Model-Based Control for Postal Automation and Baggage Handling

A.N. Tarău

Model-Based Control for Postal Automation and Baggage Handling

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College van Promoties,
in het openbaar te verdedigen op dinsdag 19 januari 2010 om 10:00 uur
door Alina Nicoleta TARĂU,
Inginer Diplomat, Technische Universiteit Boekarest, Roemenië
geboren te Tecuci, Roemenië.

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr.ir. J. Hellendoorn
Prof.dr.ir. B. De Schutter

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. J. Hellendoorn	Technische Universiteit Delft, promotor
Prof.dr.ir. B. De Schutter	Technische Universiteit Delft, promotor
Prof.dr. E.F. Camacho	Technische School Seville
Prof.dr.ir. M.P.C. Weijnen	Technische Universiteit Delft
Prof.dr.ir. M.B.M. de Koster	Erasmus Universiteit Rotterdam
Prof.dr.ir. G. Lodewijks	Technische Universiteit Delft
Ir. P. Jansz	VERBpeter (ex Siemens)

This thesis has been completed in partial fulfillment of the requirements of the Dutch Institute for Systems and Control (DISC) for graduate studies. The research described in this thesis was supported by the VIDI project "Multi-Agent Control of Large-Scale Hybrid Systems" (DWV.6188) of the Dutch Technology Foundation STW, Applied Science division of NWO and the Technology Programme of the Dutch Ministry of Economic Affairs.

TRAIL Thesis Series T2010/1, the Netherlands TRAIL Research School
P.O. Box 5017
2600 GA Delft
The Netherlands
T: +31 (0) 15 278 6046
E: info@rstrail.nl

Published and distributed by: A.N. Tarău
E-mail: alina.tarau@gmail.com

ISBN 978-90-5584-123-3

Keywords: postal automation, baggage handling, model-based control.

Printed in the Netherlands

Acknowledgments

This thesis is the result of the Ph.D. research that I have done at the Delft Center for Systems and Control within the Delft University of Technology. During the time I worked within this department I had the opportunity to learn, grow, and improve my analytical-thinking. For this I am grateful to various people who also contributed to this thesis in their own ways.

First of all I would like to thank my supervisors Hans Hellendoorn and Bart De Schutter for their guidance, criticism, and support during my Ph.D. research. I would not have been able to write this thesis without their continuous encouragement and inspiration. I also thank them for the freedom they left me when doing this research, for the time they took to read my work, and for their suggestions which helped a lot to improve my thinking. I wish to especially acknowledge Bart for showing me the beauty of using Linux and for helping me broaden my knowledge of LaTeX and Matlab.

It has been a delight for me to work with Ton van den Boom who offered me the opportunity to interact with students and to discover model predictive control.

I owe a lot of thanks to the members of my Ph.D. committee Eduardo Camacho, Margot Weijnen, René de Koster, Gabriel Lodewijks, and Peter Jansz who took the time to review this thesis. Their constructing comments and remarks helped me a lot and I am very grateful to them.

I enjoyed working with all the staff and colleagues at Delft Center for Systems and Control. I would like to thank Kitty Dukker, Ellen van den Berg-Moor, and Debby van Vondelen for being helpful and friendly every time when I had questions regarding forms and financial matters. Also thanks to Will van Geest, Arjan van Dijke, and Daan Noteboom who always answered promptly any question I had regarding ICT administrative issues. I have greatly appreciated the fruitful discussions and the friendship of Rudy Negenborn, Monique van den Berg, Eric Trottemant, Jan van Hulzen, Zsófia Lendek, Lucian Buşoniu, Sara van der Hoeven, Arturo Tejada, Lakshmi Baskar, Ali Mesbah, and Justin Rice.

I would also like to thank my husband, Thomas, for his love, support, and patience, and to my family at home for their continuous encouragement and for enduring my self-consciousness all these years.

Finally, I wish to apologize to all the persons who contributed in one way or another to my work and are not mentioned here, and to thank them all together.

Alina N. Tarău,
Delft, November 2009.

Contents

Chapter 1

Introduction

In this chapter we first present the motivation for the research addressed in this thesis. Next, we introduce the framework that we focus on together with the scope of this research. Finally, we give a short overview of the applications considered in this thesis and the main contributions.

1.1 Motivation

Transportation systems such as conveyor systems [55], traffic systems [15, 44], distribution systems [9, 49, 59], and others have always had and will continue to have a major impact on both our personal lives and society as a whole. From the earliest times we have relied on transportation systems to carry bulk resources, to get us to school or work, or to travel around the world. We have gone from horse-drawn carts and simple bicycles to high speed trains and space shuttles. What used to be considered a luxury (e.g., owning a car) is now a necessity. Also, there is an increasing need in developing safe, efficient, and reliable automated systems for transporting and sorting any kind of materials (see, e.g., [64] for systems that transport and sort fruits and vegetables).

We live in a time of continually increasing dependency on modern transportation systems. Also, due to the increasing need to transport and move faster, farther, and cheaper, we have become major users of transportation systems. Hence, the combination of the continuously increasing need for reduction of cost of the transport industry and rise of low-cost carriers requires a cost effective operation of these automated systems.

Let us now consider the applications that we focus on in this thesis, namely the postal automation in mail sorting centers and baggage handling in airports. One can notice during the last decades a considerable increase in the volume of magazines, catalogs, and plastic wrapped mail items that have to be handled by mail sorting centers. In the earliest times the process of sorting the mail involved a series of operations with human hands at work every step of the way. This manual process consumes a lot of time and human energy. Therefore, nowadays, state-of-the-art mail sorting centers are equipped with dedicated mail sorting machines in order to be able to handle the large volumes of mail. A similar need for automatization occurred also in airports where the continuing growth of the airport traffic made the manual operations of handling the baggage too expensive. Moreover, even the

conventional sorters based on conveyor belts [10] are becoming too slow in large and busy airports. Note that for large and busy airports, the baggage handling system is one of the most important factors that determines the airport's efficiency and reliability. Therefore, high speed transportation is required. To this aim, state-of-the-art baggage handling systems handle the baggage in an automated way using fast individual vehicles. These vehicles transport the bags at high speeds on a network of tracks [91].

We conclude the motivation for this research with the following remark. When the transportation demand continues to grow and the operation of transportation systems gets closer to its limits, one can invest in additional infrastructure, carriers, or sorting systems. As an alternative to solve this problem, in this research we investigate, develop, and design a more efficient operation of the considered transportation systems by employing state-of-the-art control methods [8] and optimization techniques [61] that also use domain specific knowledge.

