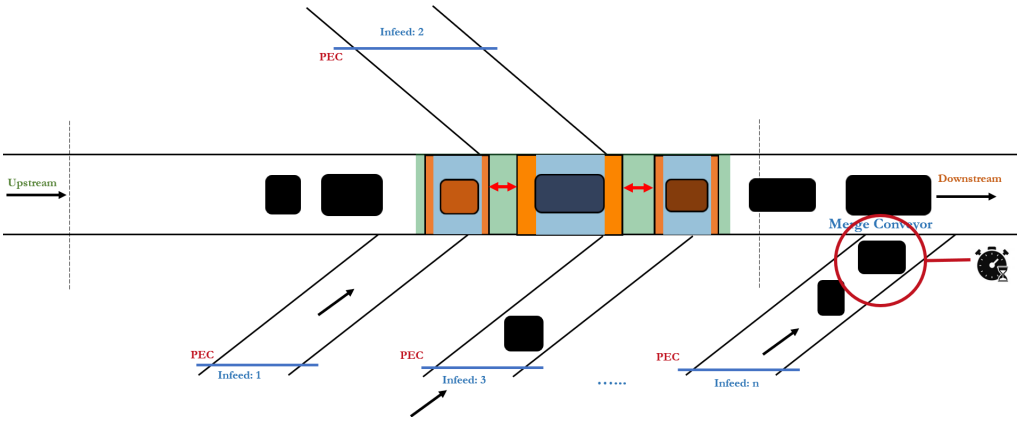


Figure 2.9: Merge Conveyor with *n* infeeds after merging

2.7. Load Balancing Techniques

As mentioned in the section 2.6, there are several techniques which are commonly practised in the industries to balance the load among different infeeds.

2.7.1. First Come First Serve (FCFS) Algorithm

The FCFS control algorithm is a commonly used algorithm in many industries, which searches for the first available empty space on the merge conveyor for parcel induction (Bals 2021). If no space is found in the given time window, the parcel is held for a certain period (Via-Stop profile) and is inducted once there is an assigned slice for it on the merge conveyor. This algorithm can lead to a high parcel throughput, but it often results in downstream infeeds being starved, leading to an imbalance.

2.7.2. Round Robin (RR) Algorithm

The Round Robin (RR) algorithm is a control algorithm that is effective in addressing imbalances in parcel delivery. This algorithm uses a pre-reserving strategy where infeeds are supplied with equal-sized windows on which they can place parcels (Meens 2017). The merge conveyor is divided into windows of equal length of the maximum parcel size, which are alternately allocated to the infeeds. An infeed may only induct parcels on spaces which are pre-allocated for that specific infeed. However, this leads to a reduction in utilization due to the reservation windows having the size of the largest parcel. In addition, there is also a possibility to have an empty window if the infeed cannot deliver a parcel for the assigned window.

2.7.3. Estimated Merge Algorithm

In the parcel merging process, the achievable distribution of each infeed is calculated based on its input parcel flow. If an infeed cannot deliver enough parcels to occupy its allocated space, there will be unused space on the merge conveyor. The Estimated Merge algorithm distributes this unused space to other infeeds proportionally to their achievable distribution.

For illustrative purposes, let us consider a scenario involving four infeeds filling up a merge conveyor, as shown in Figure 2.9. In a balanced scenario, each infeed can contribute equally to filling the merge conveyor. For the sake of explanation, let us assume that the available space on the merge conveyor is limited to 90% (removing 10% for fly-through parcels), thereby restricting each infeed to contribute only 22.5%. It is worth noting that the infeeds located upstream generally have a more consistent distribution than those located downstream. To facilitate understanding, suppose that infeed 1 and infeed 2 located upstream can contribute 30% towards filling the merge conveyor, while the remaining two infeeds can contribute 25% and 15%, respectively, in the absence of fly-through parcels. In this scenario, the fourth infeed, which can contribute up to 22.5%, can only achieve a distribution of 15%. As a result, there is an unused space of 7.5% that can be allocated among the infeeds. A summary of the distribution can be found in the table below.

Table 2.1: Estimated Merge calculation

	Available = 90%				
Infeed	Scaled w.r.t. available space	Achievable	Overcapacity	Unused Space	Allowed Distribution
1	22.5	30	7.5		25.7
2	22.5	30	7.5		25.7
3	22.5	25	7.5		23.6
4	22.5	15		7.5	15
Total	90		17.5	7.5	90

To calculate the allowed distribution the following formula is employed:

$$\begin{aligned} \text{Allowed Distribution} &= \text{Scaled Available space} + \frac{\text{Overcapacity}}{\text{Total Overcapacity}} * \text{Unused Space} \\ &= 22.5 + \frac{7.5}{17.5} * 7.5 = 25.7\% \end{aligned} \tag{2.2}$$

Each parcel received through the infeed is assigned a specific follow-up time, which serves as the minimum duration that must elapse before the subsequent parcel can be dispatched from the same infeed. This follow-up time establishes a predefined interval between the delivery of successive parcels from the infeed to the merge conveyor. By taking into account the average parcel slice, the calculated allowed distribution is transformed into the corresponding follow-up time, which is subsequently utilized by the search allocation algorithm of the merge. In essence, the infeeds left with greater allowed distribution are now capable of delivering a greater number of parcels compared to their typical capacity.

2.7.4. Early Announcement of Parcels

Apart from the search algorithms discussed earlier, there is another option to create an equal reservation point for the parcels on all the infeeds as suggested by Haneyah et al. (2013). This can be achieved by relocating the infeed PECs further upstream of an infeed for the downstream infeeds as can be seen in Figure 2.10. By implementing an equal reservation point, each infeed has the opportunity to reserve space on the merge conveyor within the same time frame, regardless of their physical placement. This means that all infeeds can make reservations at equal time intervals on the merge conveyor, even if they are located at different positions.

To establish this reservation point, it is crucial to know the sizes of parcels arriving at the downstream infeeds earlier than in the current situation. If the knowledge of parcel sizes from the upstream infeeds is delayed, there won't be enough time to properly allocate the parcels onto the merge conveyor. To achieve an equal reservation point, the positions of the infeed PECs are adjusted relative to the PEC of the furthest upstream infeed. This ensures that all parcels have the same timeframe available for reserving space on the merge conveyor. By synchronizing the timing of reservations, each infeed has an equal opportunity to allocate parcels onto the merge conveyor efficiently. A study by Bals (2021) proved that this way of early announcement of parcels can help in achieving higher balance and throughput.

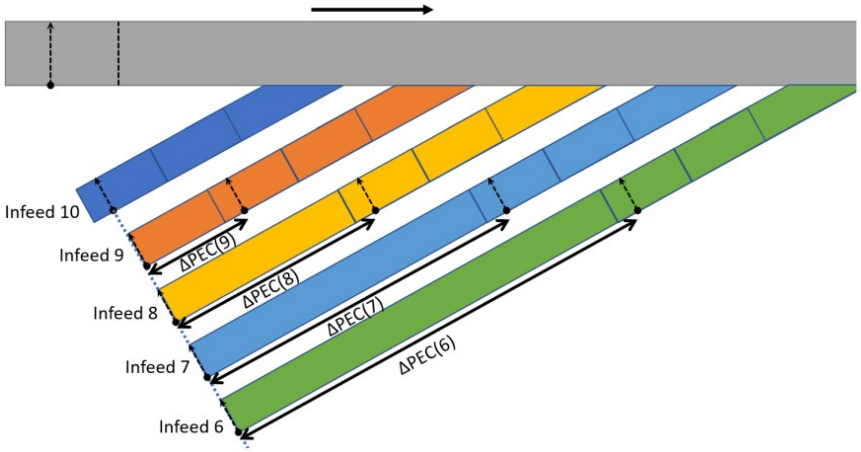


Figure 2.10: Schematic representation of the PEC location on the infeed, (Bals 2021)

2.8. Buddy Search

The Buddy Search feature in a sorting system serves to minimize unused space between parcels, thereby enhancing system utilization. This functionality is specifically applicable in Estimated Merge mode, as detailed in Section subsection 2.7.3, where parcels are assigned desired times based on permissible distribution. However, when all parcels are allocated according to their desired times, gaps can emerge, resulting in unutilized areas. To address this concern, the Buddy Search algorithm slightly deviates from the desired times while adhering to temporal constraints, effectively closing the gaps and forming contiguous ”trains” of parcels on the merge conveyor. This proactive approach significantly improves system utilization.

In practical implementation, the Buddy Search algorithm is activated once the sorting system attains a utilization level of 70% and deactivated when utilization drops to 50%, thereby avoiding undesirable oscillatory behavior around the 70% utilization threshold. Parcel allocation on the merge conveyor is accomplished by strategically placing them in close proximity to existing parcels or reservations, rather than relying solely on the first available location. Commencing from the most upstream infeed, a specific location on the merge conveyor is reserved, and subsequent parcels from various infeeds are placed directly before or after the preceding ”leading” space. Consequently, the formation of extensive parcel trains devoid of vacant intervals ensues, significantly augmenting throughput. Furthermore, compared to the conventional First-Come-First-Served (FCFS) algorithm, the Buddy Search algorithm minimizes unoccupied space between parcels and allows for additional downstream free space on the merge conveyor, thereby enabling other infeeds to reserve space more efficiently. This leads to superior load balancing within the system.

2.9. Conclusion

The first chapter of this research study put forth a series of sub-questions aimed at answering the main research question. This chapter focuses on addressing the first sub-question, which concerns the primary challenges associated with parcel merging and utilization in a parcel sorting system.

The process of parcel merging is a critical step in the sorting process. The velocity profiles of parcels play a significant role in ensuring smooth transportation and efficient merging within the system, as

discussed in subsection 2.1.1. It is essential for the parcels to be compatible in terms of size and shape to ensure smooth transportation and fit within the system. However, variable parcel dimensions can lead to uneven spacing and pose challenges for slice allocation. The timing of parcel reservations and announcements is also critical, as the merge controller must only search for available space after the infeed controller announces a parcel's arrival. The speed of the merge operation is another factor to consider, as slower processing can lead to congestion on the infeed conveyors. Nevertheless, increasing the speed is not always feasible due to the possibility of parcel slip and other kinematic constraints.

There is a trade-off between achieving higher utilization of the merge conveyor and obtaining better load balancing among the infeeds. While it may be possible to disregard balancing and achieve higher utilization, such an approach is typically not acceptable. Finally, the merge controller's search algorithm relies on a linear search with a complexity of orders of n, leading to increased search time with an increase in the number of parcels waiting to be merged. It is important to note that although this increase in search time exists, it is not readily noticeable as it operates on a millisecond scale. These challenges must be thoroughly considered and addressed to ensure a successful merge operation in a parcel sorting system.

3

Selection of Control Algorithm

This chapter provides a comprehensive literature study to find suitable control algorithm for merge zones in parcel industries or warehouse distribution centres. The chapter begins by highlighting the significance of optimal control in enhancing sorting system efficiency and discusses the trade-offs associated with increasing belt speed and layout changes. It further explores the limited existing literature in this domain and identifies the need for further investigation. The chapter then delves into the selection and classification of control algorithms, considering factors such as implementation method, and execution behaviour. section 3.1 focuses on the optimization of parcel merging in a sorting system, discussing an Integer Linear Programming (ILP) approach and a Priority Based Algorithm (PBA) for load balancing. section 3.2 investigates window re-allocation techniques to reduce imbalance among infeeds, referring to merge allocation rules and a reallocation algorithm proposed by previous researchers. section 3.3 explores the algorithms used in the vehicle merging process and their potential in improving system performance.

Optimal control of merge zones in parcel industries is a pivotal factor that greatly impacts the overall efficiency of sorting systems. It should be noted that while increasing the conveyor speed can enhance throughput, it may also lead to unintended consequences such as slips. Similarly, layout changes offer opportunities for improvement but often incur additional costs by increasing the system's footprint. Although the notion of trading off the benefits gained from layout changes against the associated costs is subject to debate, it can provide valuable insights into the cost-benefit analysis of such modifications (Fedtke et al. 2014). In addition, greater flexibility in layout and building design is achieved when the footprint is smaller. By optimizing the process of parcel merging, it is possible to enhance system throughput, reduce waiting times, and minimize wasted space. These improvements directly translate into heightened productivity and cost savings (Bals 2021). However, the existing literature in this domain is relatively limited, and previous attempts to enhance control algorithms have yet to yield optimal solutions, highlighting the need for further investigation. Notably, certain studies conducted by researchers such as Peeters (2015), Meens (2017), and Hoven (2019) have provided valuable insights into modifying control strategies. Nevertheless, additional strategies suggested in the literature (Haneyah et al. 2013; G. Kim et al. 2017) have shown potential and, in some cases, outperformed existing techniques.

The presence of multiple infeeds in the merge zone gives rise to the issue of high imbalance measures. This problem arises because the upstream infeeds tend to prioritize reserving space for parcels on the merge, resulting in increased waiting times for downstream infeed belts, as mentioned in section 2.6.

Several load balancing techniques were discussed in the section 2.7. To address the problem of improper load balancing, Ramamritham et al. (1994) classified the scheduling algorithms into four paradigms:

(I.) **Static Table-Driven Approach**: In order to carry out a series of predictable tasks, these methods produce static schedules (offline). Task allocation during execution is done using the resulting schedule, which is frequently displayed as a table.

(II.) **Static priority driven preemptive approach**: The tasks are prioritised first which means that the high-priority tasks are handled first followed by the low-priority ones.

(III.) **Dynamic Planning-based approach**: Here the tasks arrive during the execution. So, the scheduling task keeps changing concerning time. Nonetheless, the previously created tasks are executed but not eliminated.

(IV.) **Dynamic best effort approach**: This approach aims to meet the tasks' deadlines and does not promise that all the tasks are finished.

In addition to these algorithms, several queuing techniques like First-In-First-Out (FIFO), Longest Queue First (LQF), Highest Priority First (HPF), random, and Round Robin (RR) have been studied by several researchers (Jing et al. 1998; Peeters 2015). A detailed explanation of these techniques is mentioned below (Stewart 2009):

- **First-In-First-Out**: FIFO (First-In-First-Out) is a queuing technique where the first item that enters the queue is the first one to be served or processed. This technique is also known as the "first-come, first-served" method. In a FIFO queue, items are added to the end of the queue and removed from the front.
- **Longest Queue First**: Longest queue first, also known as the "largest queue first" or "workload balancing" technique, prioritizes the queue with the most number of items waiting in it. This technique is often used in situations where the service times for each item are relatively short, and the goal is to reduce the total waiting time for all items.
- **Highest Priority First**: Highest Priority First is a queuing technique that assigns priority levels to different items in the queue, with the highest-priority items being served first. This technique is often used in situations where some items are more important or urgent than others, and it is necessary to ensure that they are processed quickly.
- **Round Robin**: Round Robin is a queuing technique that allocates a fixed amount of time for each item in the queue to be processed. If an item is not processed within its allocated time, it is moved to the back of the queue and the next item is processed. This technique is often used in situations where there are many items in the queue, and it is necessary to ensure that each item receives some processing time. Round Robin is also useful for preventing any single item from monopolizing processing resources.

Often researchers draw parallels to job-shop machine scheduling problems, where the entities to be merged represent the jobs to be processed by a machine. To effectively manage the scheduling process, researchers have developed several algorithms that offer control and decision-making capabilities. The careful selection of an algorithm significantly influences the ability to achieve an optimal solution (Cormen et al. 2009). Therefore, it becomes essential to explore the various ways algorithms can be classified, as these classifications provide valuable insights into their characteristics and applications. One approach involves categorizing algorithms based on their implementation method, while another considers their execution behaviour, including series, parallel, and distributed ways of

execution. Furthermore, algorithms can be classified as deterministic or non-deterministic, among other possibilities. By understanding these classification approaches a comprehensive understanding of algorithmic properties can be gained and one can make informed choices in selecting the most suitable algorithms.

3.1. Optimization of parcel merging in a parcel sorting system

Haneyah et al. (2013) developed an Integer Linear Programming (ILP) approach for a parcel merging process that employed tilt-trays (Figure 3.1) to transport parcels. The goal of this research was to aim at maximizing utilization and minimising the imbalance in waiting times. Initially, an exact static branch and bound optimization technique was developed. With the objective of minimizing both the total and average waiting times, the approach utilizes a weighted sum to minimize an imbalance variable. The infeed with the highest value of the imbalance variable indicates the most significant imbalance within the system. The results showed that no empty trays were present and the workload is balanced. However, the experimentation proved that the time taken to obtain an exact solution is higher and hence cannot be applicable to a real-world scenario. To overcome this issue, they have presented a dynamic space allocation approach. By utilizing a dynamic allocation approach, this exact algorithm guarantees an optimal solution in any given situation when compared with the exact formulation. In the context of dynamic space allocation, the researchers implemented a Priority Based Algorithm (PBA) to prioritize and ensure a balanced flow from all infeeds, without exhibiting bias towards a single infeed. Parcels from infeeds with significantly longer waiting times compared to other infeeds are accorded higher priority. As the waiting time for unallocated parcels remains uncertain, a preliminary assignment is made to the furthest downstream unallocated tray.

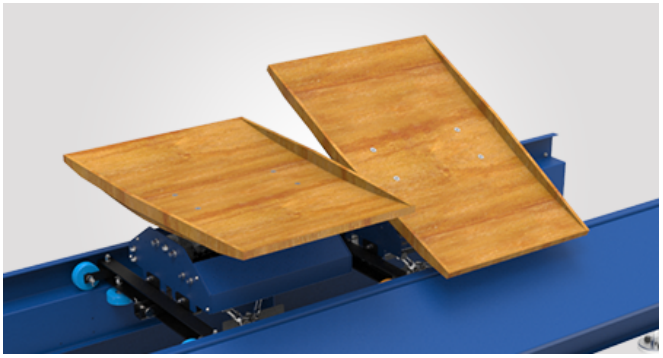


Figure 3.1: Tilt Tray, (Autotech 2023)

The dynamic assignment procedure that was developed by Haneyah et al. (2013), comprises two distinct phases: the search procedure and the reallocation procedure. During the search procedure, new parcels are assigned to unallocated trays. The reallocation procedure, on the other hand, is only initiated for parcels whose priority value surpasses a specified threshold. However, if a new parcel arrives from a high-priority infeed, the reallocation procedure takes precedence over the search procedure. Within the reallocation procedure, not only empty trays but also trays allocated to parcels with low priorities are considered.

The priority-based algorithm they put forward, is a method that gradually assigns new parcels to a specific position in a sequence. When multiple parcels are in competition, a priority function is used, as demonstrated in Equation 3.1, to determine which parcel receives the highest value. It is worth

noting that the effect of changing the value of α would be low but not zero on the overall utilization or balancing as this only plays a role in the queue procedure.

When incoming free trays become available, a group of potential parcels is identified. This is the set of candidate parcels with at most one candidate from each infeed. The length of the space window is determined based on the consecutive number of available free trays. Once the space window is equal to or larger than the largest parcel size, the potential parcels can be assigned according to their priority. If there is only one potential parcel, no priority calculation is required. In order to be eligible for assignment, the potential parcels must be able to fit within the space window, and previous parcels must have already been allocated and are expected to arrive at the relevant tray on time.

$$priority_{f,p} = \alpha * BalanceMeasure_{f,p} + (1 - \alpha) * ThroughputMeasure_{f,p} \tag{3.1}$$

- where $f \in 1, 2, \ldots, n_f$ is the set of infeeds,
- p is the sequence of numbers of the parcel on a particular infeed. For instance, parcel 1 of infeed one is denoted as $(1, 1)$ as can be seen in Figure 3.2,
- α is the weighing parameter

The researchers use the extent to which a candidate parcel can occupy the available space window on the merge conveyor as the throughput measure as described in Equation 3.2. This gives priority to the largest candidate parcel as they presume that delaying the delivery of the large parcel creates a risk of not finding another space window that can accommodate this parcel for a long time. Furthermore, the balance measure of the parcel depends on the particular infeed that transports the candidate parcel (Equation 3.3). Considering that $F' = \{f \in F \mid \exists\, p : (f, p) \in C\}$ which indicates that there exists a parcel p of the infeed f that belongs to the set of candidate parcels C.

$$ThroughputMeasure_{f,p} = \frac{l_{f,p}}{sw} \qquad \forall (f, p) \quad \in C \tag{3.2}$$

$$BalanceMeasure_{f,p} = \frac{TotW_f}{\sum_{f' \in F'} TotW_f} \qquad \forall (f, p) \quad \in C \tag{3.3}$$

- where $l_{f,p}$ is the length of the parcel p of infeed f and sw is the available space window,
- $TotW_f$ is the total waiting time of an infeed calculated as the sum of waiting times of all the parcels of that infeed.
- C is the set of Candidate parcels and sw is the space window.

The workload balancing is measured in terms of imbalance in waiting times between the infeed with maximum total waiting time and the infeed with a minimum total waiting time using the Equation 3.14.

$$ImbalanceWT = \frac{max_{f \in F}\{TotW_f\} - min_{f \in F}\{TotW_f\}}{max_{f \in F}\{TotW_f\}} * 100\% \tag{3.4}$$

In an ideal scenario, the throughput of the merge conveyor equals the sum of the maximum capacities of all the infeeds. However, in practical situations, when multiple infeeds simultaneously introduce parcels onto the merge conveyor, the downstream infeeds experience a decrease in their throughput compared to their individual maximum capacities. As more infeeds are in operation, the overall system dynamics change, leading to a reduction in the effective throughput of each individual infeed.

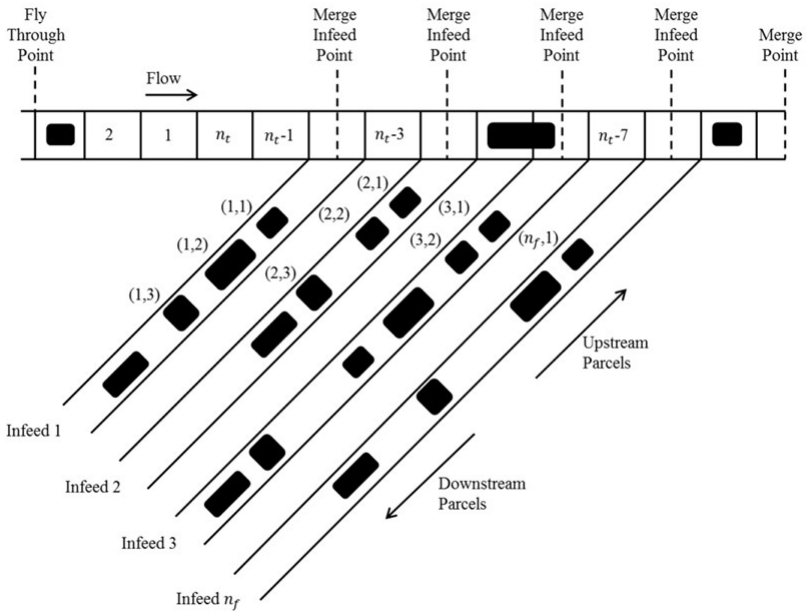


Figure 3.2: Merge area Modeling, (Haneyah et al. 2013)

It should be emphasized that the researchers consider the lengths of the parcels to range from 1 to 3 trays and calculate the inter-arrival times in terms of the number of trays so that the distance can be measured directly. The authors of (Haneyah et al. 2013), explored a merge area layout with a total of 12 possible combinations of input parameters. This was achieved by considering 3 ranges of inter-arrival times and 4 densities of fly-through parcels. They adopted a standard length of 11 trays for the infeeds in their practical implementation. Additionally, in each simulation experiment, they generated 2500 parcels on each infeed. Empirical findings demonstrate that the proposed approach successfully achieves a balance among the infeed lines. The difference in waiting times is substantially reduced from 17% to 4.2%. Moreover, the dynamic priority allocation procedure generally yields a slightly higher throughput compared to the conventional first-come-first-serve method.

3.2. Window re-allocation to reduce the imbalance among infeeds

In the pursuit of identifying new control algorithms, during the literature review different control techniques utilized in the field of BHS were also studied. Johnstone et al. (2015) discussed a set of merge allocation rules that include: (1) FIFO, (2) Feeder line priority, (3) Merge line priority, (4) Merge flush, and finally, (5) Merge timeout. Based on a set of constraints on the parcel dimensions, and spacing between the parcels, they studied a window assignment control logic system where they compared a fixed window-size algorithm to a variable window-size algorithm with a FIFO method. They concluded that the infeed's position plays a significant impact on the performance of the system and the variable length algorithm performs better when the infeed is at the desired position.

G. Kim et al. (2017), extended the research on window assignment control logic system and identified a re-allocation algorithm that tends to reallocate the assigned windows to minimize the waiting times of baggage in different infeeds. Similar to the approach of Haneyah et al. (2013), the authors of this paper distinguish between a search procedure and a reallocation procedure. Initially, incoming baggage is allocated to the earliest possible window. A list is created to keep track of eligible windows for

reallocation. Windows become eligible if they meet two conditions: the parcel has not exceeded the maximum number of window re-allocations, and the window has not been moved beyond the position of the corresponding infeed.

The reallocation procedure involves breaking the FIFO rule by switching baggage between two windows. Each possible switch is evaluated based on a benefit function.

$$SC(\ Bag(A),\ Bag(B)) = MW(C) - MW(R) \tag{3.5}$$

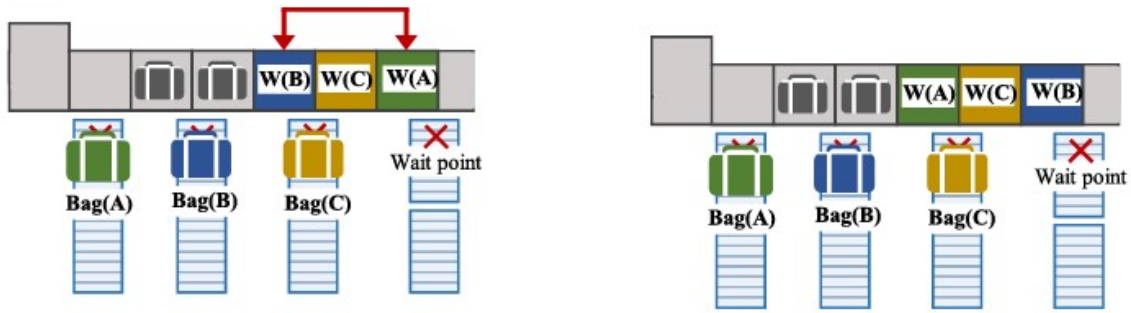
The Switching Condition (SC) for two bags is determined by evaluating the waiting times of the new and old window positions. $MW(C)$ corresponds to the maximum waiting time of the current situation, which is calculated as the ratio of the difference between the assigned window of baggage A and its location, to the conveyor velocity. On the other hand, $MW(R)$ corresponds to the maximum waiting time for the reallocation situation and is calculated as the ratio of the difference between the assigned window of baggage B and the location of baggage A, to the conveyor velocity.

$$MW(C) = \frac{Dist(W(A)) - Dist(Bag(A))}{V_{conveyor}} \tag{3.6}$$

$$MW(R) = \frac{Dist(W(B)) - Dist(Bag(A))}{V_{conveyor}} \tag{3.7}$$

This function takes into account the maximum waiting times of parcels in the current situation and the switched scenario. After calculating the benefits, the options for the reallocation process are determined by comparing the waiting times of the new and old window positions. The whole idea of reallocation is to reduce the waiting times of the baggage in the downstream flow by exchanging or swapping the windows of the parcels as depicted in Figure 3.3.

The findings demonstrate a substantial reduction in the imbalance, although complete elimination is not achieved. There is a noticeable decrease in waiting time to some extent. The level of balance varies depending on the maximum number of reallocation switches and the distribution of incoming baggage. The uniform and triangular distributions exhibit better performance with a single reallocation, whereas the exponential distribution shows improved results with two reallocations.



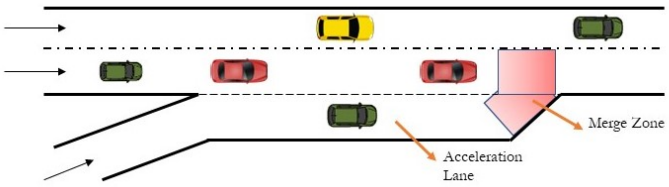
(a) Before Window - reallocation

(b) After Window - reallocation

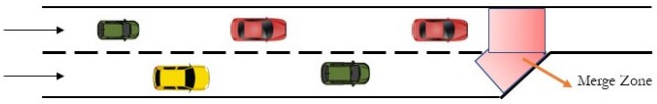
Figure 3.3: Window- reallocation algorithm, (G. Kim et al. 2017)

3.3. Optimization of vehicle merging in Traffic Management System

While parcel merging and vehicle merging involve different dynamics, it was observed that the techniques used to control the process of merging are similar. Consequently, inspiration for controlling the merging process can be drawn from traffic management systems, specifically the highway on-ramp vehicle merging and intersection management of CAVs. To reduce the challenges posed by the increasing traffic density, the transportation system has been provided paramount importance for development and has been a hot research topic for so many years now. Figure 3.4 shows different lane merge operations that usually happen in the day-to-day commute of different vehicles.



(a) Highway On-Ramp merging



(b) Lane Merging

Figure 3.4: Different merges in traffic system

Marinescu et al. (2012), proposed a merging algorithm based on their previous work on slot-based driving, which leveraged cooperation between vehicles on the main motorway and between motorway and on-ramp vehicles to achieve an efficient merging process. The algorithm aimed to maximize the utilization of road infrastructure and improve the merging manoeuvre performed by human drivers. The algorithm facilitated smooth and coordinated merging between vehicles by employing cooperative behaviour. The evaluation of the algorithm demonstrated that the proposed algorithm achieved high throughput and low delay on the on-ramp, surpassing the merging performance of the human driver model.

To identify the globally optimal passing orders of vehicles, one direct approach is to exhaustively enumerate all the possible passing orders, which, unfortunately, suffer from inevitably high computational complexity. To overcome this challenge, P. Li et al. (2017), Müller et al. (2016), Ahn et al. (2017) formulated this problem as a Mixed Integer Linear Programming (MILP), one of the exact algorithms and succeeded in providing solutions. L. Li et al. (2006) proposed an innovative solution space representation using a spanning tree, coupled with a pruning rule to search for the globally optimal passing order. These strategies, employing optimization methods and pruning rules, offer improved computational efficiency over the exhaustive enumeration approach.

Cooperative driving has emerged as a transformative approach to transportation, revolutionizing the way vehicles interact and operate on the road. Through advancements in communication technologies,

such as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) systems, vehicles have gained the ability to exchange critical information in real-time. This enables cooperative driving systems to enhance road safety, optimize traffic flow, and improve overall driving efficiency. Cooperative driving strategies can be classified into two categories (Meng et al. 2018): (1) An ”ad hoc negotiation based” which involves considering the vehicles approaching the intersection and devising short-term driving plans for them through bilateral negotiations rather than being preplanned or scheduled. This strategy aims to approximately adhere to a first-come-first-served order, allowing for some adjustments. While this approach often results in a local optimal solution in various scenarios, it lacks a globally optimal solution (Huang et al. 2012). (2) A ”Planning based” strategy that takes into account vehicles that are expected to reach the intersection within a specific spatial range and create long-term driving plans for these vehicles. Planning-based approaches, as opposed to ad hoc negotiation-based approaches, offer more flexibility in cooperative driving and have the potential for higher traffic efficiency. However, the computational requirements of planning-based approaches increase significantly as the number of vehicles involved grows.