1.2 Framework and scope of the thesis

In this thesis we focus on a specific class of transportation systems, characterized by materials[1] being processed while they are transported by conveyor systems or other transportation means[2] such as sorting machines, baggage handling, and distribution systems. These transportation systems have a common modeling framework since they are dynamic systems that exhibit both continuous and discrete dynamics. Hence, this class of transportation systems can be modeled as hybrid systems [58, 90]. Let us take as example the parts of these systems that consist of conveyor belts. Then the transport of materials on the conveyors can be modeled as a continuous process, characterized by, e.g., the speed of the conveyor, which can in principle be adjusted continuously. Actions like feeding an item on the belt, removing the item, rerouting it, etc. provide discrete actions on the system.

Next we present the scope of this study. Due to increasing demands, the focus of industry is shifting from ensuring safe and automated operation to ensuring quality, reliability, and performance maximization. But, typically, the performance of automated transportation systems is limited by mechanical capabilities (such as maximum speed of the transportation means), by the performance of the process devices (address reading devices, bar code reading devices, scanners, etc.), and also by the sorting and routing schemes. In this research we consider the mechanical capabilities and the performance of the process devices to be given.

Typical control problems of the specific class of transportation systems that we consider in this thesis — transportation systems handling materials — are the following: coordination and synchronization of the processing units, prevention of jams and deadlocks, prevention of buffer overflow, avoiding damage of the goods, maximization of performance, and cost minimization.

In this thesis we investigate methods that can be used to efficiently control the considered class of transportation systems so that their overall performance is maximized when taking into account the issues we have just enumerated — recall that the mechanical capa-

[1] We will not consider transportation systems for people, but only for materials.

[2] E.g. in large airports baggage is transported not only using conveyor systems, but also using fast individual vehicles.

bilities and the performance of the process devices are considered to be given; moreover, in this thesis we do not consider the problem of minimizing the costs. Currently, most higher-level control methods for these systems are based on centralized control and/or on ad-hoc techniques. But centralized control of large-scale systems is often not feasible in practice due to computational complexity, communication overhead, and lack of scalability, while using ad-hoc techniques, typically, does not yield the best possible performance of the system.

Note that in this thesis we consider only two applications of the class of transportation systems that handle materials, namely sorting machines in mail sorting centres and baggage handling in airports. However, the control approaches that we develop in this thesis are not restricted to the considered applications only, but they can similarly be applied to other transportation systems, e.g., power distribution systems and water management, automated guided vehicles in warehouses, or traffic systems.

1.3 Research overview

This section gives an overview of this research, emphasing the applications that we focus on.

Postal automation

First we discuss the postal automation application. There are two types of mail sorting machines, the first designed to process postcards and small letters, the second designed to handle large mail items such as newspapers, catalogs, and large letters. In this thesis we focus on the latter. These large mail items are shortly called “flats”. Briefly, a state-of-the-art flat sorting machine, operates as follows. First, the flats are fed into the machine via a feeding device. Then conveyor systems transport the flats with a constant speed towards the sorting part of the machine. Meanwhile, the stamp used for postage is voided, the address and the postal code are located, and the necessary information is extracted and printed on the flat in form of a bar code. This ensures a transport delay line of several seconds allowing the system to achieve sorting information on-line before the mail item reaches the code printing phase. Next, the flats (which have been previously identified via bar codes) are inserted into transport boxes by inserting devices; the boxes carry the pieces with constant speed and sort them into their destination bins, see, e.g., Figure 1.1 and Figure 1.2, according to the selected sorting scheme. Figure 1.1 illustrates the sorting part of a state-of-the-art flat sorting machine developed by Siemens. This flat sorting system consists of transport boxes at the top, one level of intermediate pockets that can hold several flats in order to sort the items into delivery sequence, and destination bins (the plastic bins of Figure 1.2).

The throughput of a basic system sketched above can be augmented by designing a system where the bottom part consisting of destination bins can move bidirectional with variable speed.

Then for the new system (where the bottom part can move) we will use simulation to determine a fast event-driven model. This model of the flat sorting system will then be used for model-based control. The goal of the model-based controllers is to compute the speed profile of the bin system that maximizes the throughput of the sorting machine.

Figure 1.1: Sorting part of a flat sorting machine. Picture source: Siemens AG, Infrastructure Logistics, 2009.

Figure 1.2: Dropping a flat into a bin. Picture source: Siemens AG, Infrastructure Logistics, 2009.

Problem statement One can state the control problem that we want to solve as follows. Given a buffer of flats known in advance due to the delay line of the flats' preparation phase, the optimal speed profile of the bin system has to be computed so that the throughput of the sorting machine is maximized.

Control approaches In order to compute the optimal speed profile of the bin system, we will implement and compare different variants of optimal control with various degrees of complexity, namely: (1) optimal control with a piecewise constant speed on time intervals of variable length, (2) optimal control with a piecewise constant speed on time intervals of constant length, (3) optimal control with a constant speed, and (4) model-based predictive control with a piecewise constant speed on time intervals of constant length. The considered control methods will be compared for several scenarios.

Influence of the structural changes In this thesis we will also discuss the influence of the structural changes on the throughput. In particular, we will consider structural changes such as increasing the number of feeding devices, changing their corresponding inserting point around the transport boxes, and increasing the velocity of the transport boxes.

Baggage handling

Regarding the baggage handling process in large and busy airports, we consider the most challenging part of the automation, namely the part of the baggage handling system where the bags are transported at high speeds by destination coded vehicles (DCVs) running on a network of tracks, see e.g. Figure 1.3. As illustrated in Figure 1.3, a DCV is a metal cart with a plastic tub on top, being propelled by linear induction motors similar to roller coasters.

Currently, the track networks on which the DCVs transport the baggage have a simple structure, the DCVs being routed through the system using routing schemes based on preferred routes. These routing schemes adapt to respond on the occurrence of predefined events as follows. Each junction has a logic controller and a lookup table storing preferred routes from that junction to all unloading stations. Hence, if the currently preferred route is blocked due to e.g. jams or buffer overflows, then the next to-be-preferred-route of the lookup table is chosen and the switch out of that junction is toggled accordingly. However, the load patterns of the system are highly variable, depending on, e.g., the season, time of the day, type of aircraft at each gate, or the number of passengers for each flight [17]. So, predefined routes are far from optimal. Therefore, in this thesis we will not consider predefined preferred routes, but instead we will develop and compare efficient control methods to determine the optimal routing in case of dynamic demand.

Problem statement One can state the route choice control problem as follows. Given a demand of bags (identified by their unique code) entering the DCV-based baggage handling system, and the network of tracks, the route of each DCV (from a given loading station to the corresponding unloading station) has to be computed subject to operational and safety constraints, such that all the bags to be handled arrive at their end points within given time windows.

Figure 1.3: DCVs running on a network of tracks. Photo courtesy of Vanderlande Industries.

Control approaches and control frameworks In order to efficiently determine the route choice of each DCV, we will first consider predictive and heuristic control approaches. These control methods will be implemented in a centralized, decentralized, and distributed manner. Furthermore, we will also propose a hierarchical control framework that consists of a 2-level control structure with local switch controllers at the lowest level and one higher-level supervisory network controller. In this control framework, switch controllers provide position instructions for each switch in the network. The collection of switch controllers is then supervised by a network controller that mainly takes care of the flow instructions for the switch controllers.

Computing the optimal route choice yields a nonlinear, nonconvex, mixed integer optimization problem. The computational efforts required to determine the optimal route choice are high, and therefore, solving this optimization problem may become intractable in practice. Consequently, we will also present an alternative approach for reducing the complexity of the computations by writing the nonlinear optimization problem as a mixed integer linear programming (MILP) problem. The advantage is that for MILP optimization problems solvers are available that allow us to efficiently compute the global optimal solution. The solution of the MILP problem can then be used directly or as an initial starting point for the original optimization problem. To assess the performance of the proposed control approaches and control frameworks, we will consider a benchmark case study, for which the methods will be compared over several scenarios.

1.4 Main contributions

The main contributions of this research with respect to postal automation and baggage handling are the following:

Postal automation

- We will propose an event-driven model for the continuous-time flat sorting system which has been designed such that the destination bins can move bidirectionally with variable speed.
- We will develop and compare efficient model-based control methods to compute the speed profile of the destination bins that maximizes the throughput of the flat sorting machine. In particular, we will consider variants of optimal control with gradually decreasing complexity and model predictive control.

Baggage handling

- We will propose an event-driven model for the continuous-time DCV-based baggage handling system that will be used for model-based control.
- We will develop and compare efficient model-based control methods to compute the optimal routing of DCVs transporting bags from a given origin to a given destination such that the performance of a DCV-based baggage handling system is maximized. In particular, we consider centralized, decentralized, and distributed model predictive control, and heuristic approaches. We will also propose a hierarchical control framework for determining the route choice control of a DCV-based baggage handling system.

1.5 Thesis outline

The objective of this thesis is to develop efficient control methods that can be used in order to increase the efficiency of the considered transportation systems (sorting machines for large mail items in post sorting centers, and baggage handling in airports). Figure 1.4 presents a graphical road map depicting the organization of this thesis.

According to this graphical road map, the persons interested in the postal applications only should read the thesis using the following order: Chapter 1, Section 2.1 and 2.2.1 of Chapter 2, Chapter 3, and Chapter 5. The persons interested in baggage handling only should read the thesis using the following order: Chapter 1, Chapter 2, Chapter 4, and Chapter 5.

The thesis is structured as follows. In Chapter 2 we briefly introduce the concepts of optimal control and centralized, decentralized, distributed, and hierarchical model predictive control that will be later on used in this thesis in order to optimally transport (sort or route) the to be handled items (flats or bags respectively). For these control methods we present the theoretical framework, the algorithms that can be used in order to solve the optimization problems, together with their advantages and issues.

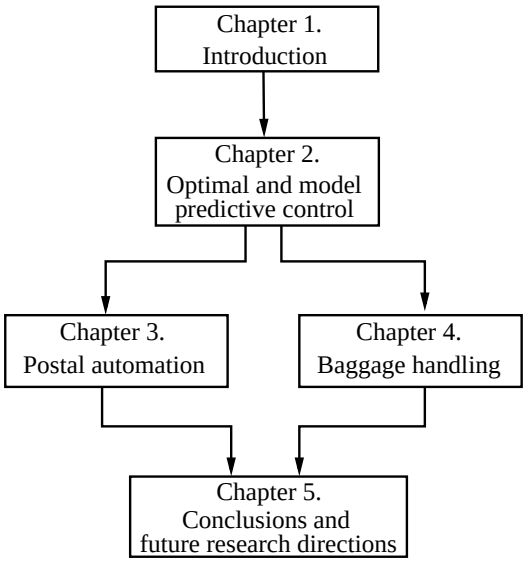


Figure 1.4: A road map of the thesis.

Next, in Chapter 3, we present the postal automation application. First, we describe the automated sorting process and the current issues of a class of flat sorting machines in general. Then, we propose a new design for a flat sorting system. Furthermore, we elaborate the simplifying assumptions made in order to obtain a fast simulation model, the continuous-time event-driven model to be used, the operational constraints, and the control objective that has to be achieved. Next, we propose several control approaches for determining the velocity of the system transporting the bins, and compare the proposed control methods based on simulations. Finally, we also discuss the influence of the structural changes on the throughput.

In Chapter 4 we present the baggage handling application. First, we describe the automated baggage handling process and the current control problems of a baggage handling system. Afterwards, we present the simplifying assumptions made in order to obtain a fast simulation model, the nonlinear event-driven model of the DCV-based baggage handling system, the operational constraints, and the desired control objective. Furthermore, we propose several control approaches for determining the optimal routing of bags through the baggage handling system and then we compared them (based on simulations) on benchmark case studies, over a set of scenarios.

Finally, in Chapter 5 we present the conclusions of this thesis and possible directions for future research.

Chapter 2

Optimal and model predictive control

In this chapter we briefly introduce the concepts of optimal control and model predictive control that will be later on used in this thesis. The chapter is structured as follows. In Section 2.1 we present the theoretical framework, the available numerical optimization algorithms, the advantages, and the issues of optimal control. As it will be noted in Section 2.1, the optimal control method becomes intractable in practice for any systems with large control horizon as the ones that we consider in this thesis. Therefore, in Section 2.2 we also introduce the concept of model predictive control where smaller optimization problems have to be solved. However, since centralized model predictive control may still require high computational efforts, in Section 2.2 we also describe the working principle of decentralized and distributed predictive control approaches. Finally, we will also introduce the concept of hierarchical control which will be then combined with the concept of model predictive control.

2.1 Optimal control

Several methods for solving dynamic optimization problems have been developed. In this section we present the general concept of optimal control, the algorithms that could be used to solve the resulting complex optimization problems, and also the advantages and the disadvantages of this approach.