To overcome the limitations of significant computational time and sub-optimality, Pei et al. (2019) proposed a computationally efficient strategy based on Dynamic Programming (DP). Initially, they formulated the merging problem for vehicles as a MILP problem to show the increase in complexity with an increase in the number of vehicles. The following Figure 3.5 describes the scenario which was considered in their paper.

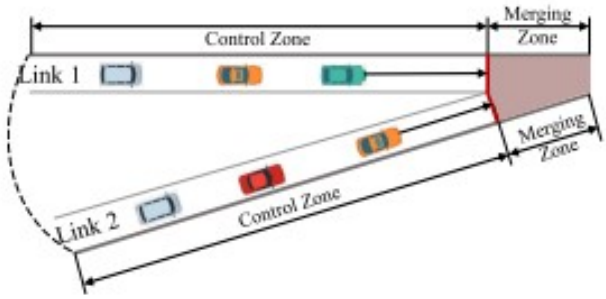


Figure 3.5: Single Lane Merge Zone, (Pei et al. 2019)

The mathematical model is as follows: To improve traffic efficiency by optimizing the passing order of vehicles, they formulate the objective function as:

$$J = \max\{t_i^{assign}\} \quad \forall \quad t_i^{assign} \in T^{assign} \tag{3.8}$$

where J represents the total access time of the vehicles, T^{assign} is the set of access times assigned to all the vehicles, and t_i^{assign} is the access time assigned to vehicle i.

Subject to the following constraints:

$$t_{i+1}^{assign} - t_i^{assign} \geq \Delta_{t_1} \tag{3.9}$$

This constraint ensures rear-end collision-free safety.

To formulate the optimization problem, a binary variable is introduced to represent the passing order between vehicle i and vehicle j. If $k_{ij} = 1$, it means that vehicle i can enter the merge zone prior to the vehicle j. Furthermore, to prevent converging collisions and ensure that only one vehicle can pass

through the merge zone, the following constraints are imposed:

$$t_i^{assign} - t_j^{assign} + M \cdot k_{ij} \geq \Delta_{t_2} \tag{3.10}$$

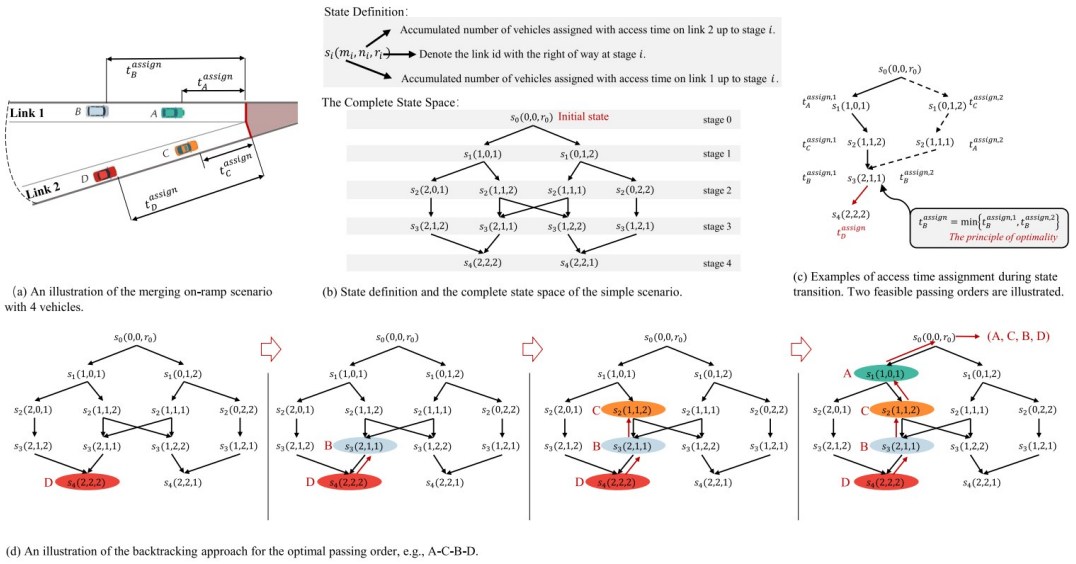
$$t_j^{assign} - t_i^{assign} + M \cdot (1 - k_{ij}) \geq \Delta_{t_2} \tag{3.11}$$

Note that the variable M represents a large positive constant. By multiplying the inequality constraint by M, the constraint becomes M times greater than the original inequality. This essentially ”relaxes” the constraint and allows the solver to explore different solutions (Bazaraa et al. 2005). Δ_{t_1} and Δ_{t_2} denote specific time difference thresholds that ensure the minimal allowable safe gaps between the vehicles for avoiding rear-end collisions and the converging collisions at the merge zone respectively.

Considering the number of vehicles in Link 1 of Figure 3.5 as m and the number of vehicles in Link 2 as n, using the Branch and Bound Method a globally optimal solution can be obtained if the numbers are small. For every vehicle in Link 1, there are n possible vehicles in Link 2 that it can interact with. For each pair of vehicles (i, j), we have one binary variable k_{ij}. Since k_{ij} can take on two possible values (0 or 1), there are 2 possible combinations for each pair. However, as the number of vehicles increases in each link, the size of the solution space increases exponentially (in the order of 2^{mn}). This proves that this method can be extremely time-consuming. To overcome this issue, they employed dynamic programming that can decrease the computational times to the order of a quadratic polynomial.

The task of determining a passing order can be viewed as the process of sequentially assigning the right of way to vehicles in the Merge Zone. Therefore, the optimization problem was regarded as a sequential decision problem, where the decision variable r_i represents the right of way. In a merging scenario, the decision variable r_i can take one of two possible values from the set of right-of-way options $RW = \{1, 2\}$. If $r_i = 1$, it indicates that a vehicle from link 1 is granted the right of way to enter the Merging Zone. Similarly, if $r_i = 2$, it signifies that a vehicle from link 2 is given the right of way.

To facilitate this decision process, the problem is divided into multiple stages, allowing it to be treated as a class of similar problems within the framework of dynamic programming. In this model, $(m + n)$ stages are required, excluding the initial stage, to assign the right of way to all vehicles in the Control Zone, where $(m + n)$ represents the total number of vehicles in the Control Zone. To simplify the notation and understanding, the stages are numbered from 0 to $(m + n)$, with stage 0 representing the initial stage and subsequent stages progressing until all vehicles have been assigned the right of way as can be seen in Figure 3.6. This approach allows for an organized and systematic allocation of the right of way and access times among the vehicles, considering each stage’s unique characteristics and requirements.



(a) An illustration of the merging on-ramp scenario with 4 vehicles.

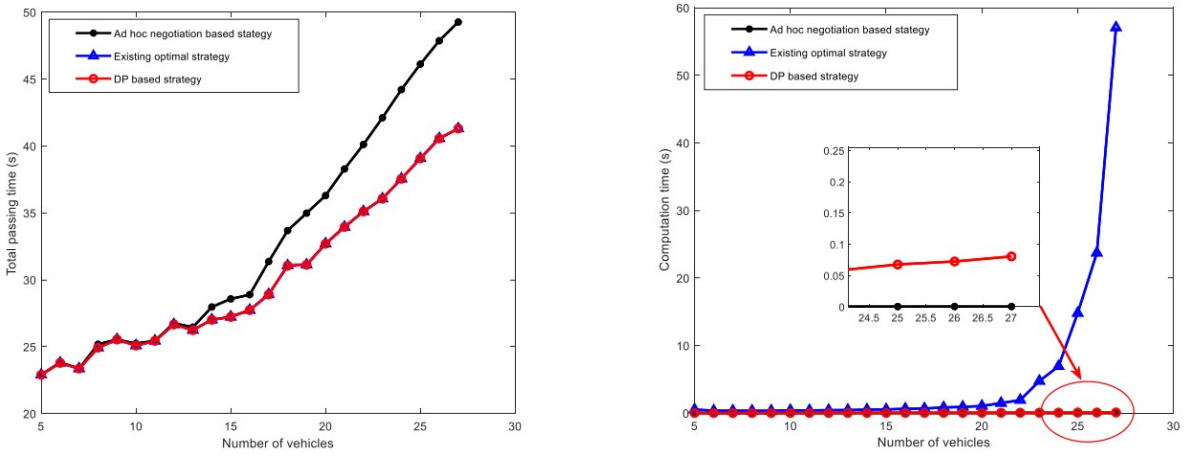
(b) State definition and the complete state space of the simple scenario.

(c) Examples of access time assignment during state transition. Two feasible passing orders are illustrated.

(d) An illustration of the backtracking approach for the optimal passing order, e.g., A-C-B-D.

Figure 3.6: Back Tracking of Dynamic Programming (Pei et al. 2019)

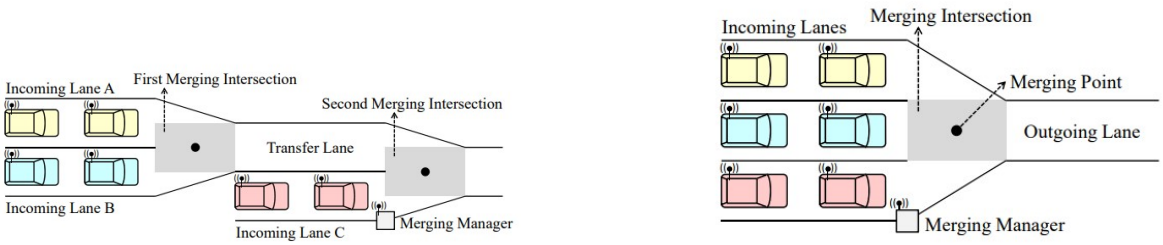
Although from the Figure 3.7a it can be seen that the existing optimal strategy and DP strategy have the same total passing time, with an increase in the number of vehicles, the computational time taken for the DP is significantly low as shown in Figure 3.7b. In a similar fashion, Lin et al. (2020) formulate and propose a DP algorithm to find the optimal solution for a two-lane vehicle merging problem. In addition to the general two-lane merging scenario, they extend their research to other scenarios as can be depicted from Figure 3.8 and improvise their DP algorithm to prove that it can efficiently minimize the average delays of the vehicles and reduce the time needed for all the vehicles to pass through the merge point when compared to some greedy algorithms.



(a) Total Passing Time vs Number of Vehicles

(b) Total Computation Time vs Number of Vehicles

Figure 3.7: Results of DP based strategy, (Pei et al. 2019)



(a) Consecutive lane merging

(b) Vehicle merging from 3 lanes

Figure 3.8: Different Scenarios of lane merging, (Lin et al. 2020)

Autonomous Intersection Management (AIM) was an early attempt by Dresner et al. (2008) to create a centralized algorithm for managing intersections with connected autonomous vehicles (CAVs). In AIM, intersections are represented as a grid of squares, with each square corresponding to a discrete time interval. When vehicles approach the intersection, they request permission to enter by providing their estimated time of arrival and velocity. The Intersection Manager (IM) uses this information to predict the vehicle's future trajectory in terms of time and space, determining which square it will occupy and when. The IM then checks for any conflicts with other vehicles' time-space reservations. If a conflict arises, the IM rejects the request, causing the vehicle to slow down and retry later after a timeout. If no reservation is assigned to a vehicle, it will stop before the intersection and make another request. If there are no conflicts, the vehicle proceeds and enters the intersection. AIM follows a query-based intersection management (QB-IM) approach, where vehicles ask the IM for safe passage and receive a YES/NO response. However, this approach can result in increased network overhead and reduced throughput.

Pei et al. (2022) extend their work (Pei et al. 2019) and propose a novel state-space formulation that represents the complete solution space of cooperative driving at signal-free intersections. Through extensive analysis, the algorithm is demonstrated to achieve optimality in cooperative driving under various traffic demand settings. The theoretical analysis provides insights into the workings of the proposed dynamic programming approach. Importantly, the results highlight that the computation time of the control algorithm is sufficiently short, enabling the real-time implementation of optimal cooperative driving.

3.4. Key Performance Indicators

As evident from the information provided above, the primary indicators (KPIs) employed to assess the models in question were throughput, utilization, and imbalance. This section will subsequently elaborate on each of these KPIs, explaining them comprehensively.

3.4.1. Throughput

Throughput, which is defined as the rate at which the material moves through the system per unit time (Thorne 2006), is a crucial metric in this industry and is usually measured in terms of the number of parcels per hour or day (a unit time). To cater to the rapidly expanding market and maintain high levels of efficiency, the conveyor systems must constantly evolve to handle higher volumes of packages. Achieving higher throughput in the parcel sorting industry is a challenging task, which requires careful planning and coordination of the infeed rates, parcel size, and velocities among many other factors.

$$Throughput = \frac{\sum(Number\ of\ Parcels\ processed)}{Time\_duration}[parcels\ per\ hour] \tag{3.12}$$

3.4.2. Utilization

Utilization is another metric that refers to the degree to which the system is being used or occupied relative to its maximum capacity (Haneyah et al. 2013). Higher utilization implies efficient usage of the conveyor system, with minimal empty space between parcels, thereby enabling more parcels to be transported within a given time frame.

$$Utilization = \frac{\sum_{n \in N} Parcel\_lengths_n + (N-1)Fixed\_gap}{V_{merge} * Time\_duration}[\%] \tag{3.13}$$

where N is the total number of parcels processed by the system and the $Fixed\_gap$, is the minimum gap that is required to avoid parcel overlapping or collision. Further, V_{merge} is the velocity of the merge conveyor and the $Time\_duration$ is the total time the system has been in running.

Both throughput and utilization are directly related to each other. When the utilization of the conveyor is low, there are more empty spaces between the parcels on the merge conveyor, leading to a reduced throughput. However, when considering the parcel size variations, there will be certain situations where a huge parcel occupies the most space on the merge conveyor which translates to less throughput. These kinds of situations are inevitable and are the primary reason for companies to consider the average throughput of the system.

3.4.3. Load Imbalance

One significant issue identified in the existing reservation control system is the notable difference in throughput between upstream and downstream infeed conveyors. To address this, a key performance indicator is introduced to measure the disparity in throughput between the minimum and maximum values for each infeed, thereby consolidating it into a single metric. Hence, the imbalance can be calculated as:

$$Imbalance = \frac{max_{f \in F}\{Throughput_f\} - min_{f \in F}\{Throughput_f\}}{max_{f \in F}\{Throughput_f\}}.100\ [\%] \tag{3.14}$$

where f is the infeed in a set of infeeds denoted by F.

3.5. Conclusion

In conclusion, the literature study conducted in this chapter helps to answer the second sub-research question that aims to identify a suitable control algorithm to overcome the issue of low utilization. The chapter emphasizes on the importance of optimal control in enhancing the efficiency of merge zones in parcel industries. The trade-offs associated with increasing belt speed and layout changes are discussed, highlighting the need for further investigation in this domain. The chapter, further explored various control algorithms for parcel merging and vehicle merging processes, aiming to improve system performance.

Upon thorough examination of the available literature, it became evident that a research gap exists in the realm of parcel industries regarding the utilization of different techniques like optimization or alternative scheduling approaches. This gap highlighted the need to explore and investigate various

methods that have the potential to enhance the parcel merging process—an essential aspect that significantly impacts the overall system throughput.

Although several researchers have employed exact algorithms such as Integer Linear Programming to address this issue, it is apparent that these approaches suffer from a notable drawback: they are time-consuming. As a result, there is a demand for alternative techniques that can offer more efficient solutions. Interestingly, dynamic programming has emerged as a promising approach in the field of vehicle lane merging, demonstrating favourable results. Nevertheless, it is important to acknowledge that this approach has not been implemented in the particular context of parcel merging thus far. Given the success it has shown in similar domains, it would be prudent to consider implementing dynamic programming in the field of parcel merging.

By incorporating dynamic programming into the parcel merging process, it is anticipated that the system's overall performance and throughput could be significantly improved. Leveraging the lessons learned from other related areas, this technique holds promise for addressing the challenges and complexities associated with parcel merging, thereby enhancing operational efficiency and customer satisfaction in the parcel industry.

4

Model Building

By creating virtual representations of real-world scenarios, simulation facilitates experimentation and evaluation, obviating the necessity for resource-intensive and time-consuming physical implementations. Simulation is a fundamental and indispensable tool across diverse fields, serving as a robust method for comprehending, analyzing, and forecasting intricate systems (Kellner et al. 1999). This capability provides researchers and practitioners with a safe and controlled environment to test alternative strategies, evaluate performance metrics, and make informed decisions. Moreover, simulation empowers the exploration of scenarios that are inherently challenging, hazardous, or infeasible to replicate in reality, thereby unlocking new insights and avenues for innovation. This chapter provides insights into the modelling steps followed and paves a path towards model experimentation which will be carried out in the chapter 5.

4.1. Modelling Steps

A model represents an actual system (Banks 1999). The development of a simulation model begins with a thorough understanding of the system that will be modelled. This understanding is established in chapter 2 through a detailed examination of the system's operations and control methods. Following this, the model conceptualization phase involves simplifying the real-world system. This includes identifying and breaking down the system to be modelled, as well as gaining a clear understanding of the relevant processes involved in the merging of parcels into the merge conveyor. The second chapter revealed that factors such as velocity profiles, parcel dimensions, and load-balancing techniques significantly impact the parcel merging process. Once all the relevant elements for accurately modelling the system are identified, the conceptualization phase can be translated into the implementation of the model. This involves implementing the system in software. Subsequently, the simulation model undergoes verification to ensure it performs correctly and validation to assess whether it can replace the real system for experimentation purposes. After the simulation model passes these checks, experiments can be carried out using the model to check for the feasibility of the developed model in real-world applications.

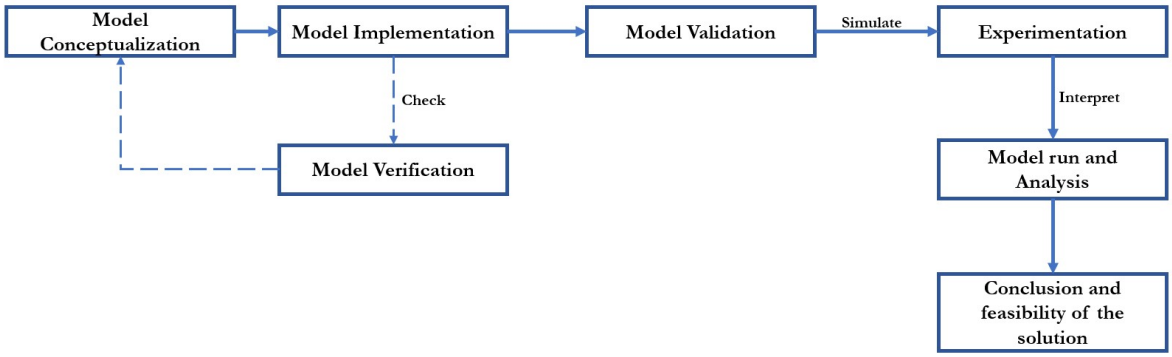


Figure 4.1: Modelling steps, Inspired from (Sharma 2015)

4.1.1. Step 1: Model Conceptualization

The initial phase of model development, as outlined in section 4.1, involves model conceptualization. This step is crucial as it sets the foundation for the entire modelling process. Model conceptualization involves identifying and defining the key elements and relationships that will be incorporated into the model. The aim of model conceptualization is to create a conceptual framework that represents the essential components and interactions within the system. During this phase, the model developer needs to make decisions regarding the scope and boundaries of the model, the level of abstraction, and the appropriate modelling techniques to be employed. Additionally, potential limitations and assumptions of the model should be acknowledged and documented. The output of this phase is a well-defined and structured conceptual model that serves as a basis for further model development, parameterization, and validation.

Model Description and Assumptions

The system consists of a single merge conveyor and six infeeds with triple infeed belts can be seen in Figure 4.2. A few technical descriptions of the system are mentioned in the Table 4.1.

Table 4.1: System Parameters

Name	**Parameter**	**#**	**Units**
Merge Conveyor	Length	15	m
Merge Conveyor	Width	1,5	m
Infeeds	Number	6	-
Infeed Position (1,2)	Distance from the merge start	1,75	m
Infeed Position (3,4)	Distance from the merge start	7,92	m
Infeed Position (5,6)	Distance from the merge start	11,16	m
Infeed inclination to merge	Angle	30	degrees

The simulation model doesn't encompass every single detail and behaviour, so it's necessary to make assumptions and abstractions when creating the model. This can result in minor inaccuracies within the simulation but can reduce the complexity of the model. It's important to acknowledge these inaccuracies since it's inevitable to include requirements, constraints, and assumptions when building a simulation model.

Model Constraints

1. The merge conveyor runs at a constant velocity of 2.2 m/s

2. Each infeed can deliver a maximum of 3200 parcels per hour
3. The inter-departure time between two parcels from the same infeed cannot be less than 1 second
4. A gap of at least 15 cm among two parcels is mandatory on the merge conveyor
5. The dimensions of the parcels cannot exceed the specified range
6. Parcels cannot be interchanged once placed on the infeed
7. No two parcels can be on the same acceleration conveyor of the infeed, as only one parcel can be accelerated or decelerated at a time.
8. In the three-belt infeed, each belt is approximately 70 cm long and the junction belt travels at a constant velocity.

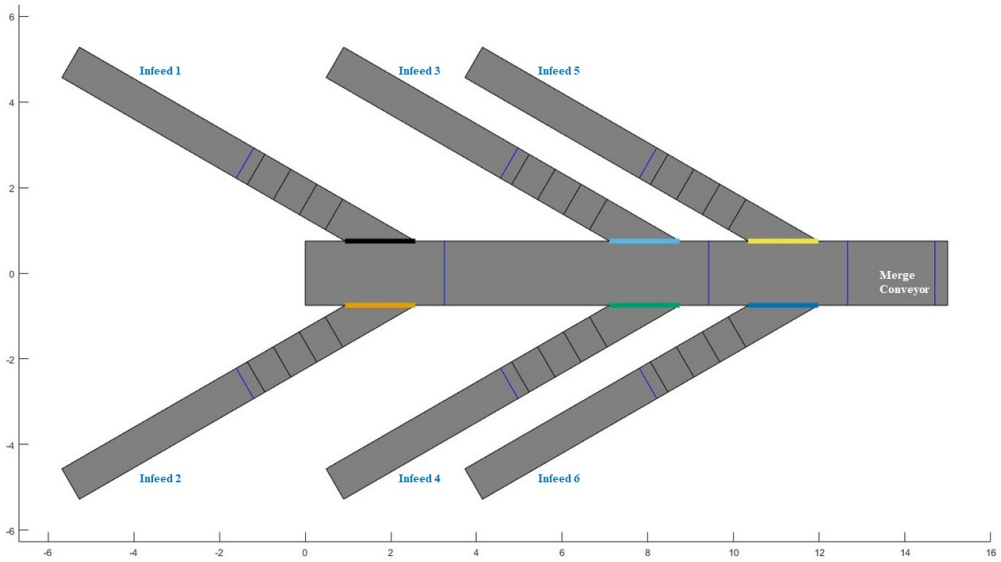


Figure 4.2: Layout of the system

Model Assumptions

1. Parcels are placed on the conveyor with the shorter side leading
2. Parcels do not slip
3. Parcels are fully rotated after they reach the merge conveyor into the flow direction of the merge conveyor
4. All the parcels are centred on the conveyors
5. The length of the parcel is known on the first infeed belt of the three infeed belts.
6. Parcels are generated in the simulation based on the maximum capacity of the infeeds using a normal distribution, while in real-time they are placed by an employee or a robot at random intervals of time.

7. Velocity profiles for the infeeds are not taken into consideration. However, the maximum time for the parcel to reach the merge conveyor is factored in to stay close to reality.

8. Parcels are considered by the algorithm for attaining the best possible sequence only when they are ready to be merged. In essence, they are on the infeed acceleration belts waiting for the command from the merge controller to start accelerating towards the merge conveyor.

Processes in the model

The entire model comprises three major processes. Each of them will be discussed in detail in this part of the chapter.

Process 1: Parcel generation

In order for a parcel and its associated processes to exist within a simulation, it is essential to construct a parcel generator. The role of the parcel generator process is to generate parcels with specific dimensions based on a normal distribution until the simulation ends. These parcel-generation events occur at random intervals of time based on the infeed capacity of 3200 parcels per hour. During this process, the parcels are only added to the infeeds if there is sufficient space available to accommodate the parcel's length while still maintaining the minimum required gap between parcels. If the infeeds cannot accommodate the incoming parcel, it is held until the previous parcel has been injected into the merge conveyor. The pseudo-code for this parcel generation can be seen in the algorithm 1 of Appendix B.

Once a parcel is generated and enters the infeed, its entry time into the infeed queue and the corresponding parcel length are recorded in an array called "parcel entries." This array serves a crucial purpose in the subsequent processes. It effectively communicates that the parcels are prepared for merging and are awaiting information on the confirmed time at which they should reach the merge point. For instance, an example of the "parcel entries" array for a system with three infeeds could resemble the following representation: [[(0.5, 3), (1, 15)], [(2, 10)], [(6, 17), (1, 34), (2, 25)]]. This list comprises the entry time and length of parcels for each of the three infeeds, formatted as [(time, length)] pairs.

In order to continue into the subsequent processes, it is important to gain an understanding of the functioning of the *merge conveyor class*, algorithm 3, in managing the merging process. Let us consider a hypothetical scenario where there are parcels of random lengths present on each of the six infeed conveyors, awaiting merging onto the merge conveyor. Prior to starting the process, an imaginary segment is generated that is aligned with the axis of the merge conveyor, which is located before the actual start of the merge conveyor itself (for clarification, please refer to the green segment in the Figure 4.3).

This segment has the same velocity as that of the merge conveyor and is the event trigger for this model. The length of this segment is predetermined to be a maximum of $500\ cm$. This can be varied but the maximum cannot exceed $500\ cm$ as the fly-through detection is located at $650\ cm$ towards the negative x direction. The maximum length of $500\ cm$ for the segment is considered to make sure that even if there is a fly-through parcel that has the maximum length of the entire parcel distribution, it is not affected by the last parcel of the segment in front of it. The forthcoming analysis will elaborate on a control algorithm that aims to efficiently occupy the given parcels within the segment. The algorithm fills the parcels in a manner that maximizes the segment's utilization. In the event that there is residual space left unoccupied after assigning slices to all the parcels, the subsequent segment begins from the trailing edge of the last-filled parcel.

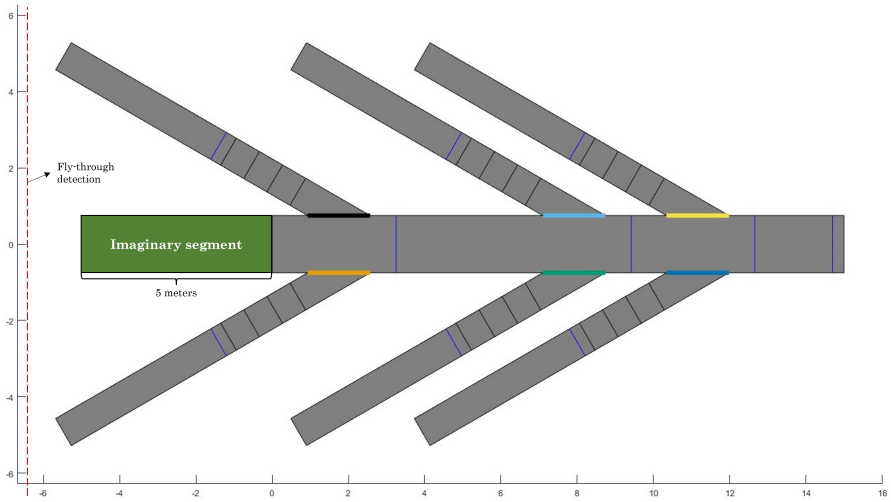


Figure 4.3: Imaginary Segment and fly-through detection point

Process 2: Control Algorithm

Following the initial entries of parcels, these entries are utilized to determine the optimal sequence for the parcels on the infeeds that can fill the entire segment while maintaining appropriate gaps between the parcels. This is where DP becomes essential. Prior to exploring how DP identifies the best sequence, it is important to comprehend how a problem can be solved using DP. According to Nayak (2020), a straightforward example of DP is the solution to the Fibonacci series. Essentially, the Fibonacci series is a sequence where each number is the sum of the two preceding numbers. To compute the Fibonacci number at position n, knowledge of the Fibonacci numbers at positions n-1 and n-2 is required. Let's consider the following example of finding fib(5). To know the value of fib(5), it is necessary to compute the value of fib(3) and fib(4), which are the two preceding numbers. This is a general recursive problem.

It is evident from the Figure 4.4 that there are several computations of the function for the same number in different stages (highlighted in the same color). If n is the number of stages, then the time complexity would be something as follows after approximation:

$$T(n) = 2 * T(n-1) + 1 \tag{4.1}$$

Breaking down this function, it can be seen that to find Fibonacci of n^{th} element, Fibonacci of $n-1$ and $n-2$ must be calculated for n+1 stages.

For the next stage:

$$T(n-1) = [2 * (T(n-2) + 1)] + 1 \tag{4.2}$$

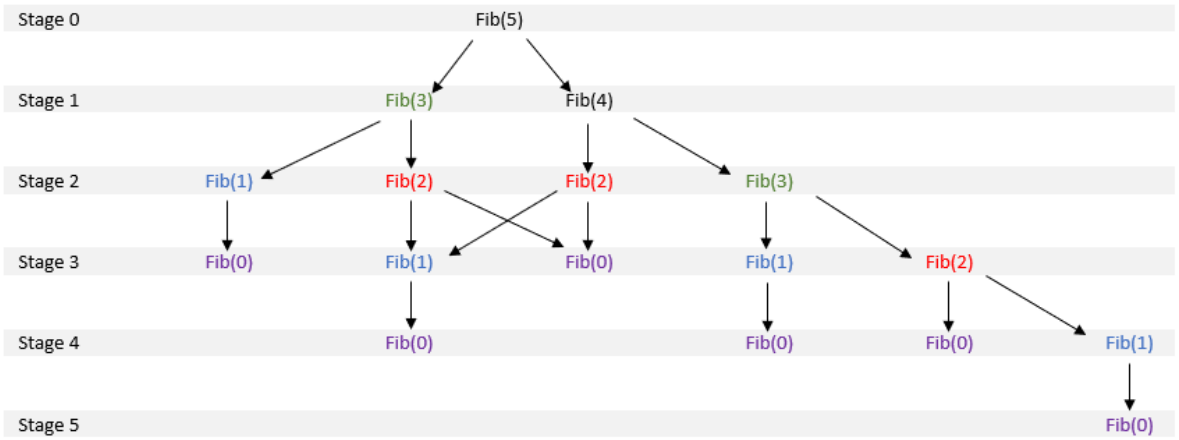


Figure 4.4: Fibonacci Series recursion tree

When doing this for k stages, the time complexity raises and reaches an order of 2^k. To avoid this huge computational time, the DP approach has a technique called '*Memoization*' (Cooper et al. 1981). Memoization is a technique commonly used in the DP to optimize the computation of recursive functions. It involves storing the results of expensive function calls and reusing those results instead of recomputing them. This helps to avoid redundant calculations by storing the values of previously solved subproblems in a data structure, such as an array or a hash table. When a subproblem needs to be solved again, the stored value is retrieved, eliminating the need for recalculation. It can be seen from Figure 4.5 that the computation time has significantly reduced to an order of n. The function is called only $n + 1$ times. By employing DP, we convert a single optimization problem in $n\ dimensions$ into $n\ separate\ one - dimensional$ optimization problems that can be solved individually. Employing DP for this problem not only decreases the computational time but ensures an exact solution that can be the best fit in a given situation.