2.1.1 Theoretical framework

Optimal control is a standard method for solving dynamic optimization problems, when those problems are expressed in continuous time. The optimal control problem consists of finding the time-varying control law $u(\cdot)$ for a given system such that an objective function J is optimized while satisfying the operational constraints imposed by the model, see, e.g., [37, 46, 53]. Hence, this is an open-loop approach (the control inputs of the system are computed using only the current state of the system and the model of the system). So, the

open-loop approach does not use any feedback to determine whether the desired goal has been achieved.

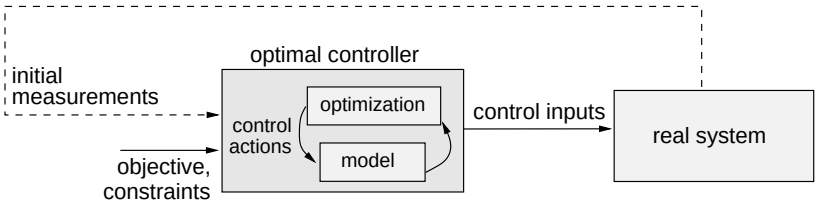


Figure 2.1: Optimal control — working principle. The control actions that the optimal controller obtains as result of solving the optimization problem over the entire simulation period, become control inputs for the real system (no feedback involved).

As illustrated in Figure 2.1, the formulation of an optimal control problem requires the following information:

- a model of the system to be controlled,
- an objective function to be optimized,
- boundary conditions and other operational constraints on the states, the inputs, and the outputs of the system, and consequently of its model (the inputs and the outputs of the system correspond to the control actions and the measurements of the optimal controller, respectively).

Then the standard formulation of an optimal control problem can be written as follows:

$$\begin{aligned}
&\min_{\mathbf{u}} J(\mathbf{x}(t_0),\mathbf{u}) \\
&\text{subject to} \\
&\qquad \Phi(\mathbf{x}(t_0),\mathbf{u}) = 0 \\
&\qquad \Psi(\mathbf{x}(t_0),\mathbf{u}) \leq 0
\end{aligned}$$

where

- $\mathbf{x}(t_0)$ is the state[1] of the system at time instant t_0 with t_0 the initial simulation time,
- $\mathbf{u}$ represents the continuous control actions for all the decision variables[1] over the simulation period $[t_0, t_0+\tau^{\text{sim}})$ with τ^{sim} the length of the simulation period,
- $\Phi(\mathbf{x}(t_0),\mathbf{u}) = 0$ is the system of equality constraints,
- $\Psi(\mathbf{x}(t_0),\mathbf{u}) \leq 0$ is the system of inequality constraints.

The system of equality and inequality constraints of the standard formulation above describes the continuous-time model of the real system and its operational constraints.

Note that for some problems the exact model of the systems can be written analytically, and then optimal control methods can give the global optimal solution only if the problem is

[1] Consider a traffic light controlled intersection with 4 arms and 4 traffic signals. Then the state of the system at a given time instant consists of the length of queues of the vehicles in front of each traffic light at that time instant. Then the decision variables can be for example the time instants when the color changes for each traffic light.

convex or if the solution can be computed analytically, see, e.g., [23, 29, 62]. However, for systems with models that cannot be written analytically, e.g., continuous-time event-driven systems as those we refer to in this thesis, one can use numerical optimization algorithms to compute the optimal solution, see Section 2.1.2. However, in this case the continuous adjustment of the control signal so as to optimize the objective function J is not possible using digital control. Hence, in practice one has to discretize the model and to compute a piecewise constant control tuple $\mathscr{U} = (\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(k^{\text{sim}}))$ where $k^{\text{sim}}\tau_{\text{s}}$ represents the length of the simulation period with τ_{s} the sampling time, $k^{\text{sim}}\tau_{\text{s}} = \tau^{\text{sim}}$, and where, for $k = 0, \ldots, k^{\text{sim}}$, $\mathbf{u}(k)$ is the vector of control inputs to be used for the time period $[t_k, t_{k+1})$ with t_k given by $t_k = t_0 + k\tau_{\text{s}}$.

Then the standard formulation of a discrete-time optimal control problem becomes:

$$\begin{aligned}
&\min_{\mathscr{U}} J^{\text{d}}(\mathbf{x}(t_0), \mathscr{U}) \\
&\text{subject to} \\
&\qquad \Phi^{\text{d}}(\mathbf{x}(t_0), \mathscr{U}) = 0 \\
&\qquad \Psi^{\text{d}}(\mathbf{x}(t_0), \mathscr{U}) \leq 0
\end{aligned}$$

where the system of equality and inequality constraints above describes the discrete-time model of the real system and its operational constraints. Examples of such formulation can be found in Chapter 3 and Chapter 4.

2.1.2 Numerical optimization algorithms

The solutions to most optimal control problems cannot be found by analytical means. As a result, it is necessary to employ numerical methods to solve optimal control problems. Over the years, many numerical procedures have been developed to solve optimal control problems as will be detailed next.

In order to solve nonlinear, nonsmooth optimization problems, one may use specialized search algorithms [31] such as *sequential quadratic programming* algorithms [28], or *pattern search* [4], *genetic algorithms* [67], and *simulated annealing* [18]. Furthermore, if the optimization problem is also a mixed integer problem, then one can solve it using the previous algorithms adapted to compute control inputs that are restricted to integer values, or other specialized *mixed-integer nonlinear programming* algorithms [35, 54], or *tabu search* [32].

Note that all these algorithms perform a *local* search starting from initial search points which are either fixed, given by the user, or randomly chosen by the optimization algorithms. As a consequence, they find *local* optimal solutions. Hence, these algorithms do not guarantee the global optimal solution. Therefore, for algorithms that start the search from fixed or random points given by the user one should use multiple initial points while for algorithms that start the search from random initial feasible solutions (randomly chosen by the optimization algorithm), one has to start the optimization several times, and hence, use *multi-run* optimization.

2.1.3 Advantages and issues

One advantage of optimal control is its ability to control systems with multiple inputs and multiple outputs, and also its explicit way of handling constraints on control actions, states,

and outputs. Another advantage is that optimal control methods can give the optimal solution when the exact model of the real system can be written analytically. However, in practice many of the systems to be controlled are highly nonlinear, and their exact and accurate model cannot be written analytically.

In theory this control method gives the global optimal solution, and consequently the best performance of the real system if the model expresses the real system accurately. Moreover, in practice determining an accurate model of a real system is not always possible. And since the optimal control method is open-loop, the model mismatch typically yields loss in the system's performance. Furthermore, in order to use the algorithms above one may have to first discretize the system. Then, depending on the dynamics of the system one may need a small sampling time, and consequently, one has to compute a large sequence of control signals since we determine the optimal control sequence for the entire simulation period. This will result in a large computation time when solving the optimization problem. Also, for systems where the optimal control method has to compute the solution of a nonlinear, nonconvex, nonsmooth, (mixed integer) optimization problem, this control method requires very large computational effort to determine the optimal solution. This occurs since the state-of-the-art numerical optimization algorithms designed to solve these complex problems can only determine local solutions (see, e.g., Section 2.1.2). In order to get closer to the global solution, one has to use multiple initial points. Therefore, for those systems, optimal control, usually, becomes intractable in practice. Another issue of optimal control is its robustness [53, chapter 9], since due to eventual disturbances, when applying the optimal control actions to the real system, the states of the system may not satisfy the imposed bounds.

2.2 Model predictive control

Since using optimal control yields high computational requirements to determine the optimal control inputs for event-driven systems as the ones we will consider in the next chapters, in this section we introduce the concept of model predictive control (MPC).

2.2.1 Centralized MPC

MPC is an on-line control design method for discrete time models that uses the receding horizon principle, see, e.g., [11, 57, 65]. Therefore, this control method is also referred to as receding horizon control or moving horizon control. Since its development in 1980 [14, 68], MPC has become the preferred control strategy for a large number of industrial processes. Currently, MPC is viewed as one of the most promissing control methods that can deal with nonlinear systems that are subject to operational constraints.

MPC is a control strategy that is typically used in a discrete-time context. Next we present the working principle of basic MPC. As sketched in Figure 2.2, at some time instant, the MPC controller measures or estimates the current state of the real system. Let this time instant be denoted by $t_k = t_0 + k\tau_s$ with t_0 the time instant when we start the simulation, τ_s the sampling time, and $k \geq 0$ an integer. Then, given a prediction model of the real system, the MPC controller computes control actions by solving an optimization problem subject to the prediction model's dynamics and its operational constraints as follows. Given

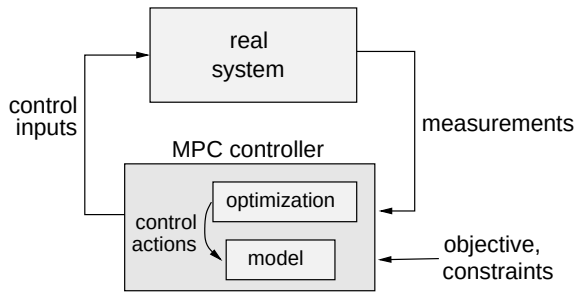


Figure 2.2: Basic MPC — working principle.

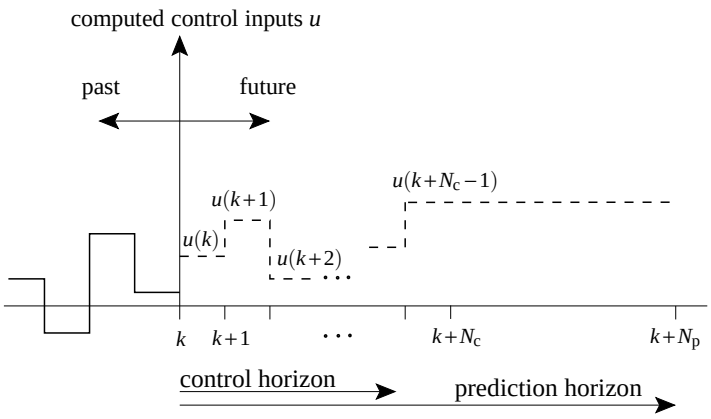


Figure 2.3: Conventional MPC when we deal with one decision variable during a sample period. At time step k the future control sequence $u(k),\ldots,u(k+N_c-1)$ is optimized such that the objective function J is minimized subject to the dynamics of the system and their operational constraints.

a prediction horizon N_p and a control horizon N_c with $N_c \leq N_p$, at time step k (corresponding to time instant t_k), the future control vectors $\mathbf{u}(k),\ldots,\mathbf{u}(k+N_c-1)$ — where $\mathbf{u}(k)$ the vector of decision variables during the time period $[t_k,t_{k+1})$ — are computed (see Figure 2.3) by solving a discrete-time optimization problem over a given period $[t_k,t_k+N_p\tau_s)$, so that the cost criterion J is optimized subject to the operational constraints. The input signal is typically assumed to become constant beyond the control horizon, i.e.,

$$\mathbf{u}(k+j) = \mathbf{u}(k+N_c-1) \text{ for } j = N_c,\ldots,N_p-1. \tag{2.1}$$

After computing the optimal control vectors, only the first control vector (corresponding to the time period $[t_k,t_{k+1})$) is implemented on the real system, and subsequently the horizon is shifted. Next, the new state of the system is measured or estimated, and a new optimization problem at step $k+1$ is solved using this new information. In this way, a feedback mechanism is introduced. Recurrently, we apply this procedure until $k = k^{\text{sim}}$.

The standard formulation of an MPC optimization problem can then be written as fol-

lows:

$$\begin{aligned}
&\min_{\mathscr{U}(k)} J_{k,N_\mathrm{c},N_\mathrm{p}}(\mathbf{x}(t_k),\mathscr{U}(k))\\
&\text{subject to}\\
&\qquad \Phi(\mathbf{x}(t_k),\mathscr{U}(k)) = 0\\
&\qquad \Psi(\mathbf{x}(t_k),\mathscr{U}(k)) \leq 0
\end{aligned}$$

where

- $\mathbf{x}(t_k)$ is the vector of state variables at time instant t_k,
- $\mathscr{U}(k)$ is the N_c-tuple that consists of all the decision variables to be applied over the prediction horizon and is defined as follows $\mathscr{U}(k) = (\mathbf{u}(k),\ldots,\mathbf{u}(k+N_\mathrm{c}-1))$,
- $\Phi(\mathbf{x}(t_k),\mathscr{U}(k)) = 0$ is the system of equality[2] constraints,
- $\Psi(\mathbf{x}(t_k),\mathscr{U}(k)) \leq 0$ is the system of inequality[2] constraints.

The main advantage of MPC over the optimal control method is that we now solve smaller optimization problems. However, this comes at the cost of loosing performance when N_c and N_p are small relative to k^{sim}. Note that in order to solve the MPC optimization problems one can use the numerical optimization algorithms presented in Section 2.1.2.