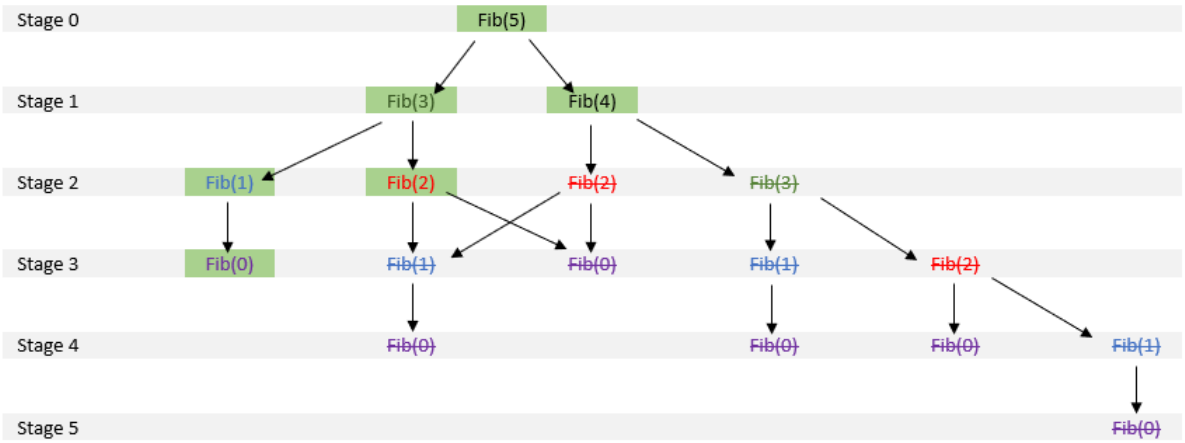


Figure 4.5: Fibonacci Series recursion tree using DP

Now that there is a clear picture of how the DP works, it can be easy to understand how it can be used to optimally utilize the entire segment with the available parcel information. To tackle the present problem using DP, it is essential to establish the state space and state transition for the model (Pei et al. 2019).

To enhance the comprehension of the DP approach, let's examine an example. It is important to note that for this hypothetical example, the safety gap requirements between the parcels are neglected for better understanding. Presume two infeeds, each containing two parcels. For illustrative purposes, we will assume a remaining 60 centimetres of gap on the segment that can accommodate the four available parcels. Infeed 1 has a queue of two parcels awaiting processing: $[10, 15]$, while Infeed 2 also has two parcels in its queue, to be merged: $[25, 22]$. The state space and state transition shall be explored from here.

State Space

The state space would encompass $(x + y + 1)$ stages, where x represents the number of parcels from Infeed 1 and y denotes the number of parcels from Infeed 2, ranging from stage S_0 to S_{x+y}. Each stage is described by a triplet state (x_i, y_i, g_i), where g represents the remaining gap after assigning a parcel, and i signifies the stage number. The initial state is denoted as $S_0(0, 0, g)$, and the final state is $S_{x+y}(x, y, g)$, in this case, $S_0(0, 0, 60)$ and $S_4(2, 2, -18)$. With an increase in the state space, there is a possibility that the computational time for DP can be higher. However, in the current situation, the state space is bounded by the amount of available space on the segment, the number of infeeds and the number of parcels in each infeed. Hence, computational time would not be a problem.

State transition

State transition refers to the progression of states after a parcel has been assigned to the segment. The state transition equation emerges as follows:

$$S_i(x_i, y_i, g_i) = h((x_{i-1}, y_{i-1}, g_{i-1}), g_i) \tag{4.3}$$

where h(.) is considered as the state transition function (Figure 4.6).

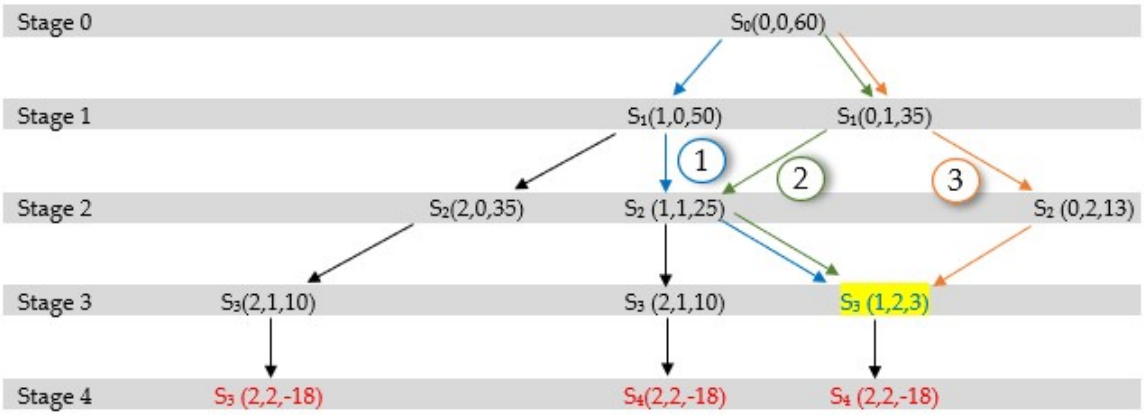


Figure 4.6: Assigning parcels using DP

As mentioned by Pei et al. (2019), the aforementioned state space and state transition possess the following properties:

Property 1: There is a decision-making process consisting of multiple stages, starting from stage 1 to stage "i." The outcomes or results of the decisions made in each stage are reflected in the parameters of a state called "S_i." In simpler terms, as the decision-making process advances from stage 1 to stage i, the choices made at each stage have an impact on the state S_i.

Property 2: The shift from one state to another happens when moving from one stage to the next stage in the decision-making process.

Property 3: Different orders of the state can attain the same state in the next transition. For instance, $S_1(1, 0, 50)$ and $S_1(0, 1, 35)$ attain the same state $S_2(1, 1, 25)$ in stage 2.

Property 4: Infeasible solutions are directly eliminated during the construction of the solution space by keeping track of the parcels that have been assigned or not.

According to Nelson (1995), the Markovian property is a stochastic property that has a form of historical dependency where the probability of each event depends only on the state attained in the previous event. By virtue of Property 1, the state exhibits the Markovian property, which represents the feasible conditions of the DP model (Blackwell 1962). Property 2 enables a remarkable reduction in the number of transitions in the DP model. Similarly, Property 3 facilitates an extensive decrease in the number of states in the DP model. Property 4, while ensuring optimality, substantially reduces the size of the solution space. Consequently, the state space solely encompasses all feasible passing orders of the parcels.

From the Figure 4.6, it becomes evident that there are multiple paths leading to the optimal stage of leaving only a 3-centimetre gap. It is worth mentioning that certain states highlighted in red are considered infeasible since a negative gap is not possible. Furthermore, paths 1 and 3 of Figure 4.6 violate constraint number 3 discussed in the subsection 4.1.1 and hence cannot be considered as a feasible solution leaving path 2 as the optimal solution for this example. Nevertheless, one limitation of the DP approach is its inherent bias towards starting from the left and searching the solution space. In the context of a parcel industry, this sequential left-to-right approach may not be optimal, as it can lead to downstream infeeds being left with insufficient space due to upstream infeeds delivering parcels at higher rates in some cases.

To mitigate this biasing issue, a priority system can be introduced based on the occupancy levels of the infeeds. By assigning higher priority to the more occupied infeed compared to others, a more balanced allocation of space can be achieved. This approach has shown promising results in previous studies (Haneyah et al. 2013), making it a viable option to ensure that the control algorithm remains unbiased towards any specific infeed. To facilitate the sorting based on the maximum filled queue, a sorting algorithm known as Heap sort is employed for the current model. However, it is worth noting that there is always a possibility to shift priorities to other infeeds in case one of the infeeds has more infeed capacity. For instance, if an infeed can deliver 10000 parcels per hour and the others can deliver relatively less number of parcels per hour, using a most filled queue technique would not be the best option as the infeed with higher capacity always has more parcels coming in than that of the others. This can result in imbalance again. Since, the capacity of all the infeeds is the same, a maximum filled queue option is suitable for the current model.

A max heap is a complete binary tree where the value of each node is greater than or equal to the values of its child nodes (Schaffer et al. 1993). This ensures that the maximum element is always at the root of the heap. The process of creating a max heap involves building the heap bottom-up, starting from the last non-leaf node and repeatedly ”sifting down” elements to their correct positions. This ensures that the largest element moves to the root of the heap. Once the max heap is constructed, the algorithm repeatedly extracts the maximum element from the root of the heap and places it at the end of the array. This is achieved by swapping the root with the last element, reducing the heap size, and then performing a ”sift down” operation to maintain the heap property. By repeatedly extracting the

maximum element and maintaining the heap property, the array eventually becomes sorted (Figure 4.7). The sorted elements are stored in the remaining portion of the input array, starting from the end.

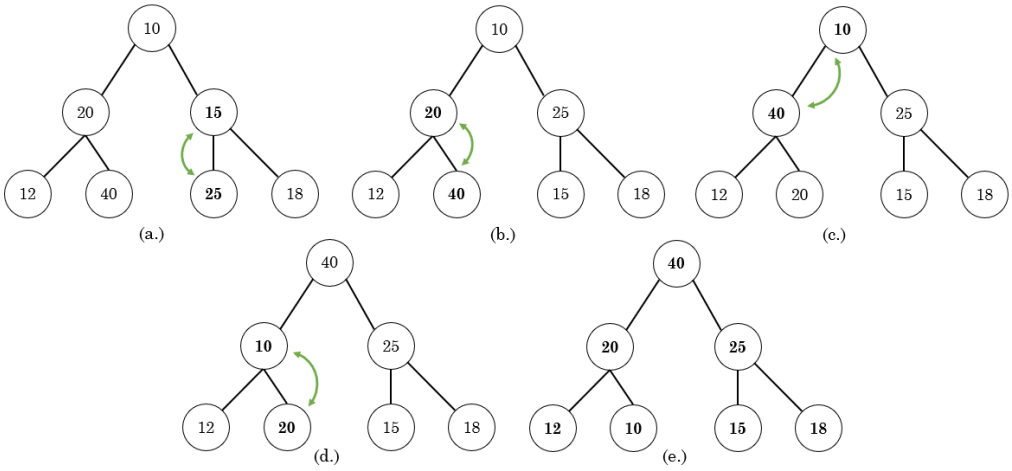


Figure 4.7: Example of a max heap sorting

Heap sort efficiently maintains a max heap, enabling quick access to the maximum element. A combination of the above two algorithms can provide an optimal solution by improving the utilization using the DP and maintaining a load balancing among the infeeds by considering the maximum filled queue first using the Heap sort algorithm. This ensures that no infeed has the highest priority and makes sure that the infeed with the maximum filled capacity is always considered.

Process 3: Handling entry and exit in a time frame

This function is responsible for managing the parcels entering and exiting a conveyor system within a specific time period. It works with multiple queues in the system. The function processes the parcels by comparing their arrival and exit times with the given time frame. It ensures that the parcels are handled in the correct order based on their arrival and exit times.

For each queue, the function checks if there are more parcels waiting to enter or exit. Inside a loop, it examines the next entry and exit parcels. If both are present and their times fall within the specified time frame, the function takes the appropriate action. If the entry parcel arrived before the exit parcel, it adds the entry parcel to the corresponding queue. If the exit parcel occurred earlier, it removes a parcel from the queue. The function continues processing parcels as long as there are both entry and exit parcels within the time frame.

In the given scenario, when a sequence of infeeds is obtained (e.g., [infeed1, infeed2, infeed3]), a calculation is performed to determine the time it takes for a specific slice on the segment of the conveyor system to reach each corresponding infeed. This calculated time serves as the exit time for the parcel from its respective infeed. If the calculated time for the assigned slice on the segment to reach the assigned infeed is denoted as x, the parcel undergoes acceleration to ensure it reaches the merge point within the specified time frame. In practical terms, if the distance between the merge point and the location of the parcel on the infeed is too great to be covered in the assigned time, the parcel needs to be accelerated at $t + x$ seconds. This acceleration allows the parcel to attain the necessary speed so that it arrives at the assigned time, ensuring timely merging within the conveyor system.

4.1.2. Step 2: Model Implementation

This section focuses on the implementation of the conceptualized model described earlier in subsection 4.1.1. It provides insights into how the model is applied in the simulation process.

There are several types of simulations that are used based on the project requirements. Based on the state change of variables, there are two primary simulation types (Maria 1997): Discrete Event Simulation (DES) and Continuous Simulation. DES models the behaviour of a system as a sequence of discrete events that occur at specific points in time. These events can include arrivals, departures, changes in state, or any other significant occurrences in the system. The simulation tracks the chronological order of events and the system's state, allowing for the analysis of system performance and behaviour. On the other hand, continuous simulation involves state variables that change continuously over time. This simulation is mainly used to simulate the behaviour of complex and dynamic systems, which can also be done by DES. While Continuous Simulation has its merits, such as the ability to model smoothly changing processes, it might not be the most suitable option for our current simulation task. Given the discrete nature of our events and the need to capture detailed interactions among distinct entities, DES emerges as the preferred choice to develop the simulation model.

Discrete Event Simulation

In various applications, queuing models have been instrumental in understanding system characteristics, primarily through analytical solutions. However, it is important to note that analytical solutions are feasible only for a limited set of problems. When dealing with complex queuing systems, simulation is commonly employed. DES has emerged as the primary tool for drawing conclusions about intricate queuing networks (Babulak et al. 2010). It is uncommon to come across simulation studies that utilize continuous simulation for analyzing queuing systems. To elaborate, queuing models are mathematical representations used to study the behaviour and performance of systems with waiting lines or queues. Analytical solutions involve solving equations and formulas to obtain precise results, providing valuable insights into the system's characteristics such as average wait times, utilization rates, and queue lengths. However, these analytical solutions are only applicable to relatively simple queuing models with certain assumptions and conditions.

As queuing systems grow in complexity, analytical solutions become increasingly challenging or even infeasible to obtain. This is where simulation techniques, specifically discrete event simulation, come into play. DES involves modelling the discrete events that occur within a system, such as customer arrivals, service completions, and queue dynamics (Allen 2011). By simulating the system over time and capturing the interactions between events, DES allows for studying and drawing conclusions about complicated queuing networks. A generic controller for a single replicate of discrete event simulation is shown in Figure 4.8. As the current system involves queues (parcels in different infeeds waiting to be merged into the merge conveyor), DES is a great tool to implement and analyse the developed control algorithm.

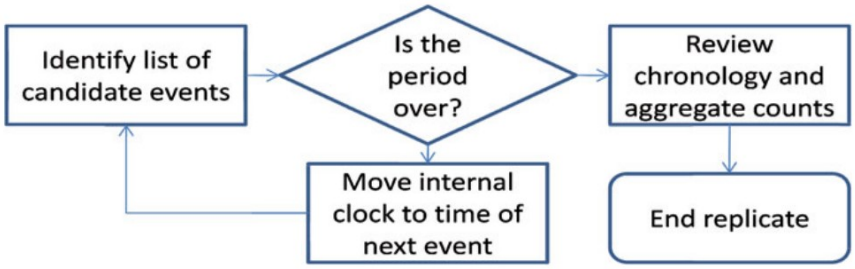


Figure 4.8: Generic controller model for a single replicate of discrete event simulation, (Allen 2011)

DES offers several advantages in modelling a wide range of systems (Banks 1999). Firstly, DES exhibits high flexibility, enabling the representation of both simple and complex systems, as long as they can be described in terms of discrete events and states. Secondly, DES is event-driven, focusing on significant events and simulating only when necessary, resulting in efficient utilization of computational resources and faster simulations. Additionally, DES captures the dynamic behaviour of a system over time, facilitating the study of system interactions and comprehension of complex scenarios. Moreover, DES is scalable and capable of handling large-scale systems by simulating only essential events, making it suitable for analyzing systems with numerous components or entities. However, DES also presents certain disadvantages. Building a DES model can be complex, necessitating meticulous identification and representation of events, entities, and system states. Furthermore, due to the discrete nature of events, DES may not capture continuous changes in the system state with high precision, leading to some level of approximation when representing system behaviour.

Software implementation

Multiple methods exist for conducting DES, and numerous software options are available, including Arena, Matlab, Simulink, Python, Demo3d, and others. Each software package has its own set of strengths and weaknesses. However, for the purpose of this study, Python was selected as the preferred software for performing the simulation because it offers various advantages that make it well-suited for DES. Firstly, Python is a versatile and widely-used programming language with a large user community, extensive documentation, and numerous libraries specifically designed for simulation purposes (Python Core Team 2019). This provides researchers with a wealth of resources and support. Secondly, Python's syntax is relatively simple and readable, allowing for easy implementation and modification of simulation models. Additionally, Python offers seamless integration with other scientific and data analysis libraries, enabling researchers to analyze simulation outputs efficiently. Lastly, Python's open-source nature and free availability make it a cost-effective option compared to commercial software packages.

With the advantages it holds and the possibilities of using various open-source libraries for simulating, Python was chosen as the mode to perform the simulations.

Hardware and Software used:
Hardware:

- Intel(R) Core(TM) i7-10850H CPU @ 2.70 GHz
- 64 GB RAM
- Windows 10 Enterprise

Software:

- Python 3.11.3
- Visual Studio Code IDE

4.1.3. Step 3: Model Verification

According to Thacker et al. (2004), model verification involves that the model implementation accurately reflects the developer's conceptual depiction of the model and its solution. To ensure that the model is verified and provides the expected results, manual checks were put into place to check for any unwanted effects.

1. *Check: Are the slices being assigned to the parcels appropriately?*
   When parcels are allocated to the slice, the anticipated result comprises the starting and ending locations of the slice, as well as the corresponding infeed from which the parcels originate. The illustration presented in Figure B.1 demonstrates that considering the parcels available at a given moment when the segment is ready to be filled, the segment is utilized to its maximum capacity.

2. *Check: Is any of the infeed delivering two parcels one behind the other, violating the inter-departure time constraint?*
   From the model constraints discussed in subsection 4.1.1, it can be seen that no two parcels can come from the same infeed one after the other and the infeed cannot deliver more than 3200 parcels per hour. Running the simulation several times proved that these constraints were not overlooked. A small example can also be seen in the Figure B.1.

3. *Check: The slice's placement on the segment should be chosen in such a way that the parcel can reach the merge.*
   In order to assign a slice, it is imperative to verify that the parcel can reach the designated slice within the specified timeframe. This verification process guarantees that parcels have the capability to reach the merge point. If a parcel is unable to meet this criterion, it will not be taken into consideration for the slice assignment. This assessment relies on the calculation of time which is based on the distance between the slice and the infeed, as well as the velocity of the merge.

4.1.4. Step 4: Model Validation

Model validation typically refers to the process of confirming that a computerized model, within its relevant domain, demonstrates a satisfactory level of accuracy that aligns with its intended purpose Sargent (2011). When validating a model, the focus is on evaluating its performance about its intended purpose or application. This purpose can vary depending on the domain and context of the model. The validation process involves comparing the model's outputs or predictions against known or observed data. This data is typically separate from the data used to train the model, ensuring an unbiased evaluation. By comparing the model's outputs with the actual outcomes or observations, one can assess the model's accuracy and determine whether it meets the desired level of performance.

Validation is not a one-time activity but an iterative process that involves refining and improving the model based on the evaluation results. If the model does not demonstrate satisfactory accuracy or align with its intended purpose, adjustments may be necessary, such as modifying the model's structure, adjusting parameters, or incorporating additional data.

There are various validation techniques as suggested by Sargent (2011):

- **Extreme Condition Tests**: The model's structure and outputs need to exhibit plausibility even when faced with highly improbable and extreme combinations of factor levels within the system. This requirement ensures that the model remains reliable and consistent in its predictions across a wide range of scenarios. As an illustration, if the levels of in-process inventories were to reach zero, it would be expected that the production output would typically be zero as well. By accounting for such unlikely conditions, the model demonstrates its robustness and ability to handle various circumstances that may arise in the system.
  Test 1: When the infeeds can deliver only 100 parcels per hour, the utilization drop should be very significant.

  Test 2: Zero merge speed should result in an error.

- **Parameter Variability - Sensitivity Analysis**: This method involves systematically modifying the input values and internal parameters of a model in order to observe and analyze their impact on the model's behaviour and output. It is crucial for the model to accurately capture the relationships that exist in the real system it represents. This technique can be employed in both qualitative and quantitative ways, allowing for the assessment of not only the directions but also the precise magnitudes of the model's outputs. Parameters that exhibit sensitivity, meaning they significantly influence the model's behaviour or output, must be carefully calibrated before the model can be properly utilized.

Sensitivity Analysis

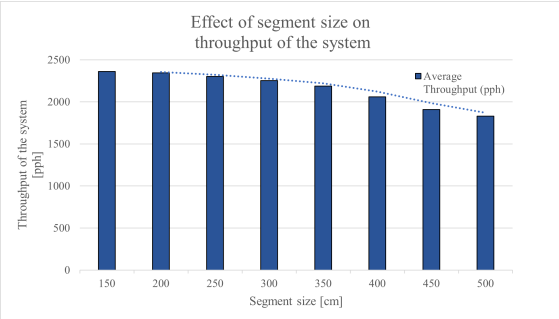
The utilization and throughput of the system are influenced by major parameters, including the infeed parcel arrival rate, the amount of space available for parcels on the infeed, and the speed of the conveyors. These findings were concluded in chapter 2 of this thesis. Additionally, the segment size is a significant factor in the current model that affects the system's utilization and throughput. In subsection 4.1.1, it was mentioned that the segment size can vary from the maximum length of the parcel distribution to a maximum of 500 cm. By varying the segment size, it is expected that changes in utilization and throughput will be observed.

In the current model, the segment size plays a crucial role in determining the number of parcels processed during each iteration. In a no fly-through parcel case, irrespective of the size of the segment, the n^{th} segment starts from the trailing edge of the last parcel on the $(n-1)^{th}$ segment. Nonetheless, when a fly-through parcel is detected, the subsequent segment begins after the trailing edge of the fly-through parcel. This means that as the segment size increases, the utilization of the system is expected to decrease when there are insufficient parcels to fill the segment. This is because larger segments are more likely to have wasted space due to the presence of fly-through parcels when there are not enough parcels to fill the segment. Having mentioned that reducing the segment size to extremely low values can conversely reduce the utilization as the algorithm struggles to allot longer parcels into the segment due to limited space.

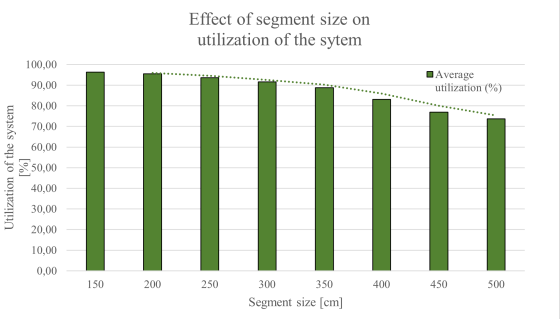
To identify an optimal segment size in the presence of fly-through, the simulation was executed ten times for segment size ranging from 150 to 500 cm with an increase of 50cm, and the analysis of the results, as illustrated in Figure 4.9, reveals that reducing the segment size has a notable impact on utilization in the presence of a fly-through. Specifically, decreasing the segment size resulted in higher throughput and utilization. To comprehend this phenomenon more effectively, let us consider an example. Suppose there are only four parcels in the entire system when a segment of 500 is generated. Assuming all parcels are of equal size of 50 cm for the purpose of explanation, it becomes evident that they occupy 260 cm (including the fixed 15 cm gap between parcels) of the 500 cm segment. However,

if a fly-through parcel is detected while the slices on this segment are being assigned, as mentioned earlier, the next segment begins after the trailing edge of the fly-through parcel.

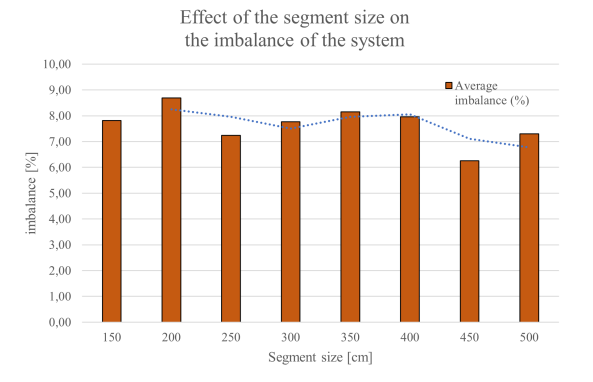
Consequently, the remaining 240 cm becomes wasted space in the segment because of the absence of any other parcel that can fill this space. A cumulative of this wasted space in similar situations throughout the simulation, results in lower utilization. However, it is worth noting that if there are parcels available in the system, the leftover segment will still be utilized until the fly-through parcel's leading edge. If it was a segment size of 150 cm for the same example, it can be easy to imagine that there is no wasted space and thus better utilization. Therefore, considering the existing constraints of the model, a smaller segment size in the presence of fly-through parcels can enhance both utilization and throughput.



(a) Average Throughput for different segment sizes



(b) Average Utilization for different segment sizes



(c) Average Imbalance for different segment sizes

Figure 4.9: Effect of Segment Size

This outcome of larger segment sizes attaining lower utilization and throughput can also be attributed to two major factors that were concluded in the chapter 2. Firstly, the timing of parcel announcements played a crucial role. Under the same parameter settings, a larger segment size led to improved throughput and utilization if the parcels could be announced slightly earlier. This means that the position of PEC must be further upstream of the infeed to have prior information on the parcel arrivals. Secondly, the infeed capacity of 3200 parcels per hour proved to be significant. If it is feasible to achieve an individual throughput surpassing 3200 parcels per hour per infeed, larger segment sizes may yield similar results to that of the smaller segment sizes. Additionally, the overall imbalance in the system is influenced by several factors. These factors include the distribution of parcel lengths, the order in which parcels are allocated, and the space constraints of the infeeds. These elements have a

more significant impact on the system's imbalance than the segment size alone.

4.2. Conclusion towards Experimentation

Within this chapter, an elaborate simulation model utilizing Dynamic Programming is constructed using discrete event simulation techniques for the specified layout which answers the third sub-research question. Throughout the development process, the concept of the control algorithm undergoes significant advancement, maturing rapidly to enhance the model's effectiveness. Additionally, the model is subjected to thorough verification and validation processes, confirming its readiness for subsequent experimentation. However, it is crucial to acknowledge that the validation and verification procedures conducted do not imply that the model is flawless or error-free. To truly assess its strengths and weaknesses, a comprehensive evaluation can only be achieved through experimentation. This evaluation will be carried out in the next chapter and will shed light on the model's performance under various conditions and scenarios, allowing for a more nuanced understanding of its capabilities.

The upcoming chapter will primarily focus on the execution of these experiments, aiming to extract valuable insights regarding the model's behaviour and performance. Through these systematic experiments, any limitations can be uncovered, identify potential areas for improvement, and gain a deeper understanding of the model's overall effectiveness.

5

Model Experimentation

Once the simulation model has been created, verified, and validated, it becomes a valuable tool for conducting experiments. Using DES, simulation experiments can be performed to explore various scenarios and compare them with the existing algorithm employed in the industry. This allows for a comprehensive assessment of the advantages and disadvantages associated with the developed model. The initial step in this process involves determining the appropriate simulation run time. This establishes the duration necessary for the simulation to run in order to facilitate a fair comparison. Subsequently, multiple scenarios can be tested under the same parameter settings, and the resulting KPIs, as discussed in chapter 3, can be utilized for comparing the outcomes. By comparing these scenarios, it becomes possible to ascertain the superiority of the current model over the existing algorithm. Furthermore, in this chapter, a cost-benefit analysis is conducted to evaluate the feasibility of further investing in the development of the proposed model for real-time applications.

5.1. Simulation time

In DES, the duration of the simulation plays a crucial role in accurately capturing the system's behaviour and evaluating its performance (Fishman 2001). A sufficiently long simulation time is essential to ensure that the system has stabilized and reached a steady-state condition. This allows for reliable measurements of KPIs and a comprehensive analysis of the system's behaviour. Running the simulation for a longer duration helps to mitigate the influence of transient behaviour, startup effects, and initial conditions. It allows the system to stabilize and reach a point where its behaviour can be reliably analyzed and evaluated. Moreover, a longer simulation time provides a more representative snapshot of the system's performance under realistic conditions, as it considers a larger sample size of parcels and captures a broader range of system dynamics.

To achieve this steady-state condition, the simulation must run for a duration that allows for an adequate number of events and interactions to occur. In the case of the developed algorithm for parcel sorting systems, the utilization metric, for example, depends on the successful passage of parcels through the entire system, including the infeed conveyors and the merge conveyor. It is only after several parcels have been processed that the utilization metric can be accurately calculated. Since the trajectory of parcels depends on the preceding parcel, the accurate behaviour of a real system can only be determined after several parcels have passed through the initial startup phase. It can be seen in Figure 5.1 that the utilization increases rapidly for a certain time and attains a plateau. To ensure that this condition is true, the simulation times of half an hour and one hour are considered. The simulation

was run for 20 iterations for both conditions to check for variability in the results.

By running simulations for 30 minutes and one hour, we can assess the algorithm's performance over extended periods and gain insights into its robustness, stability, and efficiency. These longer simulation times provide a more comprehensive understanding of the system's behaviour and allow for more confident conclusions to be drawn regarding its performance and suitability for real-world applications.

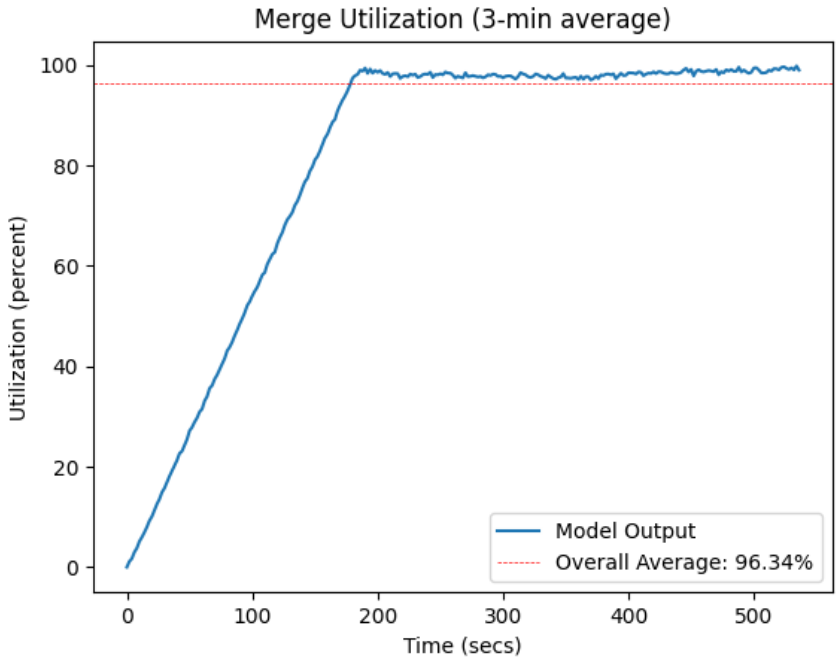
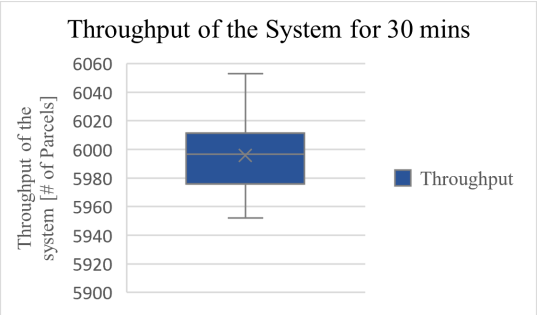


Figure 5.1: 3- minute average of the merge utilization

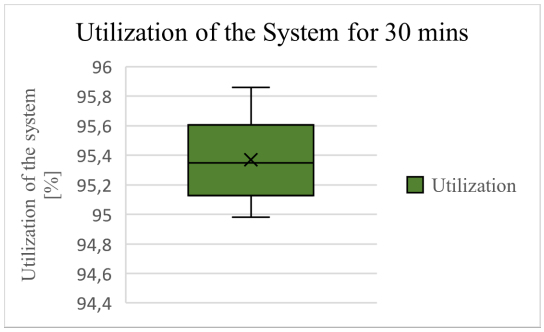
Simulation time: Half an Hour

In order to assess the algorithm's performance over an extended period, a simulation time of half an hour was chosen. This duration allows for a sufficient number of events and interactions to occur within the parcel sorting system, providing a comprehensive view of its behaviour. The simulation was executed for 20 iterations to account for any variability in the results with the fly-through occurrence rate of about 10 per cent of the total parcel generation. The graphs obtained from the simulation provide valuable insights into the system's performance during this timeframe. Key performance indicators such as throughput, utilization, imbalance, and the number of parcels delivered by each infeed are plotted and analyzed.

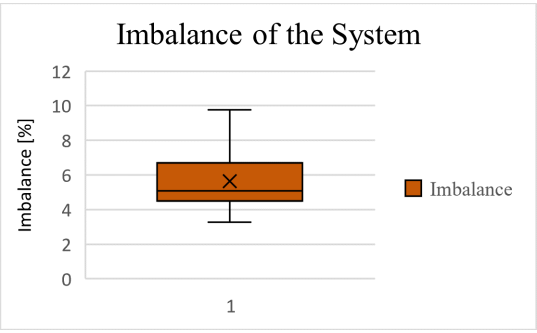
The analysis of the half-hour simulation data provides valuable insights into the performance of the parcel sorting system. By examining the box plots for key metrics including imbalance, throughput, utilization, and individual infeed throughput from Figure 5.2, we can gain a deeper understanding of the system's behaviour.



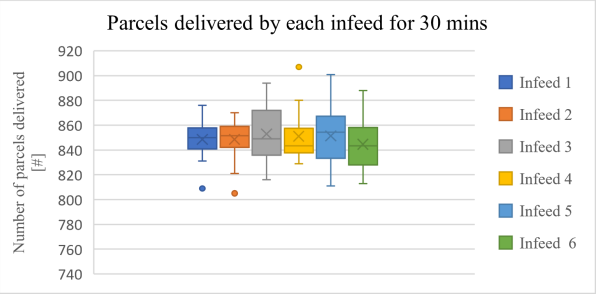
(a) Throughput of the system for 20 iterations



(b) Utilization of the system for 20 iterations



(c) Imbalance of the system for 20 iterations



(d) Individual infeed throughput for 20 iterations

Figure 5.2: KPIs for 30 minutes simulation run time

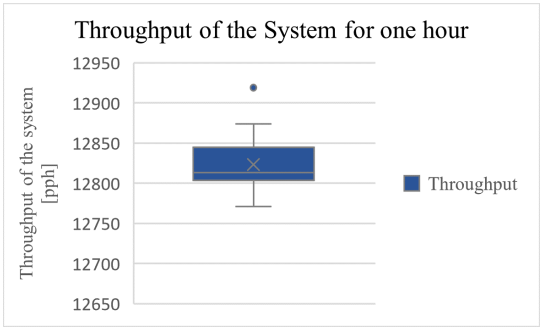
The results reveal that in terms of throughput, the merge conveyor demonstrates consistent performance, with a mean throughput of around 5990 parcels per half hour. The narrow range of throughput across iterations suggests stability in the system's overall performance. However, variations in individual infeed throughput indicate potential differences in parcel arrival rates or variations in the merging process. The system exhibits a relatively balanced distribution of parcels among the infeeds, with a median imbalance of approximately 5%. However, a couple of iterations display higher levels of imbalance, reaching up to 9.76%.

The system showcases efficient utilization, with a mean utilization value of approximately 95%. This indicates optimal usage of the available space on the merge conveyor. The relatively small range of utilization values among iterations further supports the system's consistent utilization performance. Examining the individual infeed throughput reveals variations among the infeeds. Certain infeeds consistently achieve higher throughput, while others demonstrate slightly lower throughput. These variations may be attributed to factors such as differences in parcel arrival rates or parcel dimensions even.

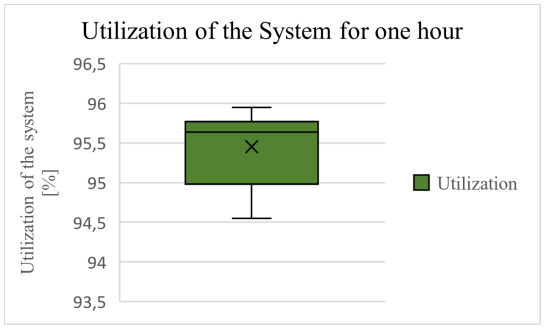
Simulation time: one hour

To gain a more comprehensive understanding of the algorithm's performance and its long-term sustainability, a simulation time of one hour was chosen. This extended duration allows for a deeper exploration of the system's behaviour and performance under various operational conditions. By running the simulation for a longer period, a larger number of events and interactions take place within the parcel sorting system, providing a more robust assessment of its operational characteristics. The one-hour simulation captures the dynamics of the system over an extended time frame, shedding light on any potential patterns, fluctuations, or trends that may emerge. The graphical representations obtained from the simulation provide valuable insights into the algorithm's ability to sustain optimal

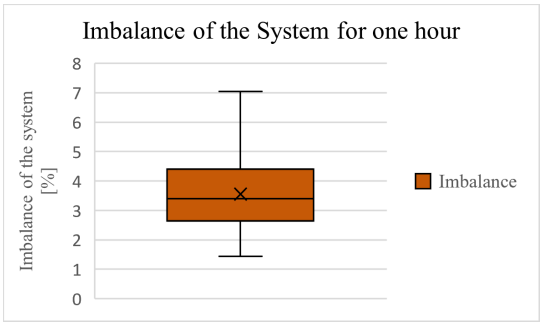
performance over a prolonged operational period.



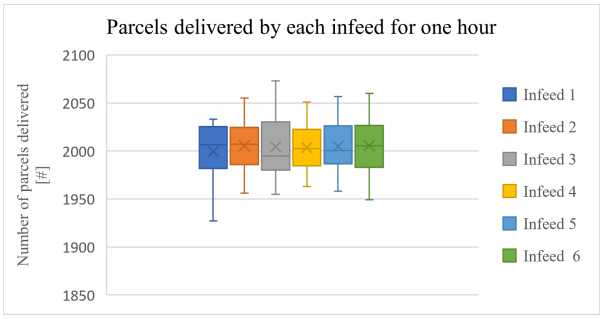
(a) Throughput of the system for 20 iterations



(b) Utilization of the system for 20 iterations



(c) Imbalance of the system for 20 iterations



(d) Individual infeed throughput for 20 iterations

Figure 5.3: one hour simulation time

The simulation results for a one-hour time period as depicted in Figure 5.3 provide valuable insights into the performance of the system. The average throughput, representing the number of parcels processed, remains relatively stable at around 12,800 parcels. This indicates that the system is able to handle a substantial volume of parcels within the given time frame. However, it is worth noting that there is some variation in throughput values across different iterations, suggesting occasional fluctuations in the system's performance.

The utilization of the merge conveyor, which measures the extent to which the system utilizes its capacity, exhibits a commendable average value of approximately 95.7%. This indicates that the system efficiently utilizes available space, minimizing empty gaps between parcels. Despite the overall high utilization, there are slight variations in utilization values throughout the simulation, suggesting some variability in the workload distribution among the infeed conveyors.

The imbalance in throughput of infeeds, which represents the disparity between the maximum and minimum throughput among the infeed conveyors, averages around 3.5%. This indicates a relatively balanced distribution of parcels among the infeeds, with no significant discrepancy in workload allocation. However, it is important to note that in a few instances, the imbalance exceeds 5%, indicating a slight deviation from perfect load balancing. This could be accounted for the reason of having parcel dimension variability. It can be possible that there might have been instances where one of the infeed had a parcel too long to be accommodated in the segment and had to wait for a very short duration thereby reducing its corresponding infeed's throughput, resulting in a higher imbalance.

Analyzing the individual throughput of each infeed conveyor, it is observed that the system achieves a relatively equitable distribution of workload among the infeeds. The average throughput for each infeed ranges from approximately 1,950 to 2,055 parcels, suggesting a balanced allocation of parcels across the different infeed conveyors.

After conducting simulations for both half an hour and one hour, it becomes evident that the KPIs tend to stabilize and maintain consistent values across multiple iterations, demonstrating their robustness for different simulation duration. Additionally, the throughput nearly doubled when the simulation duration was increased to one hour, and simultaneously, the imbalance decreased compared to the half-hour simulation. This improvement in imbalance can be attributed to the presence of a normal distribution in both parcel arrivals and parcel dimensions.

A normal distribution tends to concentrate the number of parcel arrivals and their dimensions around a central value over time (Altman et al. 1995). This characteristic ensures that the heap algorithm consistently receives at least two parcels in each infeed. The heap algorithm, which is employed for iterating and sorting the infeeds, utilizes a maximum filled queue technique to select the topmost element, remove a parcel from it, and update its state. This approach maintains a max heap structure, promoting a balanced flow from all the infeeds.

5.2. Comparison with the current algorithm

In this section, a comprehensive comparison is conducted between the newly developed algorithm and the existing algorithm currently utilized by Vanderlande. The focus of this evaluation is on KPIs including throughput, utilization and load imbalance. The primary objective is to thoroughly assess the performance of the proposed algorithm in these specific areas and identify its strengths and weaknesses when compared to the current industry-level algorithm. A moving average technique was employed to create a smoothed depiction of the data as suggested by Hyndman (2011), facilitating the identification of long-term trends or patterns by averaging values within the time frame of 3 minutes.

To ensure a realistic evaluation, various scenarios are meticulously designed to resemble a variety of operational conditions. These scenarios are broadly categorized into two groups: one without any fly-through parcels and the other with a 10% fly-through parcel occurrence rate, which closely mirrors real-world circumstances. Additionally, within these scenarios, different cases are carefully considered and will be further expounded upon in the subsequent sections. Although there can be several combinations possible to test the algorithm, the selected cases provide a holistic view of the algorithm's performance.

5.2.1. Scenario with no fly-through parcels

To ensure a comprehensive evaluation of the developed algorithm, various sub-cases have been identified based on discussions with industry experts. These sub-cases are designed to simulate different operational conditions within the context of a scenario without any fly-through parcels. The sub-cases are as follows:

Case 1: A happy flow from all six infeeds: This sub-case represents the ideal situation where all 6 infeeds are functioning smoothly, allowing for an optimal merging process.

Case 2: The first and the last infeed operating: Here, the evaluation centres on a situation where only the first and last infeeds are operational, while the middle infeeds are offline.

Case 3: Only the first four infeeds operate: In this sub-case, the system is configured to simulate a scenario where only the first four infeeds are operational, while the remaining 2 infeeds are temporarily out of service.

Case 4: Only the first two infeeds operate: This sub-case focuses on a scenario where only the first two infeeds are functioning, while the other infeeds are non-operational.

Although the second and the last case involves only two infeeds that are operating, their position plays a crucial role in the overall KPIs. The difference will be visible while discussing the results. To ensure a concise and focused explanation, this section will delve into the details of two specific cases. It is important to note that a comprehensive analysis of all cases can be found in Appendix C, providing an overview of the findings.

Case 1: A happy flow from all six infeeds

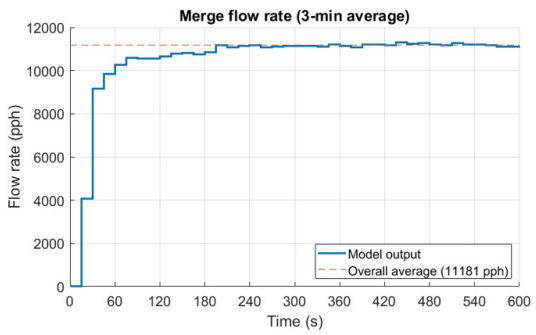
The purpose of this section is to assess the performance of the developed algorithm in comparison to the current industry standard under the specific scenario of a smooth flow of parcels from all 6 infeeds. This evaluation will provide valuable insights into the algorithm's effectiveness in optimizing the merging process when all infeeds are operational.

Table 5.1: No fly-through Happy Flow KPI comparison

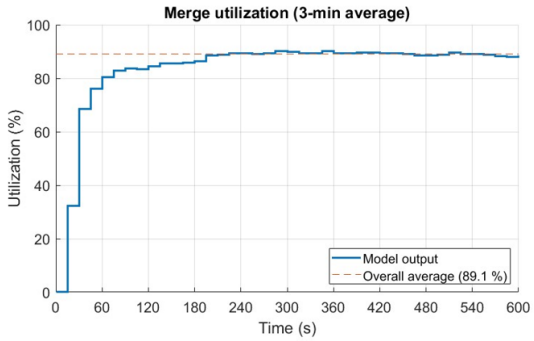
Scenario	Current Algorithm		Developed algorithm	
No fly-through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
Happy Flow	11181	89.1	12540	95.13

Current Control Algorithm

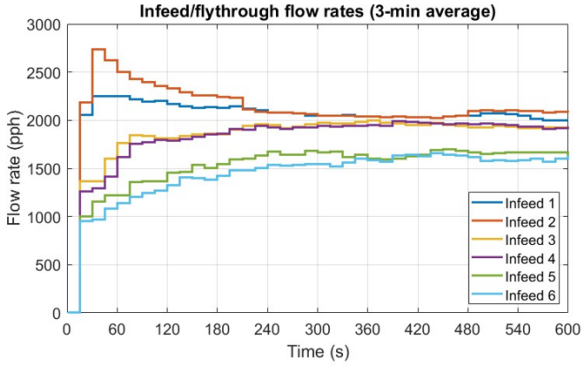
The results of the KPIs for the scenario without any fly-through parcels and a happy flow from all 6 infeeds are presented in the Table 5.1 and the graphs shown in Figure 5.4. It is important to note that the simulation was conducted using the same parameter settings with the fixed gap mode. The duration of the simulation was determined based on the time required to reach a plateau in the performance metrics. Since the current model stabilizes after a certain period of time, the simulation was extended for an additional three minutes to capture a more comprehensive view of the system's performance.



(a) 3-minute average of Throughput



(b) 3-minute average of Merge Utilization

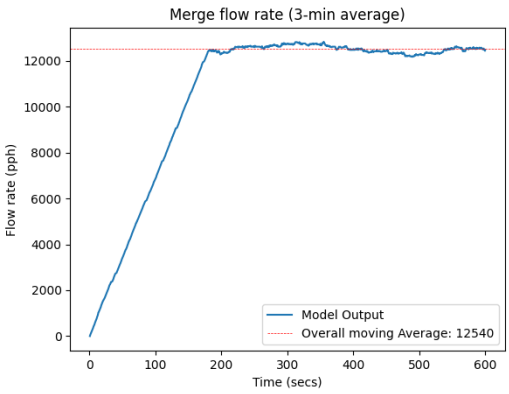


(c) 3-minute average of Infeed imbalance

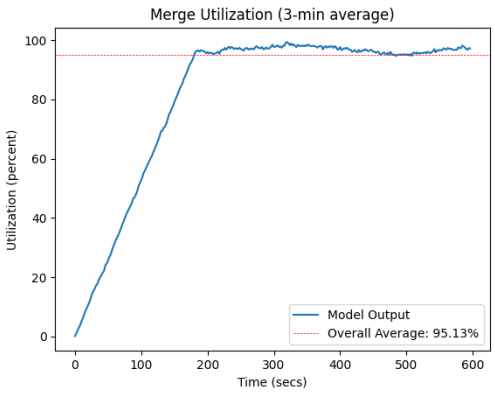
Figure 5.4: 3-minute average graphs of KPIs using current control algorithm

Developed DP-based Algorithm

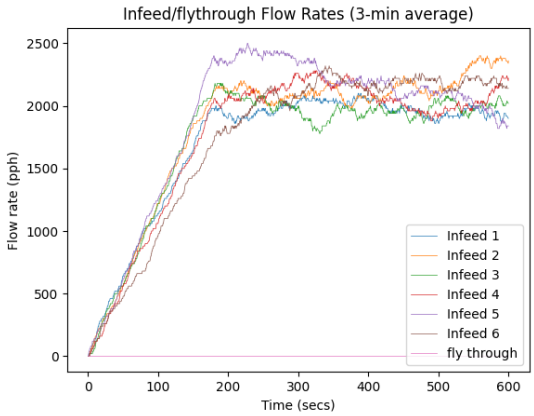
The graphs displayed in Figure 5.5 illustrate the 3-minute average trends of key performance indicators obtained using the developed control algorithm.



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization



(c) 3-minute average of Infeed imbalance

Figure 5.5: 3-minute average graphs of KPIs using developed algorithm

Comparing the performance of the current algorithm and the developed algorithm in a scenario with a smooth flow of parcels from all six infeeds, several conclusions can be drawn.

Firstly, it is evident that the developed algorithm outperforms the current industry standard in terms of throughput. The current algorithm achieves an average throughput of 11,181 parcels per hour, while the developed algorithm significantly improves this metric with a throughput of 12,540 parcels per hour. This indicates that the developed algorithm is more efficient in processing and merging parcels, resulting in higher overall throughput.

Secondly, the utilization of the system is substantially enhanced by the developed algorithm. The current algorithm achieves a utilization rate of 89.1%, whereas the developed algorithm demonstrates a remarkable improvement with a utilization rate of 95.13%. This indicates that the developed algorithm better utilizes the available resources, minimizing empty gaps between parcels and maximizing the system's capacity. Finally, from Figure 5.4c and Figure 5.5c, it can be seen that all the infeeds have more or less the same throughput in the developed algorithm when compared to the industry-level algorithm.

Case 2: The first and last infeed operating

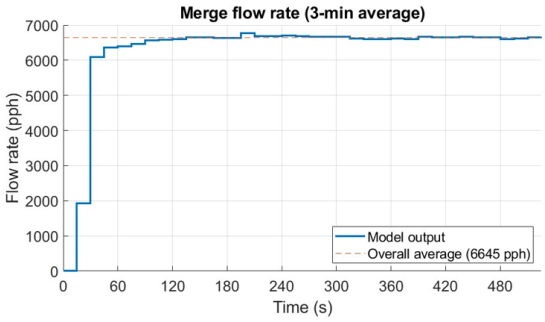
This section specifically focuses on a particular scenario in which only the first and last infeed conveyors are operational. By examining this case, valuable insights can be obtained regarding the performance of the developed algorithm compared to the current industry standard under these specific operational conditions. It is worth noting that although the infeeds considered are labelled as infeed 1 and infeed 2 in the legends of the graphs presented in Figure 5.6c and Figure 5.7c, they actually represent infeed 1 and infeed 6. This labelling discrepancy arises from the fact that, in the simulation, these infeeds are modelled as queues, resulting in their representation as infeed 1 and infeed 2. However, it is important to clarify that their original designations correspond to infeed 1 and infeed 6.

Table 5.2: No fly-through infeed 1 and 6 operating

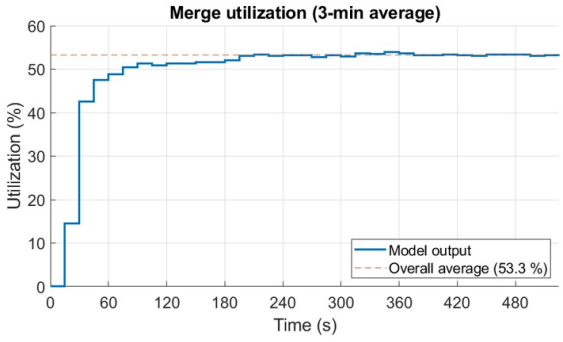
Scenario	**Current Algorithm**		**Developed algorithm**	
No fly-through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
Infeed 1and Infeed 6	6645	53.3	6929	54.31

Current Control Algorithm

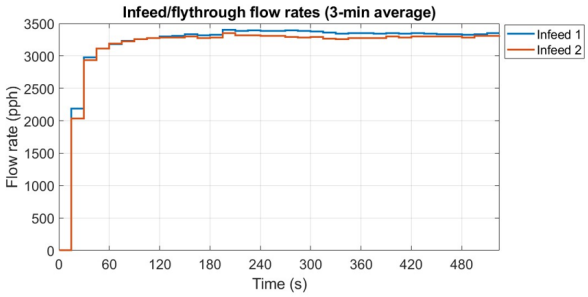
This section provides an assessment of the performance of the current control algorithm utilized in the merging process. The graphs shown in Figure 5.6 present the 3-minute average graphs depicting KPIs for the current case of first and last infeed working. The analysis of these graphs offers insights into the behaviour of the current algorithm in managing the merging process. The obtained results serve as a reference point for comparison with the developed DP-based algorithm



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization

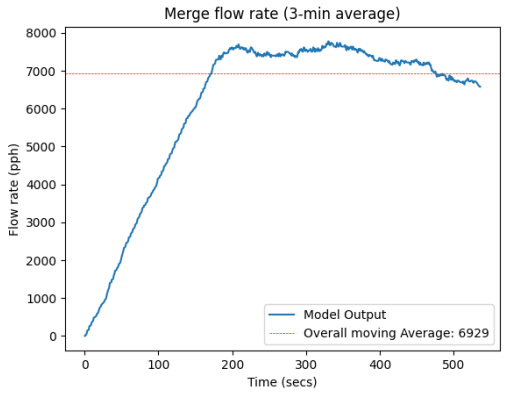


(c) 3-minute average of Infeed imbalance

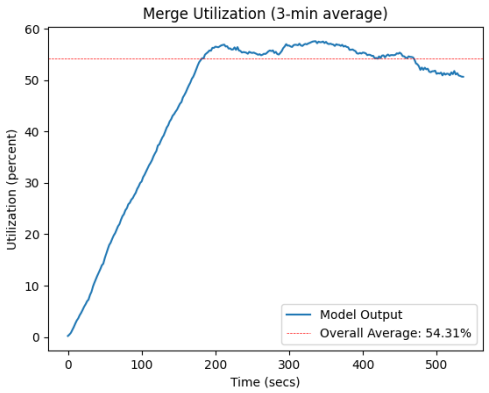
Figure 5.6: 3-minute average graphs of KPIs using current control algorithm

Developed DP-based Algorithm

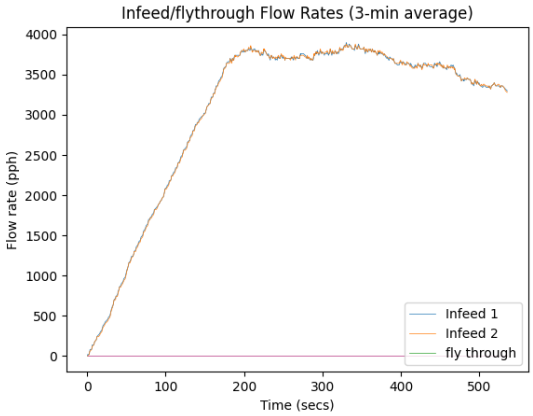
The graphs displayed in Figure 5.7 illustrate the 3-minute average trends of key performance indicators obtained using the developed control algorithm.



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization



(c) 3-minute average of Infeed imbalance

Figure 5.7: 3-minute average graphs of KPIs using developed control algorithm

The analysis of the specific situation, where only the first and last infeed conveyors are operational, provides valuable insights into the comparative performance of the developed algorithm against the current industry standard. From the comparison in Table 5.2, it is evident that in this particular case, the developed algorithm exhibits superior performance compared to the current algorithm. Specifically, the developed algorithm achieves a significantly higher average throughput of 6,929 parcels per hour, surpassing the average throughput of 6,645 parcels per hour achieved by the current algorithm.

Furthermore, the utilization improvement achieved by the developed algorithm, while modest in magnitude, still demonstrates its efficacy compared to the current industry standard. With a utilization rate of 54.31% for the developed algorithm compared to 53.3% for the current algorithm, even a relatively small difference can have a significant impact in practice. This marginal increase in utilization indicates that the developed algorithm optimizes the use of available space, reducing empty spaces and improving the overall efficiency of the merge conveyor. While the difference may seem subtle, it is indicative of the algorithm's ability to make incremental enhancements in system performance, which can accumulate and yield notable improvements over time. Thus, the developed algorithm shows promising potential in achieving more efficient utilization, thereby contributing to the overall optimization of the parcel transport system.

5.2.2. Scenario with 10 per cent fly-through parcels

Building upon the subsection 5.2.1 that focused on scenarios without fly-through parcels, this section delves into the specific case of the 10% fly-through parcel scenario. By 10% fly-through parcels, it is meant that in 10 parcels of the parcel generation, there is one fly-through parcel. Within this scenario, further sub-cases are carefully considered to provide a more nuanced analysis. The sub-cases include:

Case 1: All six infeeds are operational

Case 2: Only the first two infeeds are operational

Case 3: Only four infeeds are operational

The primary objective of this section is to demonstrate how the developed algorithm performs in comparison to the current industry standard within the context of the 10% fly-through parcel scenario.

By examining the results obtained from these sub-cases, we can gain insights into the algorithm's efficacy and its ability to handle different operational configurations.

Case 1: All the six infeeds are operational

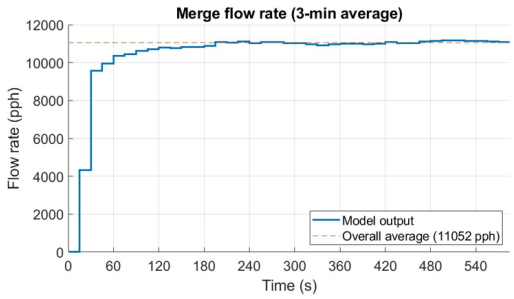
In this section, the performance of the developed algorithm is compared with the current control algorithm in a scenario where all six infeeds are operational and a 10% fly-through parcel occurrence rate is present.

Table 5.3: All the six infeeds in operational condition in the presence of a 10% fly-through parcel occurrence rate

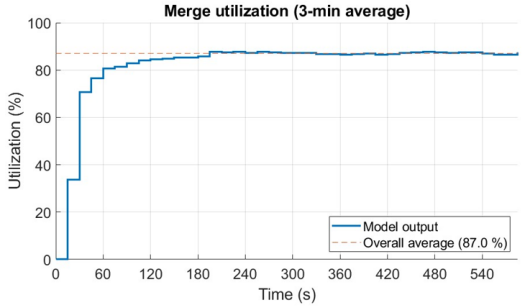
Scenario	Current Algorithm		Developed algorithm	
10% fly-through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
6 infeeds	11052	87	12597	95.11

Current Control Algorithm

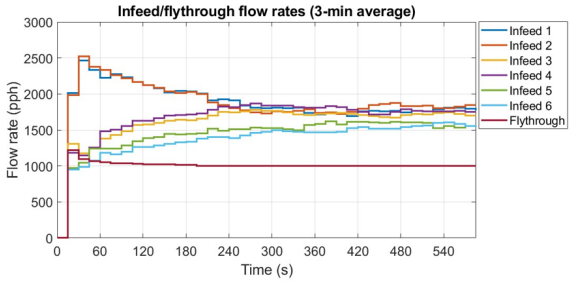
The graphs presented in Figure 5.8 depict the three-minute average patterns of important performance metrics, obtained through the utilization of the current industry-level control algorithm.



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization

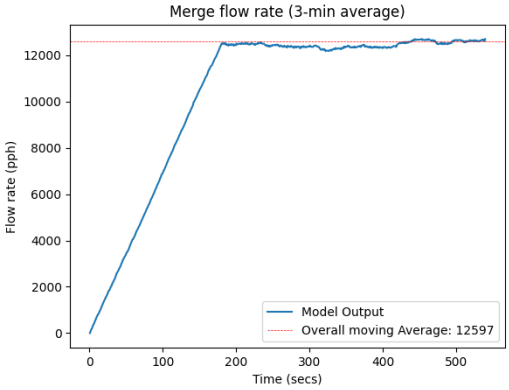


(c) 3-minute average of Infeed imbalance

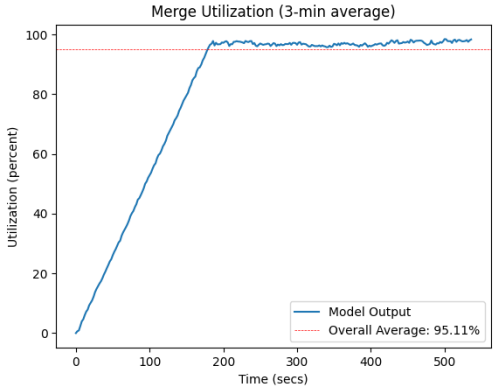
Figure 5.8: 3-minute average graphs of KPIs using current control algorithm

Developed DP-based Algorithm

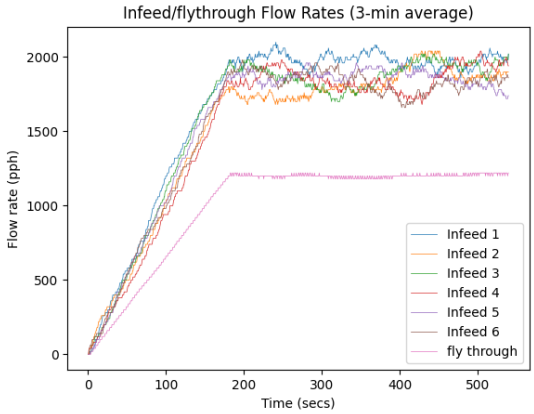
The graphs displayed in Figure 5.9 illustrate the 3-minute average trends of key performance indicators obtained using the developed control algorithm.