Furthermore, one may notice that depending on the control and prediction horizons, computing the solution of the optimization problem over the entire simulation period may still require high computational effort. In order to reduce the computational complexity, one can use variants of MPC that involve:

larger horizon shifting: Instead of applying to the real system only one control sample out of the computed control sequence, one can apply more samples and shift the horizon accordingly. This means that if at step k we have computed the control tuple $\mathscr{U}(k) = (\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+N_\mathrm{c}-1))$ with $\mathbf{u}(k+j) = \mathbf{u}(k+N_\mathrm{c}-1)$ for $j = N_\mathrm{c},\ldots,N_\mathrm{p}-1$, then one can apply $m \leq N_\mathrm{p}$ control samples $\mathbf{u}(k),\ldots,\mathbf{u}(k+m-1)$ to the real system. Accordingly, we next compute the future control tuple at step $k+m$. In this way the total computation time required to compute the control over the period $[t_0, t_{k^{\mathrm{sim}}})$ is reduced by $100\left(\frac{m-1}{m}\right)$%.

blocking: Instead of considering the control horizon constraint only expressed by (2.1), one can force the input to remain constant during some predefined intervals. So, one can define n^{block} intervals of length $\delta_1^{\mathrm{block}},\ldots,\delta_{n^{\mathrm{block}}}^{\mathrm{block}}$ so that $\sum_{i=1}^{n^{\mathrm{block}}} \delta_i^{\mathrm{block}} = \tau_\mathrm{s} N_\mathrm{p}$ with $\delta_i^{\mathrm{block}}$ an integer multiple of τ_s for $i = 1,2,\ldots,n^{\mathrm{block}}$, see, e.g., Figure 2.4. Then we compute the future control inputs $\mathbf{v}(k),\mathbf{v}(k+1),\ldots,\mathbf{v}(k+n^{\mathrm{block}}-1)$ that optimize the objective function J subject to the dynamics of the system and their operational constraints, where $\mathbf{v}(k+i)$ for $i = 1,2,\ldots,n^{\mathrm{block}}$ is the control input corresponding to the time interval $[t_0+\sum_{l=1}^{i-1}\delta_l^{\mathrm{block}}, t_0+\sum_{l=1}^{i}\delta_l^{\mathrm{block}})$ with $\sum_{i=1}^{0}\delta_i^{\mathrm{block}} = 0$ by definition.

[2]The system of equality and inequality constraints of the MPC standard formulation, describes the prediction model of the real system over the given prediction horizon and its operational constraints.

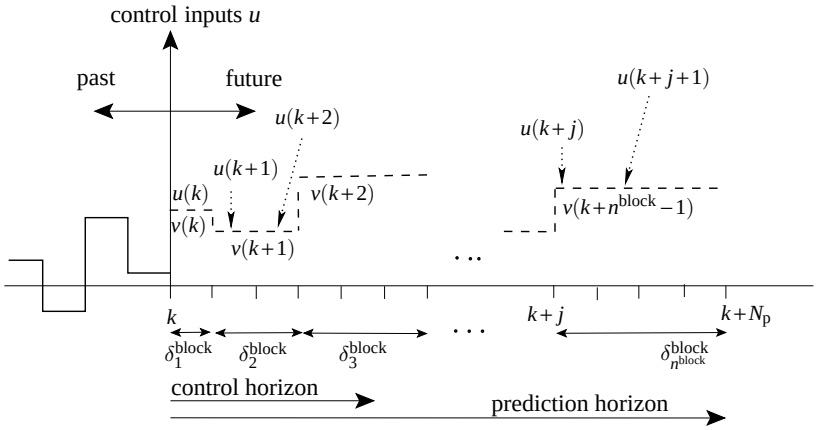


Figure 2.4: MPC with blocking (we consider one decision variable only during a sample period). For this example, at time step k, we compute the optimal future control variables $v(k),\ldots,v(k+n^{\text{block}}-1)$). The control variables $u(k),\ldots,u(k+N_{\text{p}}-1)$) are then computed according to (2.2).

Then, as illustrated in Figure 2.4, the control inputs $\mathbf{u}(k),\ldots,\mathbf{u}(k+N_{\text{p}}-1)$ can be derived as follows:

$$\mathbf{u}(k+j)=\mathbf{v}(i) \text{ for } i=1,2,\ldots,n^{\text{block}} \text{ and for all } j\in\mathbb{N} \text{ satisfying:}$$

$$\frac{\sum_{l=1}^{i-1}\delta_l^{\text{block}}}{\tau_{\text{s}}}\leq j<\frac{\sum_{l=1}^{i}\delta_l^{\text{block}}}{\tau_{\text{s}}}. \tag{2.2}$$

Another advantage of MPC is that it can handle structural changes — such as sensor and actuator failure changes in system parameters and system structure — by regularly updating or adapting the prediction model in combination with its feedback mechanism.

2.2.2 Decentralized MPC

When dealing with large-scale systems, centralized MPC is no longer tractable. Therefore, for such applications one can divide the system into subsystems, which are then independently controlled by local controllers [43, 94]. Then we deal with a decentralized control architecture, see, e.g., Figure 2.5, where given the local prediction models, each local controller solves a local optimization problem based on local information over the state of the real system. This results in sequences of local control actions that can be applied to the real system. For different applications of decentralized MPC we refer to [1, 19, 69].

So, the advantage of decentralized MPC over the centralized approach is that we now independently solve simpler and smaller optimization problems resulting in lower computational requirements and faster control. However, this advantage will be typically at the price of decreased overall performance.

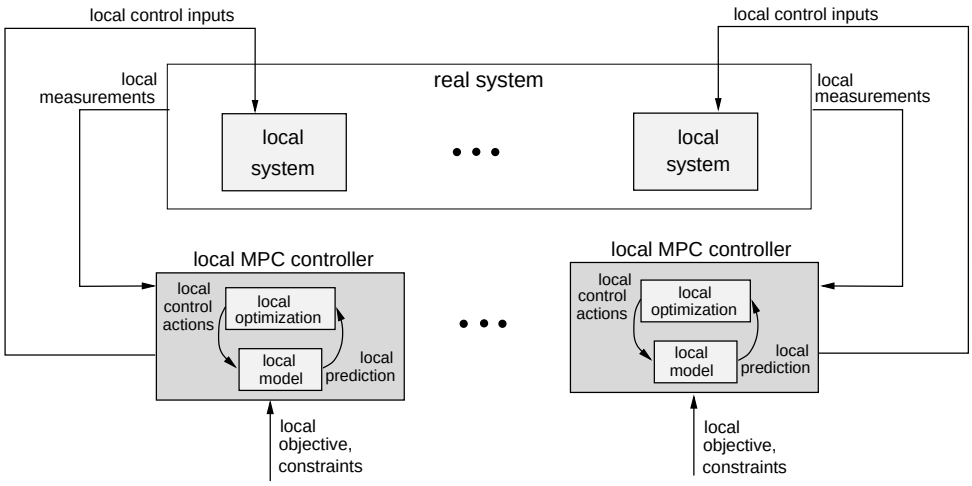


Figure 2.5: Decentralized MPC —working principle.