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization



(c) 3-minute average of Infeed imbalance

Figure 5.9: 3-minute average graphs of KPIs using developed control algorithm

The results obtained from the comparison provide valuable insights into the performance of the two algorithms in terms of throughput and utilization.

The analysis of the results revealed that the developed algorithm exhibits notable advantages over the current algorithm. In terms of throughput, the developed algorithm achieved a higher value of 12,597 parcels compared to the current algorithm's throughput of 11,052 parcels. This indicates that the developed algorithm is capable of efficiently processing and handling a larger volume of parcels within the given time frame.

Furthermore, the utilization of the merge conveyor was significantly improved with the implementation of the developed algorithm. It achieved a utilization rate of 95.11%, surpassing the utilization rate of 87% achieved by the current algorithm. This enhancement in utilization suggests that the developed algorithm optimizes the usage of available space on the merge conveyor. The imbalance in this is

more or less similar for both the algorithms, except for the fact that the upstream infeeds have a higher throughput at the beginning of the simulation (Figure 5.8c) for the current algorithm when compared to the developed algorithm.

Case 2: Only the first two infeeds are operational

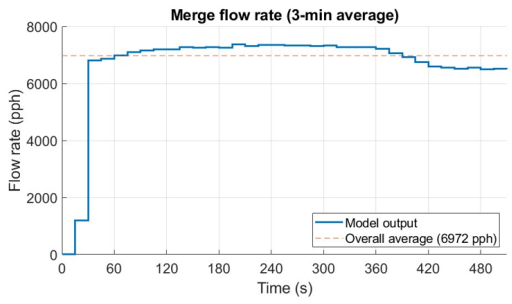
In the following section, the performance of the developed algorithm is assessed through a comparison with the current control algorithm in a specific case. This case involves the operation of the first two infeeds and a 10% occurrence rate of fly-through parcels. The primary goal is to evaluate how the developed algorithm performs relative to the current control algorithm under these specific conditions.

Table 5.4: The first two infeeds in operational condition in the presence of a 10% fly-through parcel occurrence rate

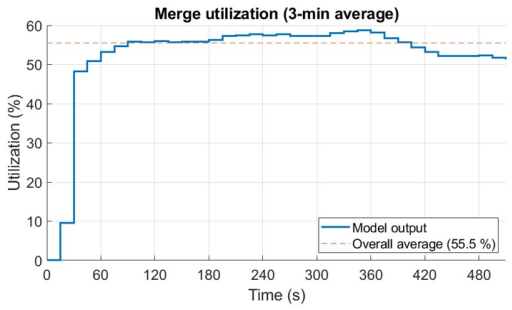
Scenario	Current Algorithm		Developed algorithm	
10% Fly through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
2 infeeds	6972	55.5	8502	65.7

Current Control Algorithm

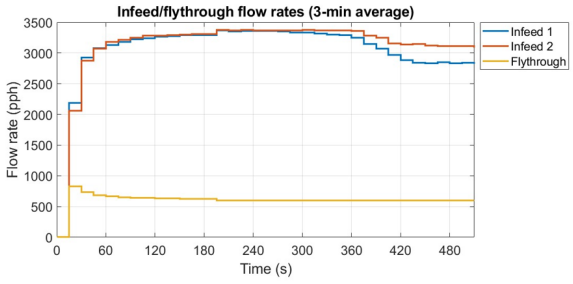
The graphs presented in Figure 5.10 depict the three-minute average patterns of important performance metrics, obtained through the utilization of the current industry-level control algorithm.



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization

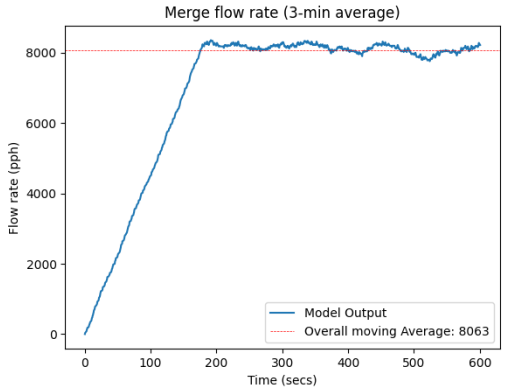


(c) 3-minute average of Infeed imbalance

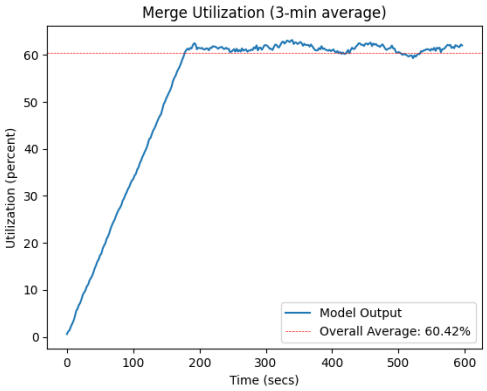
Figure 5.10: 3-minute average graphs of KPIs using current control algorithm

Developed DP-based Algorithm

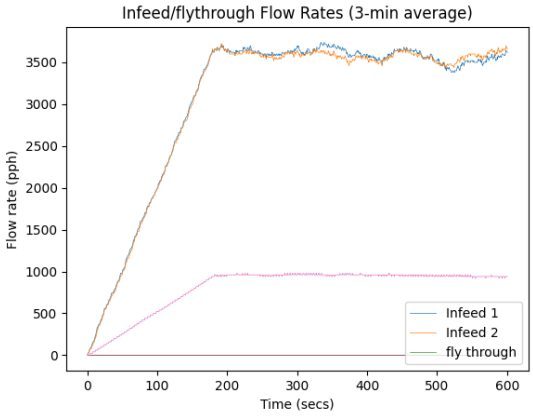
The graphs displayed in Figure 5.11 illustrate the 3-minute average trends of key performance indicators obtained using the developed control algorithm.



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization



(c) 3-minute average of Infeed imbalance

Figure 5.11: 3-minute average graphs of KPIs using developed control algorithm

Examining the Figure 5.10, Figure 5.11 and the Table 5.4 reveals notable differences between the two algorithms in terms of throughput and utilization. The current algorithm exhibits a throughput of 6,972 parcels per hour with a corresponding utilization of 55.5%. In contrast, the developed algorithm showcases improved performance, achieving a higher throughput of 8,063 parcels per hour, accompanied by a utilization rate of 60.42%. These results clearly indicate that the developed algorithm surpasses the current algorithm in terms of both throughput and utilization in the scenario where only the first two infeeds are operational while considering a 10% fly-through parcel occurrence. Furthermore, the graph from Figure 5.10c reveals that the current algorithm takes a while to achieve a state of proper balance among the infeeds in the current algorithm which is not the case for the developed algorithm as depicted in Figure 5.11c.

The observed significant increase in throughput can be attributed to the efficient control mechanism implemented by the developed algorithm. By optimizing the merging process, the algorithm ensures smoother and more streamlined parcel movement. Moreover, the higher utilization achieved by the developed algorithm underscores its ability to efficiently utilize available space on the merge conveyor, resulting in improved overall efficiency and reduced empty spaces between parcels.

5.3. Reflecting on the comparison

The outcomes of all cases are presented in Table 5.5 and Table 5.6. Additional graphs illustrating the remaining cases can be found in Appendix C, which collectively provide an extensive analysis of the algorithm's performance across different cases. The results evidently indicate that the developed algorithm has, to a significant extent, surpassed the industry-level algorithm in terms of both utilization and throughput. This notable improvement can be attributed to the careful planning that the algorithm employs for parcel handling.

Firstly, in the algorithm that is being used in the industry, as discussed in section 2.3, when a parcel is detected and announced by the infeed controller, the search algorithm of the merge controller actively seeks available space along the length of the merge conveyor, preceding the parcel's designated infeed location. Consequently, this approach leads to a considerable amount of unused space, thereby diminishing the overall throughput. On the contrary, the DP-based algorithm adopts a more efficient strategy by employing a fixed segment size that starts before the merge conveyor. During the segment generation event, the algorithm strives to utilize the available parcels while also considering the fixed gaps to fill the segment, thus effectively reducing the problem of unused gaps and resulting in better utilization and throughput.

Secondly, within the context of the no fly-through scenario, the discussion pertained to various cases, and a notable observation was made concerning the performance of Infeed 1 and 6 compared to the first two infeeds. While these two cases may appear similar, their outcomes exhibit notable distinctions. The data presented in Table 5.5 reveals that the case with the first two infeeds in operation demonstrates a higher throughput when compared to the scenario where infeeds 1 and 6 are active.

The underlying reason behind this discrepancy lies in the operational dynamics of the two cases. In the situation involving infeeds 1 and 6, the primary factor leading to the lower throughput is the waiting times experienced at infeed 6. Due to the parcel processing sequence, only after a parcel has exited infeed 6 can another parcel be announced and processed, which inevitably slows down the overall throughput. On the other hand, in the case where infeeds 1 and 2 are operational, the lower throughput can be attributed to a different cause. In this case, both infeeds continuously deliver parcels one after the other. However, the arrival rate of parcels is comparatively lower, resulting in reduced throughput and an increase in empty spaces within the system.

Finally, the graphs for the DP-based algorithm demonstrate a consistent linear growth until a specific simulation time, at which point they stabilize. This phenomenon is attributed to the inherent statistical nature of parcel arrival rate distribution. Notably, the linear progression is not observable in the industry's algorithm, as their graphing process commences only after the KPIs'average stabilises. Conversely, the DP-based algorithm plots the graphs from the onset of simulation time, resulting in a linear increase within a specific interval, as opposed to an abrupt transition.

Table 5.5: No fly-through all cases

Scenario	Current Algorithm		Developed algorithm	
No Fly-through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
6 infeeds	11181	89.1	12540	95.13
Infeed 1 and Infeed 6	6645	53.3	6929	54.31
4 infeeds	10621	84.3	11493	88.68
2 infeeds	6738	53.8	7074	56.54

Table 5.6: 10% fly-through all cases

Scenario	Current Algorithm		Developed algorithm	
10% Fly-through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
6 infeeds	11052	87	12597	95.11
2 infeeds	6972	55.5	8063	60.42
4 infeeds	10084	78.6	11977	89.28

However, the development of an algorithm alone does not guarantee its seamless implementation in real-time operations. Several additional factors come into play when considering the replacement of the current algorithm in practical applications. One significant factor is the cost implications associated with implementing the developed algorithm. Companies are likely to invest in replacing the existing algorithm only if they can reap sufficient benefits from the proposed solution. Therefore, the subsequent section will present an overview of the cost-benefit analysis, shedding light on the economic considerations and potential advantages of adopting the developed algorithm.

5.4. Cost Benefit Analysis

According to Mishan et al. (2020), Cost-benefit analysis (CBA) is a powerful decision-making tool widely used in various fields to evaluate the economic feasibility and efficiency of potential projects, policies, or interventions. With its roots in welfare economics, CBA provides a systematic framework for assessing the costs and benefits associated with different alternatives quantitatively and objectively. By considering both monetary and non-monetary factors, such as social and environmental impacts, CBA enables decision-makers to make informed judgments and prioritize resource allocation. Its application spans diverse sectors, including public policy, infrastructure development, environmental management, and healthcare, among others. The primary objective of CBA is to compare the costs and benefits of different options consistently and transparently. By monetizing both positive and negative impacts, CBA enables decision-makers to assess whether the benefits of a project or policy outweigh its costs. This analysis goes beyond the traditional financial evaluation by incorporating a broader range of factors that contribute to societal welfare and well-being.

In the material handling industry, where the adoption of new software solutions is not frequent, the decision to develop a new algorithm or software requires careful cost-benefit analysis. Conducting such an analysis is crucial as it allows companies to weigh the costs associated with the development and testing phases against the anticipated benefits and performance improvements (Erdogmus 2007). This systematic evaluation empowers decision-makers to make informed judgments and effectively allocate resources.

The foremost requirement for any newly developed algorithm is reliability and robustness. According to Azar et al. (2021), reliability refers to the capacity of an algorithm to demonstrate consistent and precise performance across diverse conditions. A reliable algorithm exhibits the capability to generate consistent outcomes consistently over time and across various environments. On the contrary, robustness characterizes the ability of a control algorithm to deliver optimal performance despite the presence of disturbances or uncertainties. A robust algorithm is adept at sustaining its performance even when the controlled system experiences alterations or fluctuations. It must perform consistently and efficiently in real-world scenarios, ensuring smooth operations in material handling systems. Additionally, compatibility with the hardware adopted by customers is crucial. For merge control algorithms, the hardware commonly used is Programmable Logic Controllers (PLCs), which are manufactured by industry-leading companies such as Siemens and others (B. J. Kim et al. 1991). Therefore, any new software being developed needs to seamlessly integrate with these widely used hardware systems.

Undertaking the development of a new algorithm or model incurs costs, both in terms of time and resources (Arm et al. 2018). Based on discussions with an expert from Vanderlande, a lead time of 6 months is considered more than sufficient to successfully develop and test a new algorithm. According to Goldratt et al. (2004), the lead time represents the duration from the initiation of the development process to the completion of the testing phase. During this period, a team of two highly skilled personnel is typically required to handle the development phase efficiently. To further illustrate the associated costs, let us consider an example. Assuming a lead time of 4 months, with two personnel working on the development phase and one or two technicians involved in the testing phase, the estimated operational costs associated just with the employee salary would be as follows:

$$\begin{aligned}\text{Operational costs} &= 4\ (months) * 4(week/month) * 40\ (hrs/week) * €100\ (/hour) * 2\ employees \\ &= €128,000\end{aligned}$$

In addition to these costs, there are additional operational expenses related to technicians and setting up the hardware to test and validate the software, which can be approximated to 80,000 euros. Therefore, the total investment required to fully develop a new model and make it operational would amount close to 200,000 euros. However, it is important to note that this example represents an ideal scenario with average pay rates, and if the lead time exceeds the projected duration, it would result in additional costs.

Justifying such a significant investment in a new model necessitates demonstrating a performance improvement of at least 5 per cent when compared to the current model. This benchmark serves as a minimum threshold to ensure that the investment is worthwhile. Achieving this performance improvement can yield several benefits for the company. Firstly, it provides a competitive edge over other market players, allowing the company to differentiate itself and attract more customers. Secondly, improved performance can lead to higher turnover rates and increased revenue, as clients value efficient and reliable material handling systems (Shaw 2011). While the immediate profit may not be significant, the expanded market reach resulting from improved performance can contribute to long-term growth and success.

Additionally, such performance improvements can enable not only the introduction of a new system but also upgrades to the existing system at clients' ends. This allows the company to continuously enhance their solutions and adapt to changing customer needs and industry trends. In the dynamic material handling industry, striving for higher performance is always a significant goal for companies

to maintain their competitiveness. However, the company believes that continuously improving the current system is more practical and beneficial in the long run, as opposed to developing a new algorithm or system in every cycle. This approach ensures that the company can maximize the value and potential of their existing software while staying responsive to customer requirements and industry advancements.

Based on the comprehensive comparison presented in section 5.2, it is evident that the developed algorithm consistently outperforms the current industry-level algorithm in multiple cases. The KPIs analyzed in the evaluation process demonstrate significant improvements when the developed algorithm is employed. These findings highlight the potential value of implementing the developed algorithm, particularly when the aim is to enhance the overall efficiency of the process. The demonstrated superiority of the developed algorithm, as indicated by the favourable KPI results, suggests that investing in the adoption and integration of this algorithm could yield substantial benefits and contribute to the optimization of the system's operations.

5.5. Conclusion

This chapter contributes to addressing the fourth and fifth sub-research questions by conducting a comprehensive evaluation of the developed algorithm for parcel sorting systems. The evaluation is carried out through model experimentation, with a particular focus on the simulation time as a crucial factor in accurately assessing system behaviour and performance. Simulations were conducted for a duration of 30 minutes and one hour to gain insights into the algorithm's robustness, stability, and efficiency over extended periods.

The simulation results consistently demonstrated the superior performance of the developed algorithm compared to the current industry-level algorithm in various scenarios. The analysis of key performance indicators revealed significant improvements when employing the developed algorithm. Specifically, the developed algorithm achieved higher throughput and utilization rates, showcasing its effectiveness in optimizing the merging process and maximizing resource utilization. These findings were consistent across scenarios with and without fly-through parcels, further confirming the algorithm's superiority under challenging operational conditions.

Furthermore, a cost-benefit analysis was conducted to assess the economic feasibility of implementing the developed algorithm, addressing the fifth research question. The analysis considered factors such as reliability, robustness, compatibility with hardware systems, and associated costs. The estimated investment required for developing and implementing the algorithm was weighed against a benchmark of at least 5% performance improvement to justify the investment. The developed algorithm surpassed this benchmark and demonstrated its potential for real-world implementation, warranting further research and investment.

In conclusion, the results of the model experimentation provide strong evidence of the developed algorithm's superior performance in terms of throughput and utilization. The findings emphasize the potential benefits and efficiency gains that can be achieved by adopting the developed algorithm in real-world parcel sorting systems.

6

Conclusion

The objective of this thesis was to develop a control algorithm with the potential to enhance the utilization of a merge conveyor in a line sorter sortation system. A conceptual design and model were developed and assessed using discrete event simulation. This chapter primarily focuses on answering the research questions and drawing conclusions based on the findings of this study. Additionally, the chapter concludes by presenting remarks and recommendations for future research in this area.

6.1. Conclusion: Answering the research questions

This section answers the research questions as defined in subsection 1.3.1. A detailed answer to each of the sub-research questions will help to conclude with the answer to the main research question.

(1) Which factors significantly impact the parcel merging process and the utilization of the merge conveyor in a sorting system?
In chapter 2, an in-depth exploration has been conducted to examine the factors that significantly impact the parcel merging process and the utilization of the merge conveyor in a sorting system. By analyzing these factors, valuable insights have been gained into their significance and implications.

The parcel merging process holds a critical role in ensuring seamless transportation and efficient merging within the sorting system. One of the crucial factors that influence this process is the velocity profiles of parcels, as discussed in subsection 2.1.1. Consideration of parcel compatibility in terms of size and shape is essential for smooth transportation and integration within the system. The presence of variable parcel dimensions can introduce challenges related to irregular spacing and slice allocation, highlighting the need to address this factor.

Another significant factor is the timing of parcel announcements on the infeeds. Proper coordination is crucial, as it directly affects the merge controller's search for available space once the infeed controller announces a parcel's arrival. Moreover, the speed at which the merge operation is performed plays a vital role. Striking a balance between the utilization of the merge conveyor and achieving load balancing among the infeeds presents a trade-off that necessitates careful consideration. While maximizing utilization might be tempting, neglecting load balancing can result in suboptimal system performance. Thus, finding an optimal balance between these objectives is also crucial.

(2) What existing approaches for similar problems suggest a suitable control algorithm for the current issue of low utilization?
The literature study concluded in chapter 3, focused on identifying an optimal control algorithm to improve the merge conveyor utilization. Emphasis was placed on implementing effective control mechanisms to enhance merge zone efficiency. Extensive exploration of various control algorithms for parcel and vehicle merging processes was conducted to enhance overall system performance.

Through the literature review, a notable research gap emerged in the utilization of optimization techniques and alternative scheduling approaches in the context of parcel industries. This highlighted the need to investigate methods that had the potential to enhance the parcel merging process—a critical aspect with significant implications for system throughput and utilization. While exact algorithms like Integer Linear Programming were utilized, their inherent time-consuming nature prompted the search for more efficient alternatives. One promising approach that demonstrated success in vehicle lane merging was dynamic programming. However, its application specifically in the field of parcel merging had yet to be explored comprehensively.

The incorporation of dynamic programming into the parcel merging process held promising prospects for significant improvements in system performance and throughput. Drawing insights from related areas, dynamic programming offered a potential solution to address the challenges and complexities associated with parcel merging, ultimately leading to enhanced operational efficiency and increased customer satisfaction in the parcel industry.

(3) How can the selected control algorithm be developed for the Line-Sorter sortation system?
This sub-research question is answered in the chapter 4. To address the question of developing the selected control algorithm for the Line-Sorter sortation system, a comprehensive approach was undertaken in this research. By leveraging the capabilities of Discrete Event Simulation, a simulation model was meticulously constructed to replicate the merging process within the Line-Sorter system. The utilization of DES was motivated by its inherent advantages, including its adaptability in representing systems of varying complexities and its ability to accurately capture the dynamic behaviour of the system over time.

Python, a versatile programming language widely recognized for its extensive documentation, thriving user community, and an array of simulation libraries, was chosen as the preferred software implementation platform for DES. The decision to adopt Python was driven by its seamless integration with various scientific and data analysis tools, making it an increasingly popular choice among researchers and practitioners. Furthermore, the open-source nature of Python, coupled with its cost-effectiveness compared to commercial software packages, further solidified its appeal for this study.

Within the developed simulation model, a virtual segment aligned with the merge conveyor was introduced, initiating before the actual merge conveyor's start. This virtual segment is the event trigger. It emulated the movement of the merge conveyor itself, synchronizing their speeds accordingly. In order to achieve optimal parcel sequencing, a control algorithm rooted in dynamic programming was meticulously designed. Recognizing the potential challenge posed by higher imbalances, it became imperative to incorporate a balancing technique capable of maintaining a desirable load balance among the various infeeds. To this end, a maximum heap sort algorithm was employed, intelligently sorting the infeeds based on the maximum filled queue technique. To evaluate the performance of the control algorithm, different KPIs identified in the chapter 3 were employed.

The developed simulation model underwent a series of rigorous verification and validation procedures, ensuring its precision, dependability, and appropriateness for conducting experiments. The outcomes of the sensitivity analysis revealed a clear relationship: when fly-through parcels were present, a smaller segment size led to increased utilization and improved throughput. It is essential to acknowledge that the model was constructed while adhering to a predefined set of constraints and assumptions, with the aim of closely aligning with the real-world conditions of the Line-Sorter sortation system. However, it is important to emphasize that the model's scope was deliberately limited to the merging process exclusively, neglecting the comprehensive modelling of the sortation zone and the overflow zone.

(4) How does the selected control algorithm perform compared to the current algorithm that is being used in the industry?
In order to assess the performance of the developed control algorithm in comparison to the industry-level algorithm, a comparative analysis was conducted in section 5.2 using the previously defined KPIs. Various scenarios were executed under consistent parameter settings to facilitate a comprehensive evaluation. The findings clearly indicate that the developed algorithm has a better performance when compared to the current control algorithm, as demonstrated by the comparative results depicted in the section 5.2.

It is important to note that while the developed algorithm exhibits advantages over the current algorithm within the existing layout, certain limitations were identified during the evaluation process. These limitations will be elaborated in the section 6.2 of this chapter. Nonetheless, based on the results and considering the current circumstances, the main conclusion drawn is that the developed algorithm holds a competitive edge over the current algorithm.

(5) What are the implications of implementing the developed control algorithm in terms of their impact on the costs for Vanderlande?
Merely developing an algorithm and claiming its superiority over the existing one would be an inadequate solution, as any changes implemented in a system typically come with associated costs in the real world. Therefore, it is imperative to conduct a cost-benefit analysis to evaluate the feasibility of further investment and development of the developed algorithm for real-time applications. While the developed model closely aligns with real-world scenarios, certain assumptions made, such as parcel slip or parcel orientation, may have an impact on the overall key performance indicators. Consequently, it becomes essential to substantiate that the algorithm's benefits justify the allocation of resources and funding.

Consultations with an expert from Vanderlande revealed that companies generally invest in researching and developing new algorithms only when the improvements achieved are at least five per cent greater than the current one in use. Such a significant improvement offers various advantages. Firstly, it establishes a competitive edge over other market players, attracting new customers and consequently increasing the company's revenue. Secondly, it enhances customer satisfaction and enables the adaptation of solutions to clients' existing systems. Given that the current model has exhibited promising results and superior performance, it represents a potential option for further research and development. Considering an estimated investment of approximately €200, 000, which can attract new customers and boost revenue, the return on investment (ROI) appears to be significantly higher.

Based on the results obtained for the developed model, the evidence presented in section 5.2 unequivocally demonstrated its superior performance. The findings revealed that the developed model surpassed the current algorithm, exhibiting KPI values that were at least five per cent higher. This

outcome represents a significant breakthrough, signifying that continued research on this algorithm has the potential to bring about substantial changes and advantages to both the system and the company.

A cumulative of all the five aforementioned sub-research questions can help in answering the main research question of:

"How can the parcel merging process and utilization of the merge conveyor in a line-sorter sortation system be improved by introducing a new control algorithm?"

By populating the imaginary segment with parcels based on available gaps during the event of segment generation, rather than searching and assigning a slice each time a parcel enters the system, it can be inferred that the utilization of dynamic programming as a control algorithm leads to superior utilization compared to other techniques. Throughout this thesis, the importance of striking a balance between utilization and load balancing has been emphasized. To ensure balanced loads among the infeeds, a max heap algorithm is employed that sorts the infeeds based on the maximum filled queue. This thesis represents an initial exploration of the implementation of dynamic programming as a control algorithm aimed at enhancing merge conveyor utilization.

6.2. Discussion and Recommendations

This section discusses the results and methodology employed in the thesis. Firstly, it is examined whether the chosen method adequately addresses the main research question. Based on the preceding information, it can be concluded that introducing a DP-based control algorithm can enhance the overall utilization of the system. The primary focus of the thesis was to identify a new control algorithm, but it could have also explored the potential for improving utilization with the existing industry-level algorithm in different layouts.

Several assumptions were made in the thesis, which could impact the obtained results. In practical scenarios, there are multiple variables at play beyond the ones considered in this thesis. For instance, the velocity profiles and the time it takes for a parcel to reach the merge were abstracted from real-life conditions to isolate the effect of velocity profiles and maintain a realistic approximation. Additionally, the current model was developed entirely using Python software. However, using different software might yield similar or different results, and this aspect needs to be verified.

The current model utilizes the maximum filled queue technique to prioritize the infeeds. However, in real-time situations, this assumption may not always hold true, as there could be an infeed with consistently higher capacity, making it the most filled queue at all times. Consequently, employing a different priority-based technique may yield slightly different throughput outcomes. Moreover, it is crucial to consider that in real-world scenarios, the capacity of infeeds can dynamically vary, while the current model assumes a constant capacity of 3200 parcels per hour for every infeed. Therefore, conducting additional experiments is necessary to evaluate the algorithm's behaviour in more realistic scenarios.

Regarding the positioning of the infeeds further downstream in the system, the assigned slice on the segment would have to travel a longer distance to reach the most downstream infeed. Consequently, parcels on the downstream infeeds may experience extended waiting times. Case 2 from subsection 5.2.1 demonstrates that the utilization remains relatively stable. Nevertheless, if infeed 6 were positioned even further downstream in the given layout under the same working conditions, the

utilization of the developed algorithm could experience a marginal decrease. Although the overall decline in utilization would not be significant in the presence of all the infeeds, a slight decrease can be expected in this specific scenario. Furthermore, if the velocities of the infeed conveyors are lower than the current situation, parcels may require more time to reach the merge conveyor, potentially resulting in decreased overall utilization and throughput.

In the context of parcels, the current research assumes a normal distribution for parcel dimensions, which may not accurately reflect real-time conditions. Furthermore, the weight of parcels plays a crucial role, as parcels can topple based on their centre of gravity, which is mass-dependent. Other kinematic constraints, such as slip, were not considered in the current research. It is worth noting that these factors can also impact the overall key performance indicators.

To further increase the utilization, similar to related research on parcel sorting system merging processes, it was evident that early knowledge of the next parcel for each infeed, i.e., parcel announcement, can provide the controller with more time to allocate a slice, regardless of the control algorithm being used. It is important to note that the developed model was designed based on a fixed gap mode between the parcels on the merge conveyor. Examining its performance with different gap modes could yield different results, and such variations should be thoroughly examined. It is recommended that Vanderlande conducts tests and analyzes the system's behaviour in light of these considerations.

Regarding costs, the cost-benefit analysis conducted in this research was based on approximate values provided by an expert from Vanderlande. However, it is important to consider that additional costs may have been overlooked. Additionally, while the control algorithm developed works well for a line sorter, it is recommended that Vanderlande tests the algorithm for different layouts and on a loop sorter as well. Despite differences in merge operations between line sorters and loop sorters, the process of allocating a slice for a parcel is similar. Conducting several tests and fine-tuning the algorithm can potentially yield promising results using dynamic programming.

While previous research has primarily focused on improving imbalance and utilization, there has been limited attention given to sustainability aspects. Although increased throughput and utilization can yield positive benefits such as reduced energy consumption, the magnitude of these effects is relatively small. A potential avenue for future research could involve identifying a control technique that can dynamically operate under different speed variations and adapt to conditions based on peak loading and normal load situations.

References

Agrawal, G.K., and S.S. Heragu. 2006. “A survey of automated material handling systems in 300-mm SemiconductorFabs.” *IEEE Transactions on Semiconductor Manufacturing* 19 (1): 112–120. https://doi.org/10.1109/TSM.2005.863217.

Ahn, H, D Del Vecchio - IEEE Transactions on Automatic, and undefined 2017. 2017. “Safety verification and control for collision avoidance at road intersections.” *ieeexplore.ieee.org,* https://ieeexplore.ieee.org/abstract/document/7987071/.

Allen, Theodore T. 2011. “Introduction to Discrete Event Simulation and Agent-based Modeling.” *Introduction to Discrete Event Simulation and Agent-based Modeling,* 1–7. https://doi.org/10.1007/978-0-85729-139-4_1.

Altman, Douglas G., and j. Martin Bland. 1995. “Statistics notes: The normal distribution.” *BMJ* 310 (6975): 298. ISSN: 0959-8138. https://doi.org/10.1136/BMJ.310.6975.298.

Arm, J., F. Zezulka, Z. Bradac, P. Marcon, V. Kaczmarczyk, T. Benesl, and T. Schroeder. 2018. “Implementing Industry 4.0 in Discrete Manufacturing: Options and Drawbacks.” 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018, *IFAC-PapersOnLine* 51 (6): 473–478. ISSN: 2405-8963. https://doi.org/https://doi.org/10.1016/j.ifacol.2018.07.106.

Autotech, Falcon. 2023. *Falcon Autotech Tilt-Tray Sorters.* Accessed June 30, 2023. https://www.falconautotech.com/tilt-tray-sorter/.

Azar, Ahmad Taher, Fernando E. Serrano, Anis Koubaa, Habiba A. Ibrahim, Nashwa Ahmad Kamal, Alaa Khamis, Ibraheem Kasim Ibraheem, et al. 2021. “Robust fractional-order sliding mode control design for UAVs subjected to atmospheric disturbances.” *Unmanned Aerial Systems: Theoretical Foundation and Applications: A Volume in Advances in Nonlinear Dynamics and Chaos (ANDC)* (January): 103–128. https://doi.org/10.1016/B978-0-12-820276-0.00012-1.

Babulak, Eduard, and Ming Wang. 2010. *Discrete Event Simulation: State of the Art.* 1–9. August. ISBN: 978-953-307-115-2. https://doi.org/10.13140/RG.2.1.2068.1767.

Bals, Pim. 2021. *A control strategy approach for automated handling systems in the Merge Zone Bals, Pim.* https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/.

Banks, Jerry. 1999. “Introduction to simulation.” *Winter Simulation Conference Proceedings* 1:7–13. ISSN: 02750708. https://doi.org/10.1145/324138.324142.

Bazaraa, M. S., John J. Jarvis, and Hanif D. Sherali. 2005. “Linear programming and network flows,” 726. https://www.wiley.com/en-ie/Linear+Programming+and+Network+Flows%2C+3rd+Edition-p-9780471703778.

Blackwell, David. 1962. “Discrete Dynamic Programming.” *The Annals of Mathematical Statistics* 33 (2): 719–726. ISSN: 0003-4851. https://doi.org/10.1214/AOMS/1177704593.

Boysen, Nils, Stefan Fedtke, and Felix Weidinger. 2017. “Truck Scheduling in the Postal Service Industry.” *https://doi-org.tudelft.idm.oclc.org/10.1287/trsc.2016.0722* 51 (2): 723–736. ISSN: 15265447. https://doi.org/10.1287/TRSC.2016.0722.

Cardenas, Ivan Dario, Wouter Dewulf, Thierry Vanelslander, Christophe Smet, and Joris Beckers. 2017. “The e-commerce parcel delivery market and the implications of home B2C deliveries vs pick-up points.” *The e-commerce parcel delivery market and the implications of home B2C deliveries vs pick-up points* 44 (2): 235–256. ISSN: 03035247. https://doi.org/10.19272/201706702004.

Chen, James C., Tzu-Li Chen, and Yu-Hsin Lee. 2023. “Simulation optimization for parcel hub scheduling problem in closed-loop sortation system with shortcuts.” *Simulation Modelling Practice and Theory* 124:102728. ISSN: 1569-190X. https://doi.org/https://doi.org/10.1016/j.simpat.2023.102728.

Cooper, Leon, and Mary W. Cooper. 1981. *Introduction to dynamic programming.* 289. Pergamon Press. ISBN: 9780080250656. http://p5070-www.sciencedirect.com.tudelft.idm.oclc.org/book/9780080250656/introduction-to-dynamic-programming.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition.* 3rd. The MIT Press. ISBN: 0262033844. https://dl.acm.org/doi/10.5555/1614191.

Cross, Tony, John Sutton, and Donald F. Wood. 1986. “Book reviews.” *Transportation Planning and Technology* 11 (1): 81–86. https://doi.org/10.1080/03081068608717331.

Dresner, Kurt, and Peter Stone. 2008. “A Multiagent Approach to Autonomous Intersection Management.” *J. Artif. Intell. Res.* 31:591–656. ISSN: 10769757. https://doi.org/10.1613/JAIR.2502.

Drissi Elbouzidi, Adnane, Abdessamad Ait El Cadi, Robert Pellerin, Samir Lamouri, Estefania Tobon Valencia, and Marie-Jane Bélanger. 2023. “The Role of AI in Warehouse Digital Twins: Literature Review.” *Applied Sciences* 13 (11). ISSN: 2076-3417. https://doi.org/10.3390/app13116746.

Erdogmus, Hakan. 2007. “Cost-Benefit Analysis of Software Development Techniques and Practices.” In *29th International Conference on Software Engineering (ICSE’07 Companion),* 178–179. https://doi.org/10.1109/ICSECOMPANION.2007.28.

Fedtke, Stefan, and Nils Boysen. 2014. “Layout Planning of Sortation Conveyors in Parcel Distribution Centers.” *https://doi.org/10.1287/trsc.2014.0540* 51 (1): 3–18. ISSN: 15265447. https://doi.org/10.1287/TRSC.2014.0540.

Fishman, George S. 2001. “Discrete-Event Simulation.” *Discrete-Event Simulation,* https://doi.org/10.1007/978-1-4757-3552-9.

Gasperin, Simon, and Dirk Jodin. 2012. "Dynamic merge of discrete goods flow – Impact on throughput and efficiency." *Logistics Journal Referierte Veröffentlichungen* 2012 (July). https://doi.org/10.2195/lj_Rev_gasperin_en_201202_01.

Gibney, Elizabeth. 2022. "Open-source language AI challenges big tech's models." *Nature* 606 (7916): 850–851. ISSN: 14764687. https://doi.org/10.1038/D41586-022-01705-Z.

Goldratt, Eliyahu M., and Jeff Cox. 2004. "The Goal: A Process of Ongoing Improvement - Eliyahu M. Goldratt, Jeff Cox - Google Books," https://books.google.co.uk/books?hl=en&lr=&id=HyxLDQAAQBAJ&oi=fnd&pg=PT4&ots=cofq7m8SvB&sig=4-y0JW1JliCF0xi68ONSM7IHwn8&redir_esc=y#v=onepage&q&f=false%20https://books.google.com.ph/books?hl=en&lr=&id=HyxLDQAAQBAJ&oi=fnd&pg=PT4&dq=The+goal:+A+process+of+ong.

Handling, Modern Material. 2023. *Sorting conveyor,* June. Accessed June 23, 2023. https://www.istockphoto.com/videos/conveyor-belt.

Haneyah, S. W.A., J. M.J. Schutten, P. C. Schuur, and W. H.M. Zijm. 2013. "Generic planning and control of automated material handling systems: Practical requirements versus existing theory." *Computers in Industry* 64 (3): 177–190. ISSN: 0166-3615. https://doi.org/10.1016/J.COMPIND.2012.11.003.

Hoven, Matthijs van den. 2019. *Utilization Improvement of a Sortation System for the Parcel Industry.* Technical report. Technical report, Eindhoven University of Technology, the Netherlands.

Huang, Shan, Adel W. Sadek, and Yunjie Zhao. 2012. "Assessing the Mobility and Environmental Benefits of Reservation-Based Intelligent Intersections Using an Integrated Simulator." *IEEE Transactions on Intelligent Transportation Systems* 13 (3): 1201–1214. ISSN: 1524-9050. https://doi.org/10.1109/TITS.2012.2186442.

Hyndman, Rob J. 2011. *Moving Averages,* edited by Miodrag Lovric, 866–869. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-04898-2. https://doi.org/10.1007/978-3-642-04898-2_380.

Jing, GG, WD Kelton, JC Arantes - … Proceedings (Cat. No …, and undefined 1998. 1998. "Modeling a controlled conveyor network with merging configuration." *ieeexplore.ieee.org,* https://ieeexplore.ieee.org/abstract/document/745851/.

Johnstone, M, D Creighton, S Nahavandi - Simulation Modelling Practice, and undefined 2015. 2015. "Simulation-based baggage handling system merge analysis." *Elsevier,* https://www.sciencedirect.com/science/article/pii/S1569190X15000131?casa_token=aiFNCiNpdY8AAAAA:y9LcU6KK7zLTl4EbffJknIMObhn1dbp3ks9azy8gpTEoi18zBj-W6FnEWR9bK1fvJkpe6j2wRdA.

Kellner, Marc I., Raymond J. Madachy, and David M. Raffo. 1999. "Software process simulation modeling: Why? What? How?" *Journal of Systems and Software* 46 (2-3): 91–105. ISSN: 0164-1212. https://doi.org/10.1016/S0164-1212(99)00003-5.

Kim, B J, J E Alleman, C S Gee, and J T Bandy. 1991. "Use of programmable logic controllers to automate control and monitoring of U. S. Army waste-water treatment systems. Final technical report" (July). https://www.osti.gov/biblio/5935158.

Kim, Gukhwa, Junbeom Kim, and Junjae Chae. 2017. "Balancing the baggage handling performance of a check-in area shared by multiple airlines." *Journal of Air Transport Management* 58 (January): 31–49. ISSN: 0969-6997. https://doi.org/10.1016/J.JAIRTRAMAN.2016.08.017.

Landschützer, Christian, Matthias Fritz, and Dirk Jodin. 2012. "KNOWLEDGE BASED ENGINEERING AND MODERN CAE FOR SORTING SYSTEMS" [in English]. *Proceedings in Manufacturing Systems* 7 (2): 69–76. ISSN: 2343-7472.

Li, L, FY Wang - IEEE Transactions on Vehicular technology, and undefined 2006. 2006. "Cooperative driving at blind crossings using intervehicle communication." *ieeexplore.ieee.org,* https://ieeexplore.ieee.org/abstract/document/4012536/?casa_token=OUnNkqIdre4AAAAA:KRrBMfBhy8LhRp-WNCoWD7Dv-VBD0PvL7q7R8LD1L-DATh_dEFMbqXqVQdAioslqtPPCi61Adg.

Li, PT, and X Zhou Methodological. 2017. "Recasting and optimizing intersection automation as a connected-and-automated-vehicle (CAV) scheduling problem: A sequential branch-and-bound search." *Elsevier,* https://www.sciencedirect.com/science/article/pii/S0191261517304782?casa_token=q_Aw0jK_tUoAAAAA:ocGlFvErY4ARGMRvuba4FCFwBaqrBfweiHmXSL7ZMEnz3ZY1bk3Gx-Xv9LAY1t_LJBaVpWEPnvU.

Lin, Shang Chien, Hsiang Hsu, Yi Ting Lin, Chung Wei Lin, Iris Hui Ru Jiang, and Changliu Liu. 2020. "A Dynamic Programming Approach to Optimal Lane Merging of Connected and Autonomous Vehicles." *IEEE Intelligent Vehicles Symposium, Proceedings,* 349–356. https://doi.org/10.1109/IV47402.2020.9304813.

Maria, Anu. 1997. "Introduction to Modeling and Simulation." In *Proceedings of the 29th Conference on Winter Simulation,* 7–13. WSC '97. Atlanta, Georgia, USA: IEEE Computer Society. ISBN: 078034278X. https://doi.org/10.1145/268437.268440.

Marinescu, Dan, Jan Čurn, Mélanie Bouroche, and Vinny Cahill. 2012. "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach." *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC,* 900–906. https://doi.org/10.1109/ITSC.2012.6338779.

McGuire, Patrick M. 2009. "Conveyors : Application, Selection, and Integration" (August). https://doi.org/10.1201/9781439803905.

Meens, Jasper. 2017. "Model Based Design Approach for Merge Balancing," Eindhoven University of Technology, the Netherlands.

Meng, Yue, Li Li, Fei Yue Wang, Keqiang Li, and Zhiheng Li. 2018. "Analysis of Cooperative Driving Strategies for Nonsignalized Intersections." *IEEE Transactions on Vehicular Technology* 67 (4): 2900–2911. ISSN: 00189545. https://doi.org/10.1109/TVT.2017.2780269.

Mishan, E.J., and Euston Quah. 2020. "Cost-Benefit Analysis" (August). https://doi.org/10.4324/9781351029780.

Mothership. 2015. *Mothership weighing and scanning.* Accessed February 15, 2023. https://mothership.sg/2022/03/ninja-vans-new-automated-hub/.

Müller, Eduardo Rauh, Rodrigo Castelan Carlson, and Werner Kraus Junior. 2016. "Intersection control for automated vehicles with MILP." *IFAC-PapersOnLine* 49 (3): 37–42. ISSN: 24058963. https://doi.org/10.1016/J.IFACOL.2016.07.007.

Nayak, Sukanta. 2020. "Dynamic programming." *Fundamentals of Optimization Techniques with Algorithms* (January): 191–221. https://doi.org/10.1016/B978-0-12-821126-7.00007-3.

Nelson, Randolph. 1995. "Markov Processes." *Probability, Stochastic Processes, and Queueing Theory,* 329–389. https://doi.org/10.1007/978-1-4757-2426-4_8. https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-1-4757-2426-4_8.

Peeters, K. 2015. "Balancing Control of Material Handling Systems." PhD diss., Eindhoven University of Technology, the Netherlands.

Peffers, Ken, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. 2014. "A Design Science Research Methodology for Information Systems Research." *https://doi-org.tudelft.idm.oclc.org/10.2753/MIS0742-1222240302* 24 (3): 45–77. ISSN: 07421222. https://doi.org/10.2753/MIS0742-1222240302.

Pei, Huaxin, Shuo Feng, Yi Zhang, and Danya Yao. 2019. "A Cooperative Driving Strategy for Merging at On-Ramps Based on Dynamic Programming." *IEEE Transactions on Vehicular Technology* 68 (12): 11646–11656. ISSN: 19399359. https://doi.org/10.1109/TVT.2019.2947192.

Pei, Huaxin, Yuxiao Zhang, Yi Zhang, and Shuo Feng. 2022. "Optimal Cooperative Driving at Signal-Free Intersections With Polynomial-Time Complexity." *IEEE Transactions on Intelligent Transportation Systems* 23 (8): 12908–12920. ISSN: 15580016. https://doi.org/10.1109/TITS.2021.3118592.

Pfohl, Hans Christian, Pascal Wolff, and Johannes Kern. 2020. "Transshipment hub automation in China's courier/express/parcel sector." *Urban Freight Transportation Systems* (January): 163–180. https://doi.org/10.1016/B978-0-12-817362-6.00009-4.

Python Core Team. 2019. *Python: A dynamic, open source programming language.* Python Software Foundation. https://www.python.org/.

Ramamritham, Krithi, and John A. Stankovic. 1994. "Scheduling Algorithms and Operating Systems Support for Real-Time Systems." *Proceedings of the IEEE* 82 (1): 55–67. ISSN: 15582256. https://doi.org/10.1109/5.259426.

Sargent, Robert. 2011. "Verification and validation of simulation models," 37:166–183. January. https://doi.org/10.1109/WSC.2010.5679166.

Schaffer, Russel, and Robert Sedgewick. 1993. "The Analysis of Heapsort." *Journal of Algorithms* 15 (1): 76–100. ISSN: 0196-6774. https://doi.org/10.1006/JAGM.1993.1031.

Sharma, Prateek. 2015. "Discrete-Event Simulation." *INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH* 4 (04). ISSN: 2277-8616. www.ijstr.org.

Shaw, Jason D. 2011. "Turnover rates and organizational performance." *http://dx.doi.org/10.1177/2041386610382152* 1 (3): 187–213. ISSN: 2041-3866. https : / / doi . org/10.1177/2041386610382152.

Stewart, William J. 2009. "Probability, Markov Chains, Queues, and Simulation." *(No Title)* (July). https://doi.org/10.2307/J.CTVCM4GTC.

Thacker, B.H., S.W. Doebling, F.M. Hemez, M.C. Anderson, J.E. Pepin, and E.A. Rodriguez. 2004. *Concepts of Model Verification and Validation.* http://inis.iaea.org/Search/search.aspx?orig_q=RN:36030870.

Thorne, David R. 2006. "Throughput: A simple performance index with desirable characteristics." *Behavior Research Methods* 38 (4): 569–573. ISSN: 1554351X. https : / / doi . org / 10 . 3758 / BF03193886/METRICS.

Vanderlande. 2023a. *Vanderlande - Parcel Innovative systems: Sorting,* March. Accessed March 23, 2023. https://www.vanderlande.com/systems/sorting/.

———. 2023b. *Vanderlande Technologies.* Accessed March 23, 2023. https://www.vanderlande.com/.

———. 2023c. *Warehousing sortation system - CROSSORTER,* May. Accessed May 23, 2023. https://www.vanderlande.com/systems/sortation/crossorter/.

———. 2023d. *Warehousing sortation system - CROSSORTER,* May. Accessed June 29, 2023. https://www.vanderlande.com/systems/sorting/crossorter-1200-and-1500/.

Vanderlande.com. 2023. *Company profile | About Vanderlande - Vanderlande industries,* March. Accessed March 23, 2023. https://www.vanderlande.com/about-vanderlande/company-profile/.

Ven, Otto Van De. 2022. *Design of a platooning algorithm for real-time control of a Posisorter's overflow zone DEPARTMENT OF MECHANICAL ENGINEERING.*

Yunardi, Riky Tri, Winarno, and Pujiyanto. 2016. "Contour-based object detection in Automatic Sorting System for a parcel boxes." *ICAMIMIA 2015 - International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation, Proceeding - In conjunction with Industrial Mechatronics and Automation Exhibition, IMAE* (July): 38–41. https://doi.org/10.1109/ICAMIMIA.2015.7507998.

A

Research Paper

Research Paper is continued from the next page

Enhancing Merge Conveyor Utilization in a Line-Sorter Sortation System by Introducing a New Control Algorithm

Revanth Sai Yayavaram
revanth.yayavaram97@gmail.com

Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology

Abstract

The parcel transport industry utilizes automated parcel sorting systems to effectively manage incoming parcels. These systems typically comprise multiple infeed conveyors that converge onto a single merge conveyor for parcel merging. However, the conventional First-Come-First-Serve merging method results in reduced utilization of the merge conveyor and throughput. To address this, a novel approach is proposed in this study, combining a dynamic programming-based control algorithm with a maximum heap algorithm. This approach aims to enhance the utilization of the merge conveyor and achieve a balanced load among the infeeds. To optimize utilization, an additional imaginary segment is introduced ahead of the merge conveyor, synchronously moving with the merge conveyor's speed. This segment is intelligently populated with parcels using dynamic programming, eliminating unnecessary empty spaces. Results demonstrate that the dynamic programming approach outperforms other techniques in terms of utilization. This research presents an initial investigation into implementing dynamic programming as a control algorithm to improve merge conveyor utilization in the parcel transport industry.

Key words: Line Sorter Sorting System, Dynamic Programming, Max Heap, Parcel Merging, Throughput, Utilization

1 Introduction

Online orders and e-commerce have experienced exponential growth, reaching unprecedented levels in the present world scenario. Today, it is common for individuals to expect their orders or parcels to be delivered within a couple of days or no more than a week. However, behind the scene, there exists a vast and complex world of Material Handling Systems (MHS) that operates to make this possible. MHS can be observed in various aspects of modern economies, such as parcel and postal services, airports handling baggage, warehouses moving pallet loads, seaports managing shipping containers, manufacturing systems transporting parts, and many other applications (S. W. Haneyah et al. 2013).

A Material Handling System encompasses a range of methods, facilities, equipment, and labour employed to move materials from one source to another (Cross et al. 1986). Customizable to specific industry needs, MHS efficiently handle diverse products, from small parts to heavy loads. Automated Material Handling Systems (AMHS), a type of MHS, employ advanced technologies to automate material movement, storage, and retrieval (Agrawal and Heragu 2006). In the parcel industry, parcel sorting systems, consisting of conveyors, sorters, scanners, and other components, play a vital role in sorting parcels based on destination, size, weight, and other criteria (Pfohl et al. 2020). Companies like UPS, DHL, and FedEx heavily rely on robust conveying systems to handle the immense influx of parcels (Cardenas et al. 2017).

A typical parcel sorting system utilizes single or multiple infeed lines to load parcels onto the merge conveyor (S. Haneyah et al. 2011). Parcels received from trucks are placed on the infeed belt by operators or robots, with variable numbers, lengths, and spacing of the infeeds. Photoelectric Cells (PECs) on the infeed conveyors assess parcel dimensions and trigger a request to reserve a slice in the available empty space on the merge conveyor through communication between the infeed and merge controllers. A typical slice and the space can be seen in Figure 1. The fly-through parcels are parcels that haven't been sorted in the initial sorting area and are reintroduced later for sorting (S. W. Haneyah et al. 2013). The current reservation algorithm proves ineffective, resulting in empty spaces on the merge conveyor and imbalanced reservations after the parcels merge, as depicted in Figure 2. This issue of decreased utilization

requires a new control algorithm to optimize the parcel merging process and improve system efficiency in parcel handling industries (Drissi Elbouzidi et al. 2023).

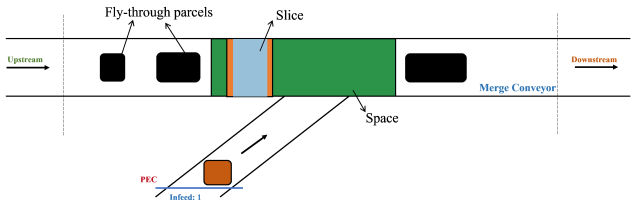


Fig. 1. Single infeed system

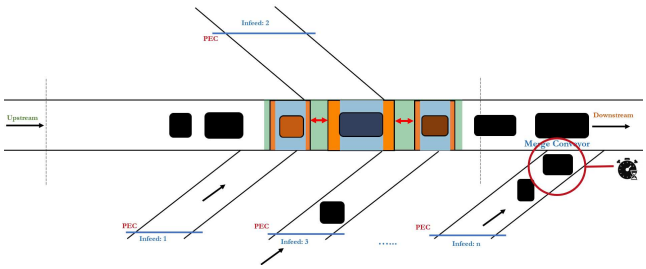


Fig. 2. Multiple infeed system

To optimize efficiency, the integration of sensors and sorting algorithms is crucial for processing thousands of parcels daily (Drissi Elbouzidi et al. 2023). This research aims to introduce a new control algorithm to enhance the utilization of the merge conveyor of a line sorter sortation system without ignoring the load balance among the infeeds. An increase in utilization can also result in higher throughput of the system. Briefly stated, throughput is the rate at which the material moves through the system per unit time (Thorne 2006), is a crucial metric in this industry and is usually measured in terms of the number of parcels per hour or day (a unit time). Utilization, on the other hand, is another metric that refers to the degree to which the system is being used or occupied relative to its maximum capacity (S. W. Haneyah et al. 2013). Leveraging advanced control algorithms and system optimization techniques, sorting systems' efficiency can be improved, facilitating more effective handling of higher parcel volumes. To achieve the goal of introducing a new control algorithm, the following research question was developed:

"How can the parcel merging process and utilization of the merge conveyor in a line-sorter sortation system be improved by introducing a new control algorithm?"

2 Research Approach

A well-supported research methodology, as formulated by Peffers et al. (2014), can also be applied to research control algorithms in existing systems. In their paper, the authors outline a methodology for information systems, which can be adapted to address problems related to technology and organizations. This methodology is well-suited for the current research, which focuses on developing and introducing a new control algorithm to improve the utilization of the main conveyor of a parcel sorting system. The research approach begins by identifying the problem of low utilization in the parcel industry's sorting systems and defining the objectives for addressing this issue. The primary concern is the efficient utilization of the main conveyor (also known as the merge conveyor) in handling the increasing volume of parcels while avoiding costly system modifications. To achieve this, the focus is on enhancing the control algorithm that manages local parcel traffic, with particular emphasis on the critical step of parcel merging. By improving the control algorithm, it is expected that higher utilization of the conveying system can be achieved.

The next step involves the development of an efficient control algorithm that can improve the parcel merging process and increase utilization. To accomplish this, potential algorithms that have shown promising results in similar applications will be explored. This exploration will involve a thorough review of the literature and an examination of existing algorithms to identify the most suitable solution for the present problem.

Once the control algorithm is developed, the next step is to design and develop a simulation model of the sorting system, specifically tailored to the given layout. The simulation model will serve as a virtual representation of the parcel sorting system, allowing for the demonstration of how the new control algorithm can be implemented in the parcel merging process of the line-sorter sortation system. Through a series of experiments conducted using the simulation model, the effectiveness of the solution in enhancing utilization can be evaluated. The experiments will consider different scenarios and parameters to assess the algorithm's performance and its impact on key performance indicators such as utilization, throughput, and imbalance.

In the final step, the experiments conducted in the previous step will be utilized to evaluate the developed algorithm. Key performance indicators (KPIs) such as utilization, throughput, and imbalance will be employed to compare the performance of the developed algorithm with that of the current algorithm used in the industry. This evaluation will provide evidence to demonstrate that the developed algorithm is a viable solution to the problem of low utilization. Additionally, the costs associated with implementing the current algorithm will be analyzed to assess the economic implications of adopting the new control algorithm.

3 Literature Review

Efficient control algorithms play a vital role in optimizing sorting systems in the parcel industry. While increas-

ing conveyor speed improves throughput, it can lead to unintended consequences like slips. Layout changes offer opportunities for improvement but increase system footprint and cost. Balancing the benefits and costs of layout changes is crucial for cost-benefit analysis (Fedtke and Boysen 2014). Limited literature exists on control algorithm enhancement, highlighting the need for further investigation. Previous studies by Peeters (2015), Meens (2017), and Hoven (2019) have provided valuable insights into modifying control strategies. Additional strategies by S. W. Haneyah et al. (2013) and Kim et al. (2017) in the literature show potential and outperform existing techniques.

The presence of multiple infeeds in the system results in high imbalance measures. Load balancing techniques such as FIFO, Longest Queue First, Highest Priority First, random, and Round Robin have been studied by several researchers (Jing et al. 1998; Peeters 2015). According to Stewart (2009), FIFO follows a first-come-first-served method; the Longest Queue First prioritizes the queue with the most waiting items; the Highest Priority First serves items with higher priority first; Round Robin allocates a fixed time for each item in the queue. All these techniques aim to reduce waiting times and improve load balancing.

Often researchers draw parallels between job-shop machine scheduling problems and parcel merging. The selection of an algorithm significantly impacts the ability to achieve optimal solutions (Cormen et al. 2009). Algorithms can be classified based on their implementation method, execution behaviour, determinism, and other factors. Understanding these classifications helps in selecting suitable algorithms.

S. W. Haneyah et al. (2013) developed an Integer Linear Programming (ILP) approach for optimizing parcel merging using tilt-trays in a parcel sorting system. They aimed to maximize utilization and minimize waiting time imbalances. The research involved two phases: an exact static branch and bound optimization technique and a dynamic space allocation approach. The latter utilized a Priority Based Algorithm (PBA) to prioritize parcel flow from all infeeds, based on waiting times. The researchers conducted experiments and found that the dynamic allocation approach achieved a balanced workload and improved throughput. The proposed approach reduced waiting time imbalances from 17% to 4.2% and resulted in slightly higher throughput compared to the conventional branch and bound method.

During the literature review, Johnstone et al. (2015) proposed merge allocation rules for baggage handling systems (BHS), including FIFO, feeder line priority, merge line priority, merge flush, and merge timeout. They compared fixed and variable window-size algorithms with a FIFO method, finding that the variable length algorithm performed better when the infeed was in the desired position.

Kim et al. (2017) extended this research by introducing a reallocation algorithm in the window assignment control logic system. They tracked eligible windows for reallocation based on the maximum number of reallocations and the position of the corresponding infeed. The reallocation procedure involved switching baggage between windows and breaking the FIFO rule. They evaluated each potential switch using a benefit function that considered the waiting times of the new and old window positions. The findings showed significant reductions in imbalance and waiting times, although complete elimination was not achieved. The level of balance depended on factors such as the maximum number of reallocation switches and the distribution of incoming baggage. Uniform and triangular distributions performed better with a single reallocation, while the exponential distribution showed improved results with two reallocations.

Traffic management systems, particularly highway on-ramp merging and intersection management of connected autonomous vehicles (CAVs), provide inspiration for controlling the parcel merging process. Researchers have proposed various merging algorithms based on cooperative behaviour, optimization methods, and dynamic programming (Marinescu et al. 2012). To determine optimal passing orders, Li and Methodological (2017), Müller et al. (2016), and Ahn et al. (2017) formulated the problem as a Mixed Integer Linear Programming (MILP) and provided solutions. Li et al. (2006) proposed a solution using a spanning tree representation coupled with pruning rules, offering improved computational efficiency.

Cooperative driving strategies can be categorized as "ad hoc negotiation-based" or "planning-based" (Meng et al. 2018). Ad hoc negotiation-based strategies approximate a first-come-first-served order through bilateral negotiations, while planning-based approaches create long-term driving plans. Planning-based approaches offer more flexibility but require more computational resources. To overcome computational challenges, Pei et al. (2019) proposed a computationally efficient strategy based on Dynamic Programming (DP). Their DP algorithm optimizes passing order and access times, achieving lower computational times compared to exhaustive enumeration. Lin et al. (2020) extended the DP algorithm to various merging scenarios, demonstrating its efficiency in minimizing delays and passing times compared to greedy algorithms. Pei et al. (2022) proposed a novel state-space formulation for cooperative driving at signal-free intersections. Their dynamic programming approach achieved optimality in cooperative driving and enabled real-time implementation.

As evident from the information provided above, the primary indicators employed to assess the models in ques-

tion were throughput, utilization, and imbalance. This part of the literature review section will subsequently elaborate on each of these KPIs, explaining them comprehensively.

Throughput
Throughput is the rate at which material moves through the system per unit time (Thorne 2006). In the parcel sorting industry, throughput is measured as the number of parcels processed per unit of time, such as parcels per hour or day.

$$\text{Throughput} = \frac{\sum(Number\ of\ Parcels\ processed)}{Time\_duration}[parcels\ per\ hour] \tag{1}$$

Utilization
Utilization is another metric that refers to the degree to which the system is being used or occupied relative to its maximum capacity (S. W. Haneyah et al. 2013). Higher utilization implies efficient usage of the conveyor system, with minimal empty space between parcels, thereby enabling more parcels to be transported within a given time frame.

$$Utilization = \frac{\sum_{n\in N} Parcel\_lengths_n + (N-1)Fixed\_gap}{V_{merge} * Time\_duration}[\%] \tag{2}$$

where N is the total number of parcels processed by the system and the $Fixed\_gap$, is the minimum gap that is required to avoid parcel overlapping or collision. Further, V_merge is the velocity of the merge conveyor and the $Time\_duration$ is the total time the system has been in running.

Both throughput and utilization are directly related to each other. When the utilization of the conveyor is low, there are more empty spaces between the parcels on the merge conveyor, leading to a reduced throughput. However, when considering the parcel size variations, there will be certain situations where a huge parcel occupies the most space on the merge conveyor which translates to less throughput. These kinds of situations are inevitable and are the primary reason for companies to consider the average throughput of the system.

Load Imbalance
Another significant issue identified by many researchers is the notable difference in throughput between upstream and downstream infeed conveyors. To address this, a KPI is introduced to measure the disparity in throughput between the minimum and maximum values for each infeed, thereby consolidating it into a single metric. Hence, the imbalance can be calculated as:

$$Imbalance = \frac{max_{f\in F}\{Throughput_f\} - min_{f\in F}\{Throughput_f\}}{max_{f\in F}\{Throughput_f\}}.100\ [\%] \tag{3}$$

where f is the infeed in a set of infeeds denoted by F.

4 Simulation Model

The development of a simulation model begins with a thorough understanding of the system that will be modelled. Following this, the model conceptualization phase involves simplifying the real-world system. This includes identifying and breaking down the system to be modelled, as well as gaining a clear understanding of the relevant processes involved in the merging of parcels into the merge conveyor. Once all the relevant elements for accurately modelling the system are identified, the conceptualization phase can be translated into the implementation of the model. This involves implementing the system in software. Subsequently, the simulation model undergoes verification to ensure it performs correctly and validation to assess whether it can replace the real system for experimentation purposes. After the simulation model passes these checks, experiments can be carried out using the model to check for the feasibility of the developed model in real-world applications.

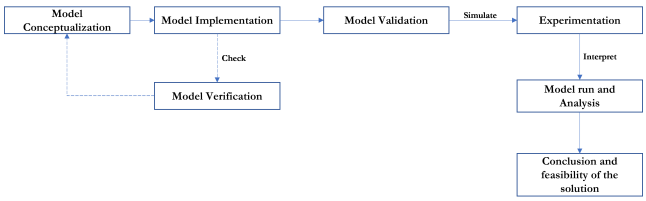


Fig. 3. Modelling steps, Inspired from (Sharma 2015)

4.1 Model Conceptualization

The initial phase of model development involves conceptualizing the model. This crucial step establishes the foundation for the entire modelling process by identifying and defining key elements and relationships. The goal is to create a conceptual framework that represents the essential components and interactions within the system. The output is a well-defined and structured conceptual model that serves as the basis for subsequent development, parameterization, and validation.