2.2.3 Distributed MPC

In order to increase the performance of decentralized MPC, the concept of distributed MPC has been introduced. Distributed MPC is an extension of decentralized MPC, where the local MPC controllers also exchange information regarding their future control actions while solving local optimization problems, see e.g., Figure 2.6. Typically, the objective of this communication is to achieve some degree of coordination among the local controllers without solving a centralized MPC problem. This topic has been addressed in [12, 13, 20, 60, 66, 93] where, e.g., serial versus parallel and synchronous versus asynchronous coordination schemes are tackled. In serial computation schemes, only one local controller at a time performs computations, while in parallel schemes, multiple local controllers perform computations simultaneously. When the computations are performed in parallel, the local controllers have to wait or not for one another when it comes to sending and receiving information and determining which actions to take; also they can send and receive information and determine their actions at any time or at specific time instants. The asynchronous coordination schemes have a big advantage over the synchronous coordination schemes, namely that the local controllers do not have to wait for other agents to perform their computations — they just have to include the newly received information from neighboring local controllers at any time while solving their optimization problems.

In this work we will not focus on developing new coordination schemes for distributed MPC, but we will analyze the trade-off between performance and computation time needed for solving nonlinear, nonconvex, mixed integer optimization problems with multiple local minima, when applying efficient centralized, decentralized, and distributed MPC approaches.

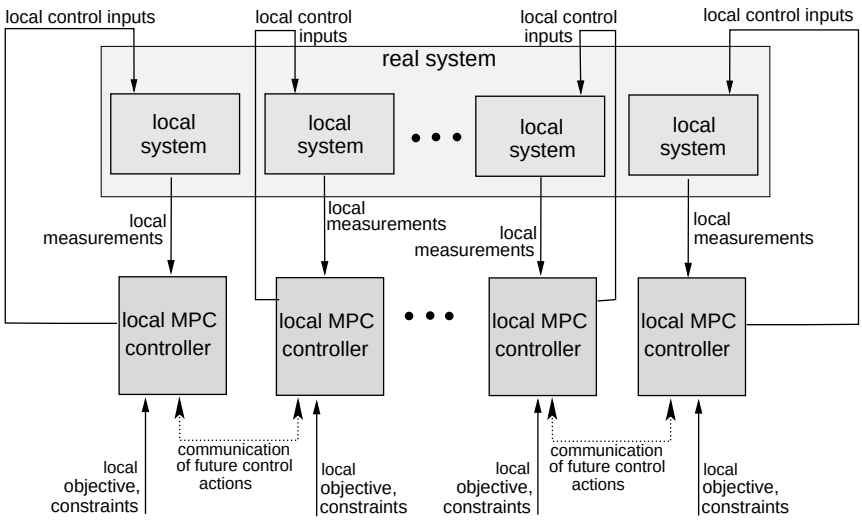


Figure 2.6: Distributed MPC —working principle.

2.2.4 Hierarchical MPC

In a hierarchical control set-up, see, e.g., [26, 72], the control tasks are distributed over time and space. Such a set-up consists of several levels of control, where controllers of supervisory and strategic functionality reside at higher levels, while at lower levels the local controllers must guarantee specific operational objectives. At any level, the local controllers must communicate their outcomes and requirements to the lower levels (sometimes these controllers even negotiate their outcomes and requirements with the lower and higher levels).

Using MPC in a hierarchical framework involves multiple control levels with authority relationships between the local MPC controllers on the different levels as illustrated in Figure 2.7.

This framework can be characterized as follows:

- It consists of multiple control levels with authority relationships between the local controllers on the different levels (local controllers at higher levels — also called supervisory controllers — have authority over the controllers at lower levels, whereas the local controllers within a control level have equal authority relationships).

- In general, the local controllers on different levels have different objectives.

- At higher levels typically less detailed models are considered, whereas at lower levels more detailed models will be used.

- The different levels of control deal with different time scales. Typically the lower levels in this hierarchy update their actions with a faster frequency than the higher levels.

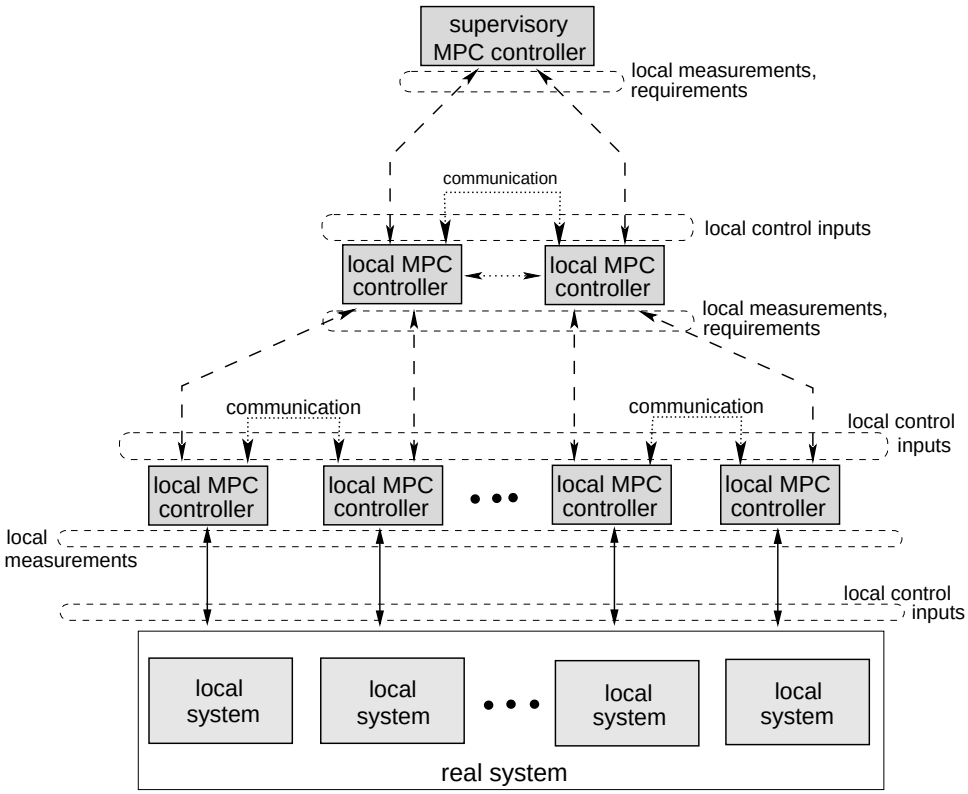


Figure 2.7: Hierarchical MPC with 3 levels of control. The local controllers on each level communicate their outcomes and requirements to the lower level, and negotiate their requirements with the higher levels.