Process 1: Parcel Generation
The parcel generator process is crucial for simulating the existence of parcels and their associated processes. Parcels are generated based on specific dimensions derived from a normal distribution throughout the simulation. Generation occurs randomly according to the infeed capacity of 3200 parcels per hour. Parcels are added

to the infeeds only if sufficient space is available, maintaining the required gap between parcels. If space is unavailable, the incoming parcel is held until the merge conveyor can accept it.

Generated parcels enter the infeed and their entry time and length are recorded in an array called "parcel entries." This array serves as crucial communication for subsequent processes, indicating that the parcels are ready for merging and awaiting confirmation of their arrival time at the merge point. The "parcel entries" array represents the entry time and length of parcels for each infeed, formatted as [(time, length)] pairs.

In order to proceed with the control algorithm process, there are a few things that must be understood. Consider a hypothetical scenario with random-length parcels on each of the six infeed conveyors, waiting to merge onto the merge conveyor. Before starting the process, an imaginary segment is generated aligned with the merge conveyor's axis, located before the actual start of the merge conveyor (see the green segment in Figure 4). This segment has the same velocity as the merge conveyor and serves as the event trigger for the model. Its length is predetermined to be a maximum of 500 cm, ensuring compatibility with the fly-through detection located at 650 cm. The control algorithm described in the forthcoming analysis aims to optimize the segment's utilization by efficiently filling the given parcels. If there is residual space after assigning slices to all the parcels, the subsequent segment starts from the trailing edge of the last-filled parcel.

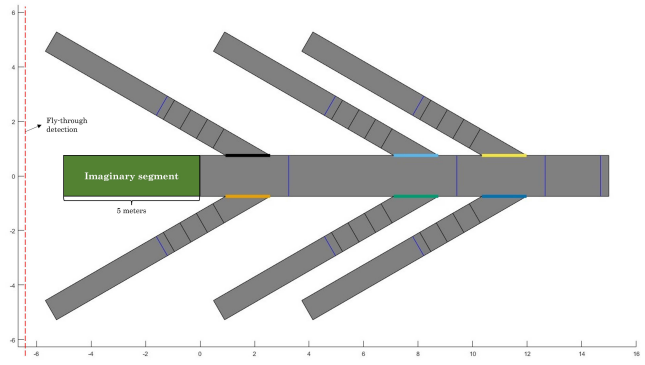


Fig. 4. Imaginary Segment and fly-through detection point

Process 2: Control Algorithm

Following the initial entries of parcels, these entries are utilized to determine the optimal sequence for the parcels on the infeeds that efficiently fill the entire segment while maintaining appropriate gaps between the parcels. This is where DP becomes essential. To tackle the present problem using DP, it is essential to establish the state space and state transition for the model.

Consider a hypothetical scenario with two infeeds, each containing two parcels. For simplicity, we neglect the gap requirements. Infeed 1 has parcels $[10, 15]$ in its queue, while Infeed 2 has parcels $[25, 22]$. The corresponding state space and state transitions for this example will be explained from here on.

State Space: The state space consists of $(x + y + 1)$ stages, denoting the number of parcels from Infeed 1 (x) and Infeed 2 (y), ranging from S_0 to S_{x+y}. Each stage is described by a triplet state (x_i, y_i, g_i), where g_i represents the remaining gap and i is the stage number. The initial state is $S_0(0, 0, g)$, and the final state is $S_{x+y}(x, y, g)$.

State Transition: The state transition equation is given by:

$$S_i(x_i, y_i, g_i) = h((x_{i-1}, y_{i-1}, g_{i-1}), g_i) \tag{4}$$

where $h(.)$ represents the state transition function.

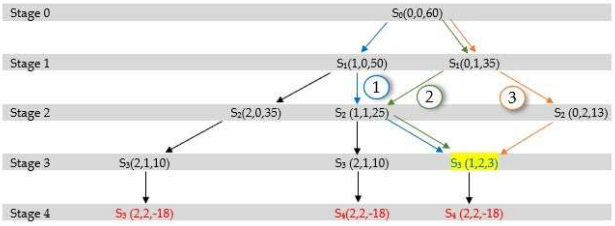


Fig. 5. Parcel assignment using DP

As mentioned by Pei et al. (2019), the aforementioned state space and state transition possess the following properties:

(1) There is a decision-making process consisting of multiple stages, starting from stage 1 to stage "i." The outcomes or results of the decisions made in each stage are reflected in the parameters of a state called "S_i." In simpler terms, as the decision-making process advances from stage 1 to stage i, the choices made at each stage have an impact on the state S_i.
(2) The shift from one state to another happens when moving from one stage to the next stage in the decision-making process.
(3) Different orders of the state can attain the same state in the next transition. For instance, $S_1(1, 0, 50)$ and $S_1(0, 1, 35)$ attain the same state $S_2(1, 1, 25)$ in stage 2.
(4) Infeasible solutions are directly eliminated during the construction of the solution space by keeping track of the parcels that have been assigned or not.

According to Nelson (1995), the Markovian property is a stochastic property that has a form of historical dependency where the probability of each event depends only on the state attained in the previous event. By virtue of Property 1, the state exhibits the Markovian property,

which represents the feasible conditions of the DP model (Blackwell 1962). Property 2 enables a remarkable reduction in the number of transitions in the DP model. Similarly, Property 3 facilitates an extensive decrease in the number of states in the DP model. Property 4, while ensuring optimality, substantially reduces the size of the solution space. Consequently, the state space solely encompasses all feasible passing orders of the parcels.

From the Figure 5, it becomes evident that there are multiple paths leading to the optimal stage of leaving only a 3-centimetre gap. However, certain states highlighted in red are considered infeasible since a negative gap is not possible. Moreover, paths 1 and 3 depicted in Figure 5 are unattainable due to the presence of an inter-departure time constraint, which prevents two parcels from the same infeed from being consecutively merged. Consequently, these paths are deemed infeasible and cannot be regarded as viable solutions. Nevertheless, one limitation of the DP approach is its inherent bias towards starting from the left and searching the solution space. In the context of a parcel industry, this sequential left-to-right approach may not be optimal, as it can lead to downstream infeeds being left with insufficient space due to upstream infeeds delivering parcels at higher rates in some cases.

To mitigate this biasing issue, a priority system can be introduced based on the occupancy levels of the infeeds. By assigning higher priority to the more occupied infeed compared to others, a more balanced allocation of space can be achieved. This approach has shown promising results in previous studies (S. W. Haneyah et al. 2013), making it a viable option to ensure that the control algorithm remains unbiased towards any specific infeed. To facilitate the sorting based on the maximum filled queue, a max heap algorithm is employed for the current model. However, it is worth noting that there is always a possibility to shift priorities to other infeeds in case one of the infeeds has more infeed capacity. For example, if one infeed has a significantly higher parcel delivery rate compared to the others, using a most filled queue technique may not be optimal. In this case, the infeed with higher capacity will consistently receive a larger volume of parcels compared to the others. This can result in imbalance again. Since, the capacity of all the infeeds is the same, a maximum filled queue option is suitable for the current model.

Process 3: Handling parcels entries and exits
The function manages the entry and exit of parcels in a time-limited conveyor system, handling multiple queues. It processes parcels by comparing their arrival and exit times within the designated timeframe, ensuring proper order based on these times. For each queue, the function examines the next entry and exit parcels within a loop. If both parcels fall within the timeframe, the function takes the appropriate action. If the entry parcel arrives before the exit parcel, it adds the entry parcel to the corresponding queue. If the exit parcel occurs earlier, a parcel is removed from the queue. This process continues as long as there are both entry and exit parcels within the timeframe.

In the given scenario, a calculation is performed to determine the time it takes for a specific slice on the conveyor system to reach each corresponding infeed in a sequence. This calculated time serves as the exit time for the parcel. If the calculated time to reach the assigned infeed is denoted as x, the parcel undergoes acceleration to reach the merge point within the designated timeframe. If the distance between the merge point and the parcel's location on the infeed is too great to cover in the assigned time, the parcel needs to be accelerated at $t+x$ seconds. This acceleration ensures the parcel arrives at the assigned time, enabling timely merging within the conveyor system.

4.2 Model Implementation

Simulation is a fundamental and indispensable tool across diverse fields, serving as a robust method for comprehending, analyzing, and forecasting intricate systems (Kellner et al. 1999). Specifically, Discrete Event Simulation (DES) is employed to model system behaviour by representing events that occur at specific points in time. These events encompass arrivals, departures, state changes, or other significant occurrences within the system (Babulak and Wang 2010). By tracking the chronological order of events and the system's state, simulation enables the analysis of system performance and behaviour. Moreover, DES is particularly suitable for simulating intricate and dynamic systems, which makes it the chosen approach for this research. While analytical solutions provide precise results for simpler queuing models, they are often infeasible for complex systems. DES, on the other hand, models discrete events and interactions within the system, allowing for the analysis of complicated queuing networks. DES offers flexibility, efficiency, scalability, and the ability to capture dynamic behaviour. However, it requires careful model construction and may involve some level of approximation (Banks 1999).

Python was selected as the preferred software for simulation due to its versatility, extensive libraries, readability, integration capabilities, and cost-effectiveness. With its advantages and availability of open-source libraries, Python serves as the ideal choice for conducting the simulations in this study (Python Core Team 2019). The hardware setup for this research consists of an Intel(R) Core(TM) i7-10850H CPU operating at a clock speed of 2.70 GHz, accompanied by 64 GB of RAM. The operating system utilized is Windows 10 Enterprise. The software employed for the research includes Python 3.11.3 as the programming language and Visual Studio Code IDE as the development environment.

4.3 Model Verification and Validation

According to Thacker et al. (2004), model verification involves that the model implementation accurately reflects the developer's conceptual depiction of the model and its solution. To ensure that the model is verified and provides the expected results, manual checks were put into place to check for any unwanted effects. Firstly, the allocation of slices to parcels was examined to ensure proper assignment based on the starting and ending locations of the slice and the corresponding infeed. Secondly, it was verified that no two parcels from the same infeed occurred consecutively, and the constraint of not exceeding 3200 parcels per hour was upheld. Lastly, the placement of slices on the segment was validated to ensure that parcels could reach the merge within the specified timeframe.

Model validation, according to Sargent (2011), involves confirming that a computerized model demonstrates a satisfactory level of accuracy aligned with its intended purpose. The validation process compares the model's outputs against known or observed data, separate from the training data, to assess its performance. Validation is an iterative process that includes refining the model based on evaluation results. Various validation techniques were employed, such as extreme condition tests, and parameter variability through sensitivity analysis.

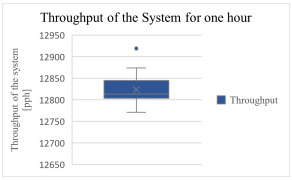
Extreme condition tests assess the model's plausibility and reliability under highly improbable combinations of factors. Parameter variability through sensitivity analysis systematically modifies input values and parameters to analyze their impact on the model's behavior. These validation techniques ensure the model's accuracy, robustness, and ability to handle diverse scenarios.

From the sensitivity analysis, it was evident that segment size plays a critical role in the utilization and throughput of the system. Smaller segment sizes result in higher utilization and throughput, especially when dealing with fly-through parcels. Larger segment sizes can lead to wasted space and lower utilization because of the capacity of the infeeds. Reducing the segment size improves both utilization and throughput, considering the existing constraints. Factors such as the timing of parcel announcements and the infeed capacity also influence the impact of segment size. Achieving earlier parcel announcements and exceeding the infeed capacity of 3200 parcels per hour per infeed can mitigate the negative effects of larger segment sizes.

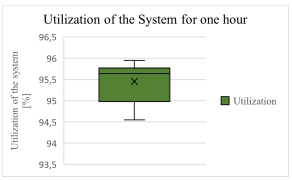
4.4 Model Experimentation

In DES, the simulation duration is crucial for accurately evaluating system behaviour and performance. A longer simulation time ensures system stabilization and steady-state conditions, minimizing the impact of transient behaviour and initial conditions. By running the simulation for an extended period, a more representative assessment of the system's performance under realistic conditions is obtained, considering a larger sample size of parcels and capturing a wider range of system dynamics. To achieve a steady state, the simulation must allow for an adequate number of events and interactions, particularly for metrics like utilization that depend on the passage of multiple parcels through the system. Only after several parcels have been processed can the accurate behaviour of the real system be determined, as parcel trajectories are influenced by preceding parcels and the initial startup phase.

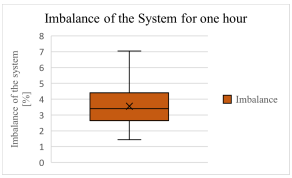
To evaluate the algorithm's performance over an extended period, a simulation time of one hour was selected. This duration captures an ample number of events and interactions within the model, offering a comprehensive understanding of its behaviour. The simulation was repeated 20 times to consider result variability. Analyzing the generated graphs, crucial metrics including throughput, utilization, imbalance, and infeed parcel counts provide valuable insights into the system's performance during this timeframe.



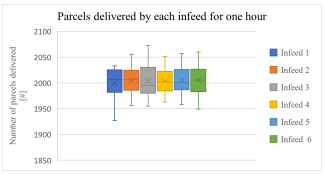
((a)) Throughput of the system



((b)) Utilization of the system



((c)) Imbalance among the infeeds



((d)) Individual infeed throughput

The simulation results, depicted in Figure 1, provide valuable insights into the system's performance during a one-hour time period. The average throughput remains stable at around 12,800 parcels, indicating efficient processing capacity. The utilization of the merge conveyor is commendably high at approximately 95.7%, ensuring effective space utilization. The imbalance in throughput among the infeeds averages around 3.5%, indicating a balanced workload distribution. However, there are occasional instances where the imbalance exceeds 5%. Overall, the system achieves equitable workload allocation among the infeed conveyors, with average throughput ranging from 1,950 to 2,055 parcels per infeed.

Comparison with industry level algorithm
A comprehensive comparison is conducted between the newly developed algorithm and the current industry-

level algorithm, focusing on key performance indicators such as throughput, utilization, and load imbalance. The primary objective is to thoroughly assess the performance of the proposed algorithm in these areas and identify its strengths and weaknesses compared to the current algorithm.

To ensure a realistic evaluation, various scenarios are meticulously designed to resemble both typical operational conditions and exceptional situations. These scenarios are broadly categorized into two groups: one without any fly-through parcels and the other with a 10% occurrence rate of fly-through parcels, closely mirroring real-world circumstances. Within these scenarios, different cases are considered to provide a holistic view of the algorithm's performance. For explanation purposes, the first case from each scenario will be explained in detail.

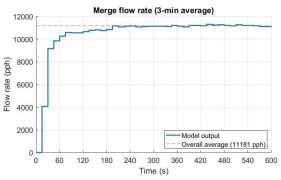
Scenario 1: Without Fly-through Parcels

(1) Happy flow from all six infeeds
(2) First and last infeeds operating, while middle infeeds offline
(3) Only the first four infeeds operating, with the remaining two infeeds temporarily out of service
(4) Only the first two infeeds operating, with the other infeeds non-operational

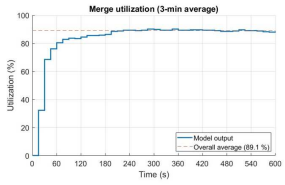
Table 1
No fly through all cases

Scenario	Current Algorithm	Current Algorithm	Developed Algorithm	Developed Algorithm
No fly through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
Happy Flow	11181	89.1	12540	95.13
Infeed 1 and Infeed 6	6645	53.3	6929	54.31
4 infeeds	10621	84.3	11493	88.68
2 infeeds	6738	53.8	7074	56.94

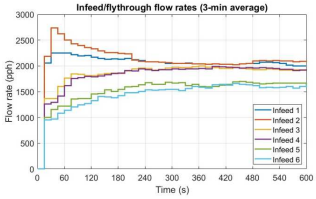
Case 1: Happy flow Comparing the performance of the current algorithm and the developed algorithm in a scenario with a smooth flow of parcels from all six infeeds reveals several notable findings. Firstly, the developed algorithm outperforms the current industry standard in terms of throughput. The current algorithm achieves a throughput of 11,181 parcels per hour, while the developed algorithm significantly improves this metric with a throughput of 12,540 parcels per hour. This signifies the enhanced efficiency of the developed algorithm in processing and merging parcels, resulting in higher overall throughput. Additionally, the utilization of the system is substantially enhanced by the developed algorithm. The current algorithm achieves a utilization rate of 89.1%, whereas the developed algorithm demonstrates a remarkable improvement with a utilization rate of 95.13%. This showcases the developed algorithm's ability to better utilize available resources, minimizing empty gaps between parcels and maximizing the system's capacity. Finally, a comparison of the infeed throughputs between the two algorithms (as shown in Figure 7(c) and Figure 8(c)) indicates that the developed algorithm achieves more balanced infeed throughputs when compared to the industry-level algorithm.



((a)) 3-minute average of Merge flow rate

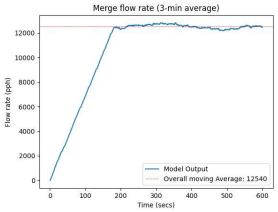


((b)) 3-minute average of Merge Utilization using

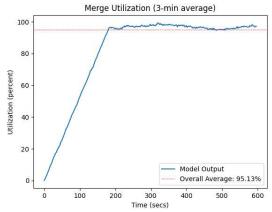


((c)) 3-minute average of Infeed imbalance

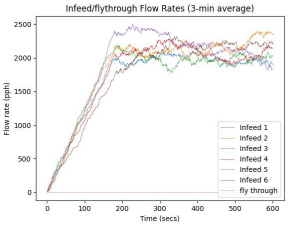
Fig. 7. KPIs using the current industry-level control algorithm in no-fly-through case



((a)) 3-minute average of Merge flow rate



((b)) 3-minute average of Merge Utilization



((c)) 3-minute average of Infeed imbalance

Fig. 8. KPIs using the Developed DP-based control algorithm in no-fly-through case

Scenario 2: With 10% Fly-through Parcels

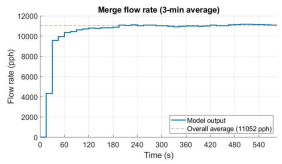
(1) All six infeeds operational
(2) Only the first two infeeds operational
(3) Only four infeeds operational

Table 2
10% fly through all cases

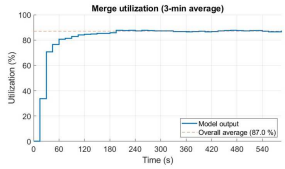
Scenario	Current Algorithm	Current Algorithm	Developed Algorithm	Developed Algorithm
10% fly through	Throughput [pph]	Utilization [%]	Throughput [pph]	Utilization [%]
6 infeeds	11052	87	12597	95.11
4 infeeds	10084	78.6	11977	89.28
2 infeeds	6972	55.5	8063	60.42

Case 1: Six infeeds operating with a 10% fly-

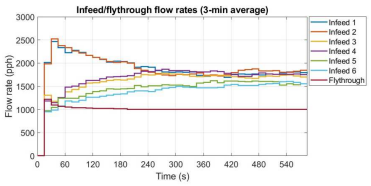
through occurrence rate The comparison of the two algorithms provides valuable insights into their performance regarding throughput and utilization. The results indicate significant advantages of the developed algorithm over the current algorithm. The developed algorithm achieved a higher throughput of 12,597 parcels, compared to the current algorithm's throughput of 11,052 parcels, demonstrating its efficiency in processing a larger volume of parcels within the given time frame. Additionally, the utilization of the merge conveyor improved substantially with the developed algorithm, reaching a utilization rate of 95.11% compared to the current algorithm's rate of 87%. This improvement suggests that the developed algorithm optimizes space utilization on the merge conveyor. Regarding imbalance, both algorithms exhibit similar patterns, although the current algorithm initially has higher upstream infeed throughput during the simulation compared to the developed algorithm (Figure 9(c)).



((a)) 3-minute average of Merge flow rate



((b)) 3-minute average of Merge Utilization using

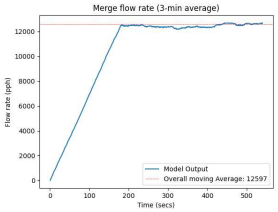


((c)) 3-minute average of Infeed imbalance

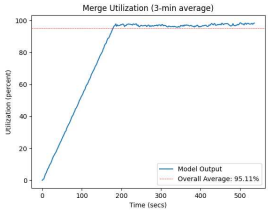
Fig. 9. KPIs using the current industry-level control algorithm in no-fly-through case

From the analysis presented in both Table 1 and Table 2, it becomes evident that the algorithm based on dynamic programming (DP) has exhibited a substantial degree of superiority over the algorithm utilized at the industry level, both in terms of utilization and throughput. This noteworthy enhancement can be attributed to the meticulous planning principles employed by the DP-based algorithm.

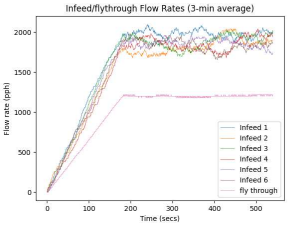
In the algorithm currently employed by the industry, upon the detection and announcement of a parcel by the infeed controller, the search algorithm of the merge controller actively seeks available space along the length of the merge conveyor prior to the designated infeed location of the parcel. Consequently, this approach results in a significant amount of unutilized space that cannot accommodate additional parcels. In contrast, the DP-based algorithm adopts a more efficient approach by employing a fixed segment size that initiates before the



((a)) 3-minute average of Merge flow rate



((b)) 3-minute average of Merge Utilization



((c)) 3-minute average of Infeed imbalance

Fig. 10. KPIs using the Developed DP-based control algorithm in no-fly-through case

merge conveyor.

During the process of segment generation, the DP-based algorithm strives to optimize the utilization of available parcels while also considering fixed gaps within the segment. This strategic approach effectively mitigates the issue of unused gaps, leading to better utilization of space and subsequently improving overall throughput.

The development of a new algorithm requires careful consideration beyond its creation. Factors such as cost implications and practicality play a significant role in determining its successful implementation. Conducting a cost-benefit analysis is essential to assess the economic feasibility and potential advantages of adopting the new algorithm.

According to Mishan and Quah (2020), Cost-benefit analysis (CBA) is a powerful decision-making tool used across various fields. It systematically evaluates the costs and benefits of different options, considering both monetary and non-monetary factors. In the material handling industry, where the adoption of new software solutions is infrequent, conducting a cost-benefit analysis is crucial for informed decision-making (Erdogmus 2007).

Reliability and robustness are crucial aspects of a newly developed algorithm (Azar et al. 2021),. Reliability ensures consistent and precise performance across diverse conditions, while robustness enables optimal performance in the presence of disturbances or uncertainties. Compatibility with widely used hardware systems, such as Programmable Logic Controllers (PLCs), is also essential for seamless integration.

Developing a new algorithm incurs costs in terms of time and resources (Arm et al. 2018). Operational costs associated with employee salaries and additional expenses for testing and validation should be considered. Speaking to an expert at the company, justifying the investment in a new algorithm requires demonstrating a performance improvement of at least five per cent compared to the current algorithm.

Improved performance not only provides a competitive edge but also leads to higher turnover rates and increased revenue (Shaw 2011). It allows for system upgrades and adaptation to customer needs and industry trends. Based on the comprehensive comparison of KPIs, the developed algorithm consistently outperforms the current industry-level algorithm. Thus, implementing the developed algorithm can enhance efficiency and process performance, contributing to the optimization of system operations.

5 Conclusion and discussion

By implementing dynamic programming as a control algorithm, the utilization of the merge conveyor can be significantly improved compared to other techniques. This research emphasizes the importance of balancing utilization and load distribution. A max heap algorithm is employed to ensure balanced loads among the infeeds. While this study focuses on the implementation of dynamic programming to enhance merge conveyor utilization, it serves as an initial exploration of this control algorithm.

In discussing the results and methodology, it can be concluded that the chosen method successfully addresses the main research question, enhancing overall system utilization. However, the research could have also explored the potential for improving utilization with the existing industry-level algorithm in different layouts which was left out of the scope. Assumptions made in the study, such as abstracting velocity profiles and software selection, may impact the obtained results and should be validated in practical scenarios. A limitation using this approach will be prevalent when the downstream infeeds are located further downstream in the system, thereby increasing the waiting times of the parcels on those infeeds.

The current model's prioritization using the maximum filled queue technique and all the infeeds having the same capacity, may not always hold true in real-time situations. Different priority-based techniques could yield slightly different throughput outcomes. While the utilization remains consistent in most cases, certain layout configurations or infeed positions can affect the developed algorithm's performance. Additionally, testing the model with different gap modes is recommended to fully understand its behaviour and potential variations in results as only a fixed safety gap of 15cm was considered in between each parcel.

6 Future Research

The developed algorithm aimed to closely simulate real-world conditions, but certain assumptions played a crucial role. Recommendations and remarks based on the research findings include considering the position and velocity of infeed conveyors to avoid long waiting times for parcels on downstream infeeds. It is important to conduct additional experiments to evaluate the algorithm's behaviour in more realistic scenarios. Factors such as parcel weight and kinematic constraints should be considered as they can impact system performance. Early knowledge of the next parcel (parcel announcement) can improve slice allocation and system efficiency. Future research should focus on sustainability aspects and explore dynamic control techniques. Costs should be carefully analyzed, taking into account potential overlooked expenses. Finally, this control algorithm can be tested on different layouts and different sorters to identify if it can perform well in all situations.

References

Agrawal, G.K., and S.S. Heragu. 2006. "A survey of automated material handling systems in 300-mm SemiconductorFabs." *IEEE Transactions on Semiconductor Manufacturing* 19 (1): 112–120. https://doi.org/10.1109/TSM.2005.863217.

Ahn, H, D Del Vecchio - IEEE Transactions on Automatic, and undefined 2017. 2017. "Safety verification and control for collision avoidance at road intersections." *ieeexplore.ieee.org,* https://ieeexplore.ieee.org/abstract/document/7987071/.

Arm, J., F. Zezulka, Z. Bradac, P. Marcon, V. Kaczmarczyk, T. Benesl, and T. Schroeder. 2018. "Implementing Industry 4.0 in Discrete Manufacturing: Options and Drawbacks." 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018, *IFAC-PapersOnLine* 51 (6): 473–478. ISSN: 2405-8963. https://doi.org/https://doi.org/10.1016/j.ifacol.2018.07.106.

Azar, Ahmad Taher, Fernando E. Serrano, Anis Koubaa, Habiba A. Ibrahim, Nashwa Ahmad Kamal, Alaa Khamis, Ibraheem Kasim Ibraheem, et al. 2021. "Robust fractional-order sliding mode control design for UAVs subjected to atmospheric disturbances." *Unmanned Aerial Systems: Theoretical Foundation and Applications: A Volume in Advances in Nonlinear Dynamics and Chaos (ANDC)* (January): 103–128. https://doi.org/10.1016/B978-0-12-820276-0.00012-1.

Babulak, Eduard, and Ming Wang. 2010. *Discrete Event Simulation: State of the Art.* 1–9. August. ISBN: 978-953-307-115-2. https://doi.org/10.13140/RG.2.1.2068.1767.

Banks, Jerry. 1999. "Introduction to simulation." *Winter Simulation Conference Proceedings* 1:7–13. ISSN: 02750708. https://doi.org/10.1145/324138.324142.

Blackwell, David. 1962. "Discrete Dynamic Programming." *The Annals of Mathematical Statistics* 33 (2): 719–726. ISSN: 0003-4851. https://doi.org/10.1214/AOMS/1177704593.

Cardenas, Ivan Dario, Wouter Dewulf, Thierry Vanelslander, Christophe Smet, and Joris Beckers. 2017. "The e-commerce parcel delivery market and the implications of home B2C deliveries vs pick-up points." *The e-commerce parcel delivery market and the implications of home B2C deliveries vs pick-up points* 44 (2): 235–256. ISSN: 03035247. https://doi.org/10.19272/201706702004.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition.* 3rd. The MIT Press. ISBN: 0262033844.

Cross, Tony, John Sutton, and Donald F. Wood. 1986. "Book reviews." *Transportation Planning and Technology* 11 (1): 81–86. https://doi.org/10.1080/03081068608717331.

Drissi Elbouzidi, Adnane, Abdessamad Ait El Cadi, Robert Pellerin, Samir Lamouri, Estefania Tobon Valencia, and Marie-Jane Bélanger. 2023. "The Role of AI in Warehouse Digital Twins: Literature Review." *Applied Sciences* 13 (11). ISSN: 2076-3417. https://doi.org/10.3390/app13116746.

Erdogmus, Hakan. 2007. "Cost-Benefit Analysis of Software Development Techniques and Practices." In *29th International Conference on Software Engineering (ICSE'07 Companion),* 178–179. https://doi.org/10.1109/ICSECOMPANION.2007.28.

Fedtke, Stefan, and Nils Boysen. 2014. "Layout Planning of Sortation Conveyors in Parcel Distribution Centers." *https://doi.org/10.1287/trsc.2014.0540* 51 (1): 3–18. ISSN: 15265447. https://doi.org/10.1287/TRSC.2014.0540.

Haneyah, S. W.A., J. M.J. Schutten, P. C. Schuur, and W. H.M. Zijm. 2013. "Generic planning and control of automated material handling systems: Practical requirements versus existing theory." *Computers in Industry* 64 (3): 177–190. ISSN: 0166-3615. https://doi.org/10.1016/J.COMPIND.2012.11.003.

Haneyah, Sameh, Johann Hurink, Marco Schutten, Henk Zijm, and Peter Schuur. 2011. "Planning and Control of Automated Material Handling Systems: The Merge Module," 281–286. https://doi.org/10.1007/978-3-642-20009-0_45.

Hoven, Matthijs van den. 2019. *Utilization Improvement of a Sortation System for the Parcel Industry.* Technical report. Technical report, Eindhoven University of Technology, the Netherlands.

Jing, GG, WD Kelton, JC Arantes - ... Proceedings (Cat. No ..., and undefined 1998. 1998. "Modeling a controlled conveyor network with merging configuration." *ieeexplore.ieee.org,* https://ieeexplore.ieee.org/abstract/document/745851/.

Johnstone, M, D Creighton, S Nahavandi - Simulation Modelling Practice, and undefined 2015. 2015. "Simulation-based baggage handling system merge analysis." *Elsevier,* https://www.sciencedirect.com/science/article/pii/S1569190X15000131?casa_token=aiFNCiNpdY8AAAAA:y9LcU6KK7zLTl4EbffJknIMObhn1dbp3ks9azy8gpTEoi18zBj-W6FnEWR9bK1fvJkpe6j2wRdA.

Kellner, Marc I., Raymond J. Madachy, and David M. Raffo. 1999. "Software process simulation modeling: Why? What? How?" *Journal of Systems and Software* 46 (2-3): 91–105. ISSN: 0164-1212. https://doi.org/10.1016/S0164-1212(99)00003-5.

Kim, Gukhwa, Junbeom Kim, and Junjae Chae. 2017. "Balancing the baggage handling performance of a check-in area shared by multiple airlines." *Journal of Air Transport Management* 58 (January): 31–49. ISSN: 0969-6997. https://doi.org/10.1016/J.JAIRTRAMAN.2016.08.017.

Li, L, FY Wang - IEEE Transactions on Vehicular technology, and undefined 2006. 2006. "Cooperative driving at blind crossings using intervehicle communication." *ieeexplore.ieee.org,* https://ieeexplore.ieee.org/abstract/document/4012536/?casa_token=OUnNkqIdre4AAAAA:KRrBMfBhy8LhRp-WNCoWD7Dv-VBD0PvL7q7R8LD1L-DATh_dEFMbqXqVQdAioslqtPPCi61Adg.

Li, PT, and X Zhou Methodological. 2017. "Recasting and optimizing intersection automation as a connected-and-automated-vehicle (CAV) scheduling problem: A sequential branch-and-bound search." *Elsevier,* https://www.sciencedirect.com/science/article/pii/S0191261517304782?casa_token=q_Aw0jK_tUoAAAAA:ocGlFvErY4ARGMRvuba4FCFwBaqrBfweiHmXSL7ZMEnz3ZY1bk3Gx-Xv9LAY1t_LJBaVpWEPnvU.

Lin, Shang Chien, Hsiang Hsu, Yi Ting Lin, Chung Wei Lin, Iris Hui Ru Jiang, and Changliu Liu. 2020. "A Dynamic Programming Approach to Optimal Lane Merging of Connected and Autonomous Vehicles." *IEEE Intelligent Vehicles Symposium, Proceedings,* 349–356. https://doi.org/10.1109/IV47402.2020.9304813.

Marinescu, Dan, Jan Čurn, Mélanie Bouroche, and Vinny Cahill. 2012. "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach." *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC,* 900–906. https://doi.org/10.1109/ITSC.2012.6338779.

Meens, Jasper. 2017. "Model Based Design Approach for Merge Balancing," Eindhoven University of Technology, the Netherlands.

Meng, Yue, Li Li, Fei Yue Wang, Keqiang Li, and Zhiheng Li. 2018. "Analysis of Cooperative Driving Strategies for Nonsignalized Intersections." *IEEE Transactions on Vehicular Technology* 67 (4): 2900–2911. ISSN: 00189545. https://doi.org/10.1109/TVT.2017.2780269.

Mishan, E.J., and Euston Quah. 2020. "Cost-Benefit Analysis" (August). https://doi.org/10.4324/9781351029780.

Müller, Eduardo Rauh, Rodrigo Castelan Carlson, and Werner Kraus Junior. 2016. "Intersection control for automated vehicles with MILP." *IFAC-PapersOnLine* 49 (3): 37–42. ISSN: 24058963. https://doi.org/10.1016/J.IFACOL.2016.07.007.

Nelson, Randolph. 1995. "Markov Processes." *Probability, Stochastic Processes, and Queueing Theory,* 329–389. https://doi.org/10.1007/978-1-4757-2426-4_8. https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-1-4757-2426-4_8.

Peeters, K. 2015. "Balancing Control of Material Handling Systems." PhD diss., Eindhoven University of Technology, the Netherlands.

Peffers, Ken, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. 2014. "A Design Science Research Methodology for Information Systems Research." *https://doi-org.tudelft.idm.oclc.org/10.2753/MIS0742-1222240302* 24 (3): 45–77. ISSN: 07421222. https://doi.org/10.2753/MIS0742-1222240302.

Pei, Huaxin, Shuo Feng, Yi Zhang, and Danya Yao. 2019. "A Cooperative Driving Strategy for Merging at On-Ramps Based on Dynamic Programming." *IEEE Transactions on Vehicular Technology* 68 (12): 11646–11656. ISSN: 19399359. https://doi.org/10.1109/TVT.2019.2947192.

Pei, Huaxin, Yuxiao Zhang, Yi Zhang, and Shuo Feng. 2022. "Optimal Cooperative Driving at Signal-Free Intersections With Polynomial-Time Complexity." *IEEE Transactions on Intelligent Transportation Systems* 23 (8): 12908–12920. ISSN: 15580016. https://doi.org/10.1109/TITS.2021.3118592.

Pfohl, Hans Christian, Pascal Wolff, and Johannes Kern. 2020. "Transshipment hub automation in China's courier/express/parcel sector." *Urban Freight Transportation Systems* (January): 163–180. https://doi.org/10.1016/B978-0-12-817362-6.00009-4.

Python Core Team. 2019. *Python: A dynamic, open source programming language.* Python Software Foundation. https://www.python.org/.

Sargent, Robert. 2011. "Verification and validation of simulation models," 37:166–183. January. https://doi.org/10.1109/WSC.2010.5679166.

Sharma, Prateek. 2015. "Discrete-Event Simulation." *INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH* 4 (04). ISSN: 2277-8616. www.ijstr.org.

Shaw, Jason D. 2011. "Turnover rates and organizational performance." *http://dx.doi.org/10.1177/2041386610382152* 1 (3): 187–213. ISSN: 2041-3866. https://doi.org/10.1177/2041386610382152.

Stewart, William J. 2009. "Probability, Markov Chains, Queues, and Simulation." *(No Title)* (July). https://doi.org/10.2307/J.CTVCM4GTC.

Thacker, B.H., S.W. Doebling, F.M. Hemez, M.C. Anderson, J.E. Pepin, and E.A. Rodriguez. 2004. *Concepts of Model Verification and Validation.* http://inis.iaea.org/Search/search.aspx?orig_q=RN:36030870.

Thorne, David R. 2006. "Throughput: A simple performance index with desirable characteristics." *Behavior Research Methods* 38 (4): 569–573. ISSN: 1554351X. https://doi.org/10.3758/BF03193886/METRICS.

B

Simulation Model

B.1. Simulation Process

The utilization of a Process Description Language is essential to describe the processes employed in the model. This language, expressed in pseudo-code, provides a comprehensive representation of the aforementioned processes. It is worth noting a couple of important aspects. Firstly, the frequent use of ”self.xxx” in the descriptions, where ”self” refers to the actor itself, while ”xxx” denotes an attribute, function, or process associated with the actor. Secondly, functions and processes are enclosed in parentheses ’()’, allowing for the inclusion of arguments as inputs. In some cases, ”self” is used as an argument, indicating that another actor’s process refers to the initial actor and can modify its attributes as needed.

B.1.1. Infeed Class

The Infeed class, algorithm 1, represents a data structure for managing parcels in a queue with a specified capacity. It has various methods for adding parcels, calculating the used space, moving parcels from the queue to allotted slots, and removing parcels from allotted slots. The class keeps track of the parcels, capacity, remaining percentage of the queue, held parcels, removed parcels, slot allotments, and added parcels.

Algorithm 1 Infeed Class

```
Input: number, capacity
Output: An instance of Infeed
number ← number
parcels ← []
capacity ← capacity
remaining_percent ← 100.0
held_parcels ← []
removed_parcels ← []
slot_alloted ← []
added_parcels ← []

function calculate_used_space()
    Input: None
    Output: The used capacity of the queue
    used_capacity ← 0
    for each parcel in parcels
        used_capacity ← used_capacity + parcel[0] + safety_gap
    end for
    for each parcel in slot_alloted
        used_capacity ← used_capacity + parcel[0] + safety_gap
    end for
    return used_capacity - safety_gap
end function

function add_parcel(parcel_length, id)
    Input: parcel_length, id
    Output: None
    current_used_capacity ← calculate_used_space()
    used_capacity_after_parcel_added ← current_used_capacity + parcel_length + safety_gap
    remaining_percent_after_parcel_added ← 100.0 - ((used_capacity_after_parcel_added * 100) /
capacity)
    if remaining_percent_after_parcel_added < 0 then
        hold until remaining_percent > parcel_length
        held_parcels.append([parcel_length, id])
        return
    else
        parcels.append([parcel_length, id])
        added_parcels.append([parcel_length, id])
        remaining_percent ← remaining_percent_after_parcel_added
    end if
end function
=0
```

Algorithm 2 Infeed Class (Continued)

```
function move_to_alloted()
    Input: None
    Output: None
    if not parcels then
        return
    end if
    removed_parcel ← parcels[0]
    slot_alloted.append(removed_parcel)
    parcels.pop(0)
end function

function remove_parcel()
    Input: None
    Output: None
    if not slot_alloted then
        return
    end if
    removed_parcel ← slot_alloted[0]
    removed_parcels.append(removed_parcel)
    slot_alloted.pop(0)
    current_used_capacity ← calculate_used_space()
    remaining_percent ← 100.0 - ((current_used_capacity * 100) / capacity)
end function =0
```

B.1.2. Merge Conveyor Class

The following code, algorithm 3, defines a class called Merge Conveyor representing a merge conveyor system. It contains an initialization method that sets the number of infeeds, infeed capacity, and parcel gap. The class includes various functions to generate random parcel lengths and unique IDs, fill the infeeds with parcels before the algorithm starts, generate fly-through parcels, generate parcels for the infeeds, and generate sequences of parcels using different approaches like heap and dynamic programming. Additionally, there are functions to handle the entry and exit of parcels within a specified time frame. It is worth mentioning that only the necessary parts of the code are elaborated to make it better for understanding purposes.

Algorithm 3 Merge Conveyor Class

```
Input: num_infeeds, infeed_capacity, parcel_gap
Output: An instance of MergeConveyor
num_infeeds ← num_infeeds
infeeds ← [Infeed(i, Infeed_capcaity) for i in range(num_infeeds)]
merge_conveyor_parcels ← []
parcel_entries ← [[] for _ in range(num_infeeds)]
parcel_exits ← [[] for _ in range(num_infeeds)]
fly_through_entries ← []
fly_through_locations ← []
id_count ← 1

function generate_random_parcel_length(max_length, min_length, mean_parcel_length, std_dev)
    Input: max_length, min_length, mean_parcel_length, std_dev
    Output: Randomly generated transport parcel length
    ... (Code for generating parcel length)

function generate_id()
    Input: None
    Output: Generated ID
    ... (Code for generating unique ID)

function generate_fly_through_parcels()
    Input: None
    Output: None
    ... (Code for generating fly-through parcels)

function generate_parcels()
    Input: None
    Output: None
    ... (Code for generating parcels)
=0
```

Algorithm 4 Merge Conveyor Class (continued)

```
function generate_sequence_using_heap(last_parcel_trailing_edge, segment_size)
    Input: last_parcel_trailing_edge, segment_size
    Output: Sequence array
    infeeds_copy ← a copy of infeeds' parcels
    infeeds_heap ← an empty heap
    indices ← an array of zeros
    sequence_array ← an empty array
    segment_left ← segment_size

    for i ← 0 to NUM_INFEEDS - 1
        infeed_sum ← 0
        for each parcel in infeeds_copy[i][indices[i]:]
            infeed_sum ← infeed_sum + parcel[0]
        end for
        append [infeeds[i].capacity - infeed_sum, i] to infeeds_heap
    end for
    heapify infeeds_heap

    while segment_left > 0
        [most_filled_queue_remaining_percent, most_filled_queue_number] ← pop infeeds_heap
        if last_parcel_infeed = most_filled_queue_number
            break
        if not infeeds_copy[most_filled_queue_number]
            break
        if indices[most_filled_queue_number] ≥ length of infeeds_copy[most_filled_queue_number]

            break
        released_parcel_length, id ← infeeds_copy[most_filled_queue_number][indices[most_filled_queue_number]]

        if released_parcel_length ≤ segment_left
            append [last_parcel_trailing_edge, last_parcel_trailing_edge + released_parcel_length,
most_filled_queue_number, id] to sequence_array
            last_parcel_trailing_edge ← last_parcel_trailing_edge + released_parcel_length

        increment indices[most_filled_queue_number] by 1
        infeeds_heap ← an empty heap
        for i ← 0 to NUM_INFEEDS - 1
            infeed_sum ← 0
            for each parcel in infeeds_copy[i][indices[i]:]
                infeed_sum ← infeed_sum + parcel[0]
            end for
            append [infeeds[i].capacity - infeed_sum, i] to infeeds_heap
        end for
        heapify infeeds_heap
        segment_left ← segment_left - released_parcel_length
    end while
    return sequence_array and end function
=0
```

Algorithm 5 Merge Conveyor Class (continued)

```
function generate_DP_sequence_array(last_parcel_trailing_edge, segment_size, fun, last_parcel_infeed)
    Input: last_parcel_trailing_edge, segment_size, fun, last_parcel_infeed
    Output: Sequence array
    sequence_array ← []
    ans ← fun(segment_size, [0] * 6, "", {}, last_parcel_infeed)
    sequence ← ans[1]
    indices ← [0] * 6
    for idx in sequence:
        idx ← int(idx)
        parcel, id ← self.infeeds[idx].parcels[indices[idx]]
        sequence_array.append([last_parcel_trailing_edge, last_parcel_trailing_edge + parcel, idx, id])
        last_parcel_trailing_edge += parcel
        indices[idx] += 1
    return sequence_array

function get_min_gap_sequence_DP(gap_left, indices, sequence, memo, last_used_infeed)
    Input: gap_left, indices, sequence, memo, last_used_infeed
    Output: Min gap sequence
    Procedure:
        if gap_left is already computed in the memo table then
            Return the stored value for gap_left
        End If
        Initialize final_ans_gap as gap_left and final_ans_sequence as an empty string
        for each queue from 0 to NUM_INFEEDS do
            if last_used_infeed is the same as the current queue then
                Continue to the next iteration of the loop
            End If
            if queue index is out of range or the parcel length at the queue index is greater than gap_left then
                Continue to the next iteration of the loop
            End If
            Get the parcel length and id from the queue at the current index
            Increase the index of the current queue by 1
            Recursively call get_min_gap_sequence_DP with reduced gap_left, updated indices, appended sequence, memo, and current queue as last_used_infeed
            if the returned gap is smaller than final_ans_gap then
                Update final_ans_gap with the returned gap and final_ans_sequence with the returned sequence
        End For
        Store final_ans_gap and final_ans_sequence in the memo table for the current gap_left, indices, and last_used_infeed
        Return final_ans_gap and final_ans_sequence
=0
```

Algorithm 6 MergeConveyor Class (continued)

```
function get_min_gap_sequence_DP_with_heap(gap_left, indices, sequence, memo, last_used_infeed)
    Input: gap_left, indices, sequence, memo, last_used_infeed
    Output: Min gap sequence
    Procedure:
        if gap_left is already computed in the memo table then
            Return the stored value for gap_left
        Initialize final_ans_gap as gap_left and final_ans_sequence as an empty string
        Get the queue order based on the current indices and last_used_infeed
        for each queue in the queue order do
            if queue index is out of range or the parcel length at the queue index is greater than gap_left then
                Continue to the next iteration of the loop
            Get the parcel length and id from the queue at the current index
            Increase the index of the current queue by 1
            Recursively call get_min_gap_sequence_DP_with_heap with reduced gap_left, updated indices, appended sequence, memo, and current queue as last_used_infeed
            if the returned gap is smaller than final_ans_gap then
                Update final_ans_gap with the returned gap and final_ans_sequence with the returned sequence
        Store final_ans_gap and final_ans_sequence in the memo table for the current gap_left, indices, and last_used_infeed
        Return final_ans_gap and final_ans_sequence
function get_order(indices, last_used_infeed)
    Input: indices, last_used_infeed
    Output: Queue order
    ... (Code for getting the order of queues)

function handle_entry_exit_parcels_in_time_frame(start_time, end_time)
    Input: start_time, end_time
    Output: None
    ... (Code for handling entry and exit parcels within a time frame)

while time ≤ SIMULATION_TIME
    ... (Code for the simulation loop)
=0
```

B.2. Model Verification

Check 1 and 2: For various segment sizes, the simulation was performed multiple times to verify if the maximum size of the segment exceeds the specified cap. From the Figure B.1, it is apparent that, for a segment setting of 200 cm, the difference between the start and end of the segment does not exceed 200 cm. Additionally, as expected, the next segment starts immediately after the previous segment ends. It is also evident that no two slices are consecutively assigned to the same infeed. This implies that the constraint on inter-departure times is not violated.

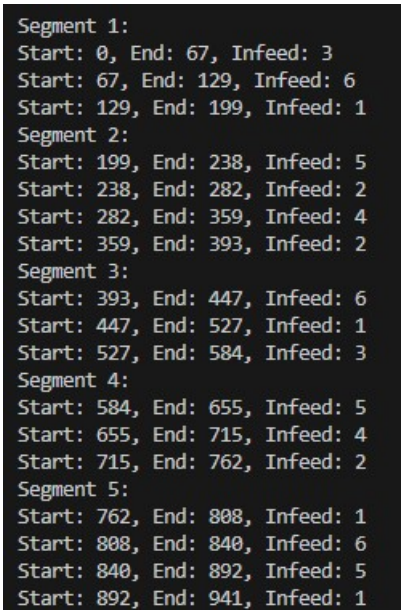


Figure B.1: Slice allocation in the segment

Check 3: Based on a simulation, the times at which parcels enter and exit are sampled and recorded in the Table B.1. As explained in item 4.1.1, the parcel entries and exits are stored as a list containing the time and length information of each incoming or outgoing parcel from the infeed. This test assists in determining if the parcel can reach the merge before the designated segment slice reaches the infeed.

Table B.1: Parcel entry and exit verification

Parcel entries	Parcel exits
[4.4 , 68, id-35]	[5.69 , 68, id-35]
[5.45 , 63, id-39]	[7.755 , 63, id-39]
[8.06, 58, id-51]	[9.265, 58, id-51]
[1.87 , 51, id-22]	[4.773 , 51, id-22]
[2.27 , 76, id-23]	[7.232 , 76, id-23]
[6.08 , 65, id-42]	[8.932 , 65, id-42]

B.3. Model Validation

B.3.1. Extreme Condition tests

As discussed in subsection 4.1.4, in extreme conditions the model should perform as expected. To do so, the following tests were performed and checked if the expected outcomes are met or not.

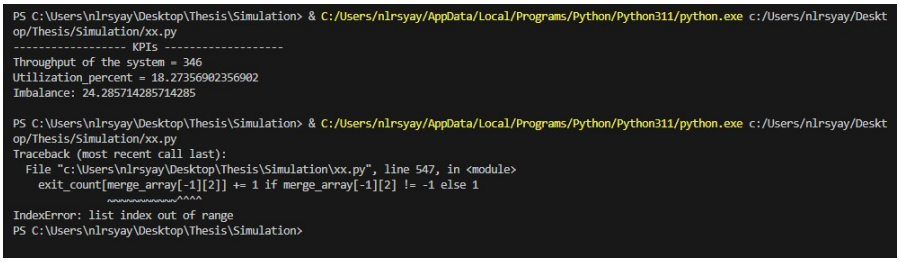


Figure B.2: Both extreme condition tests

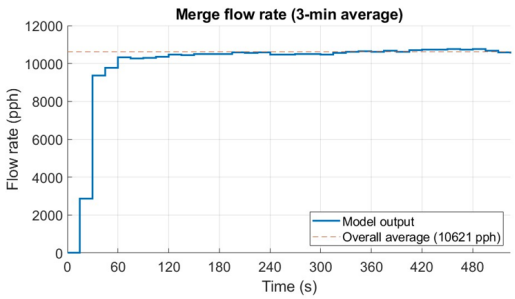
C

Model Experimentation

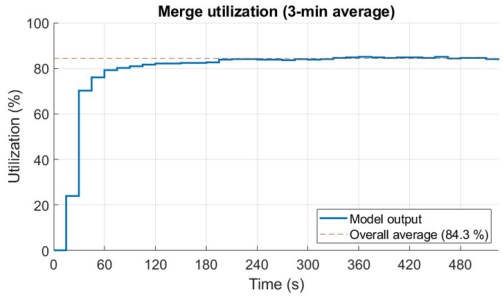
C.1. Scenario: No flythrough

C.1.1. Case 3: No fly-through and 4 infeeds

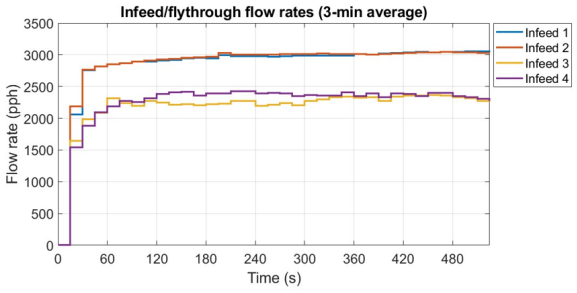
Current FCFS Control Algorithm



(a) 3-minute average of Merge flow rate



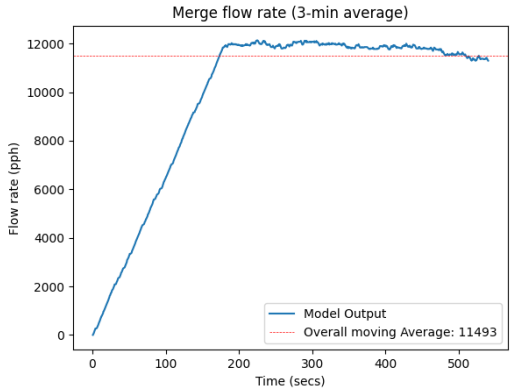
(b) 3-minute average of Merge Utilization



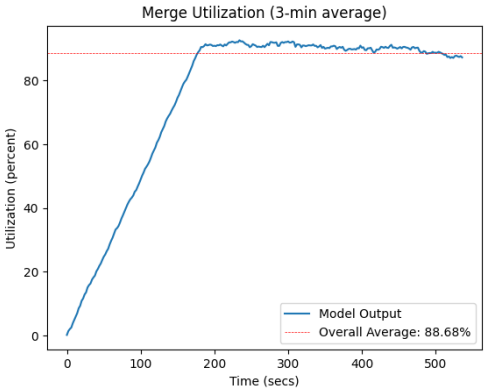
(c) 3-minute average of Infeed imbalance

Figure C.1: 3-minute average graphs of KPIs using current control algorithm

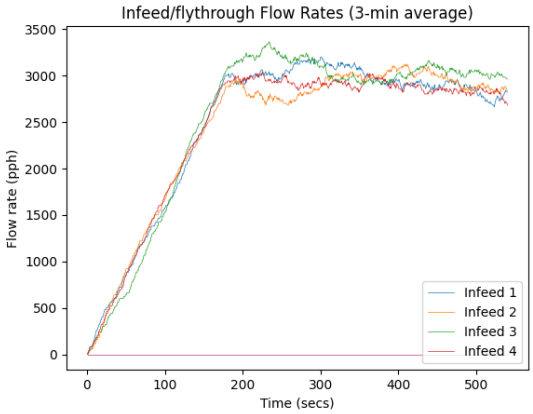
Developed DP-based Algorithm



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization

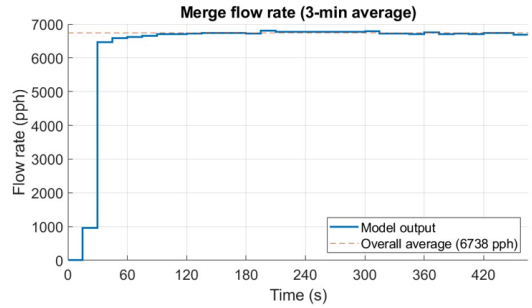


(c) 3-minute average of Infeed imbalance

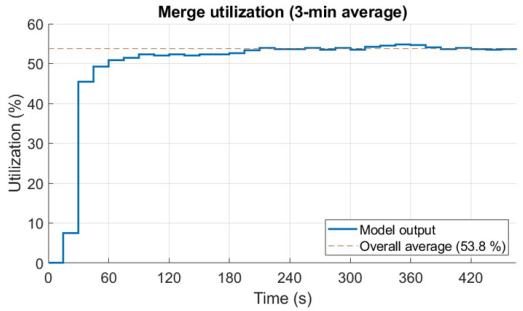
Figure C.2: 3-minute average graphs of KPIs using developed control algorithm

C.1.2. Case 4: No fly-through and 2 infeeds

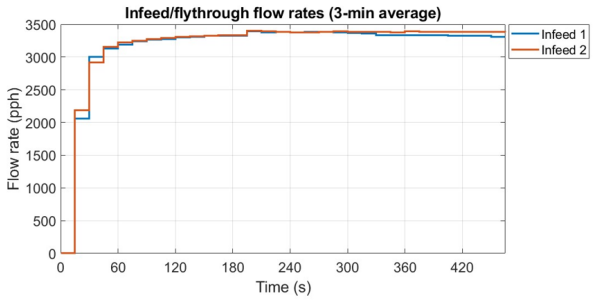
Current FCFS Control Algorithm



(a) 3-minute average of Merge flow rate



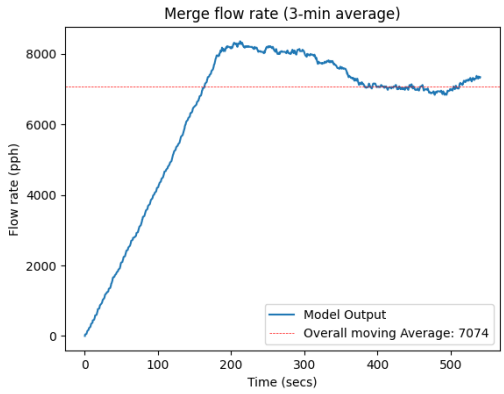
(b) 3-minute average of Merge Utilization



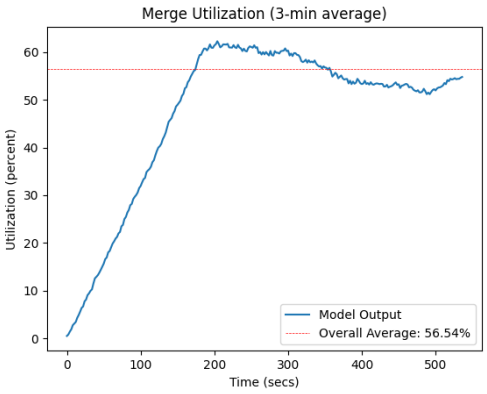
(c) 3-minute average of Infeed imbalance

Figure C.3: 3-minute average graphs of KPIs using current control algorithm

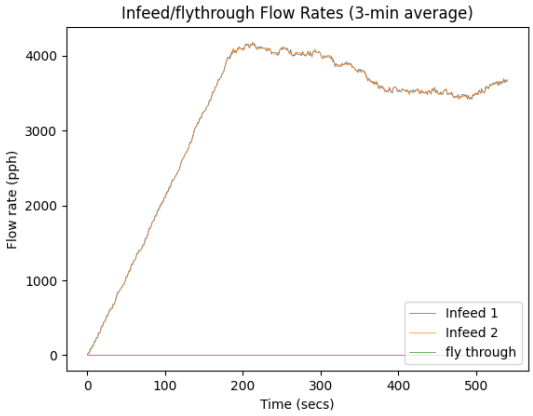
Developed DP-based Algorithm



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization



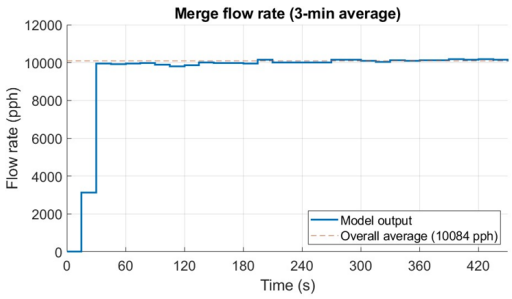
(c) 3-minute average of Infeed imbalance

Figure C.4: 3-minute average graphs of KPIs using developed control algorithm

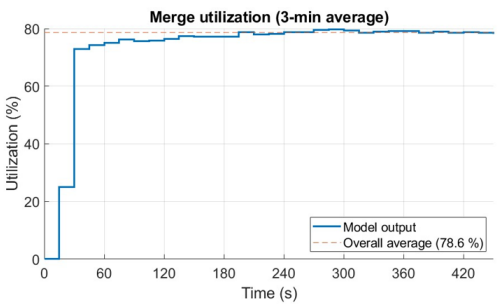
C.2. Scenario: 10% fly-though

C.2.1. Scenario 7: 10 % fly-through and 4 infeeds

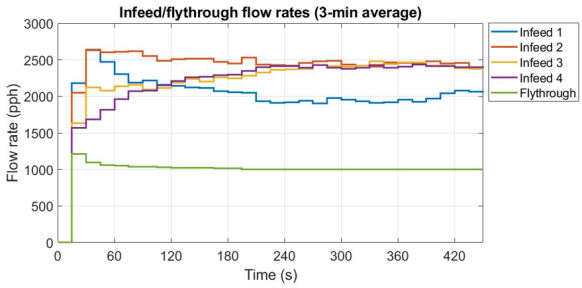
Current FCFS Control Algorithm



(a) 3-minute average of Merge flow rate



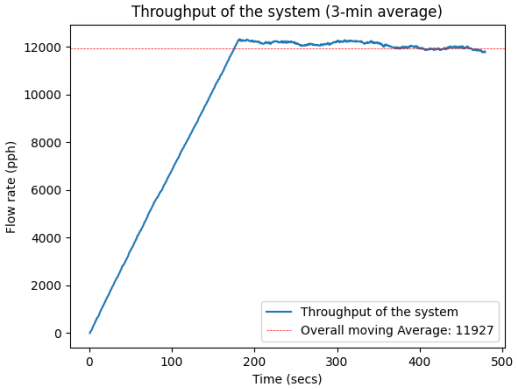
(b) 3-minute average of Merge Utilization



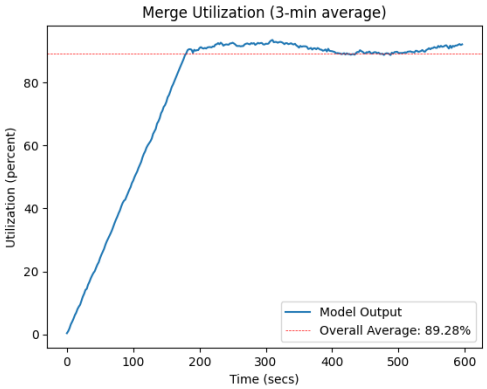
(c) 3-minute average of Infeed imbalance

Figure C.5: 3-minute average graphs of KPIs using current control algorithm

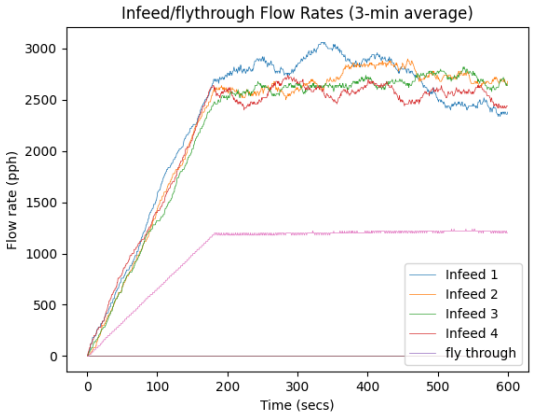
Developed DP-based Algorithm



(a) 3-minute average of Merge flow rate



(b) 3-minute average of Merge Utilization



(c) 3-minute average of Infeed imbalance

Figure C.6: 3-minute average graphs of KPIs using developed control algorithm