The use of MPC in a hierarchical framework [42, 70, 71] has already proven its usefulness in controlling transportation systems, see e.g., [6, 21, 24].

2.3 Summary

In this chapter we have introduced the concepts of optimal control and model predictive control which will be later on used in this thesis for solving nonlinear, nonconvex, mixed integer optimization problems with multiple local minima. For event-driven systems (as the ones that we will deal with in the next chapters) where we have to determine the optimal solution of nonlinear and nonconvex optimization problems, optimal control becomes intractable in practice for large horizon due to the high computational effort required. For these systems model predictive control (MPC) offers a reduction in the overall computation time by solving smaller optimization problems (over a relatively small prediction horizon only, instread of computing the optimal solution over the entire simulation period). However, centralized MPC may still become intractable in practice for large-scale systems. Therefore, one

can decompose the large-scale systems into subsystems, and accordingly solve local MPC optimization problems. The advantage of decentralized MPC is a lower computation time since we now independently solve local optimization problems that are smaller and simpler. However, this comes at the cost of suboptimality. But, by including communication and coordination between local controllers, one obtains distributed MPC which can improve the efficiency of the system. Finally, we have also presented a hierarchical control framework consisting of multiple control levels with authority relationships between the local controllers on the different levels. This control framework will then later on be used in combination with the MPC concept.

Chapter 3

Postal automation

In this chapter we consider state-of-the-art flat sorting machines. The chapter is structured as follows. In Section 3.1 we describe the automated sorting process and the current issues of a flat sorting machine. Afterwards, in Section 3.2, we propose a new design for the flat sorting machine. The simplifying assumptions and the continuous-time event-driven model to be used are presented in Section 3.3. Furthermore, in Section 3.4 we detail the operational constraints together with the control objective. In Section 3.5, we propose several control approaches for determining the velocity of the system transporting the bins. The analysis of the simulation results and the comparison of the proposed control methods are elaborated in Section 3.6. In Section 3.6 we also discuss the influence of the structural changes on the throughput. Finally, in Section 3.7, we draw the conclusions of this chapter and we present possible directions for future research.

Parts of this chapter have been published in [75], [79], and [84].

3.1 State-of-the-art solutions

In this section we briefly describe the process performed by a state-of-the-art flat sorting machine and its current issues.

3.1.1 Process description

The procedure performed by a flat sorting machine consists of two processes: preparing the flats and sorting them. During the preparation phase, the stamp used for postage is voided. Next, the address and the postal code are located and the necessary information is extracted and printed on the flat in the form of a bar code. Conveyor systems transport the flats during the preparation phase with a constant speed. This ensures a transport delay line of several seconds allowing the system to collect sorting information on-line before the mail item reaches the code printing phase. The performance of the reading device, the length of these conveyor belts, and their speed determine the maximal amount of time available to prepare the mail for sorting. If the delivery information is not machine-readable, an image of the flat will be transmitted automatically to the video coding system. An operator views the address image on a monitor, reads the delivery information, and enters it via a keyboard. If

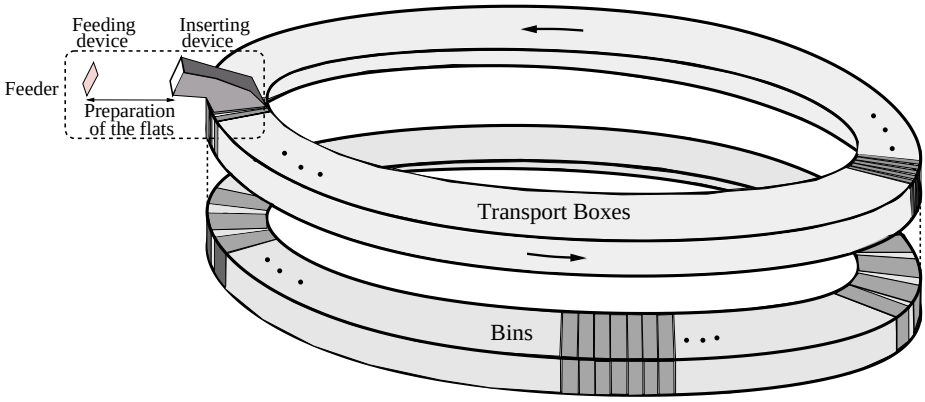


Figure 3.1: Sorting part of a flat sorting machine. A full bin is immediately replaced with an empty one.

the delivery information is not obtained during the preparation phase, then an identification code is assigned to the item and the flat is inserted into a transport box. This flat will be then dropped into a special bin where the non machine-readable are collected. However, if the delivery information is acquired when the flat is already in the transport box, then this mail item will be reassigned to the correct destination bin. When the mail item leaves the preparation phase, it is inserted into a transport box of the sorting process by the inserting device, as sketched in Figure 3.1. The transport boxes are sustained at the top part of a flat sorting machine and move counterclockwise (top view) with a constant speed. At the bottom part of this machine destination bins are aligned. Note that this bottom part does not move in any direction (this part is static). Then the transport box carries the flat and deposits it by dropping it into a destination bin according to the destination or postal code of the flat. This is how currently most of the flat sorting machines are working.

Note that for the sake of simplicity of explanation, in the remainder of the paper we will also use the terms *box* and *bin* when referring to a *transport box* and a *destination bin* of a flat sorting machine.

3.1.2 Current issues

The low-level control problems of this system consist of determining the feeding rate of the sorting machine [56], positioning of the transport box when inserting the flat, and synchronizing transport boxes and bins when dropping a flat in its corresponding destination bin. At a higher level of control important problems are how to allocate the destinations to the bins and how to sort the mail items in delivery sequence order.

Other issues related to the state-of-the-art mail sorting systems in general, and applicable also to the considered flat sorting machine are: locating the destination address and extracting the necessary information, and also designing optical character recognition machines. These topics have been treated to a very large extent in, e.g., [40], [52], [97]. However, to the author's best knowledge there is no public work analyzing how the efficiency of this system can be increased.

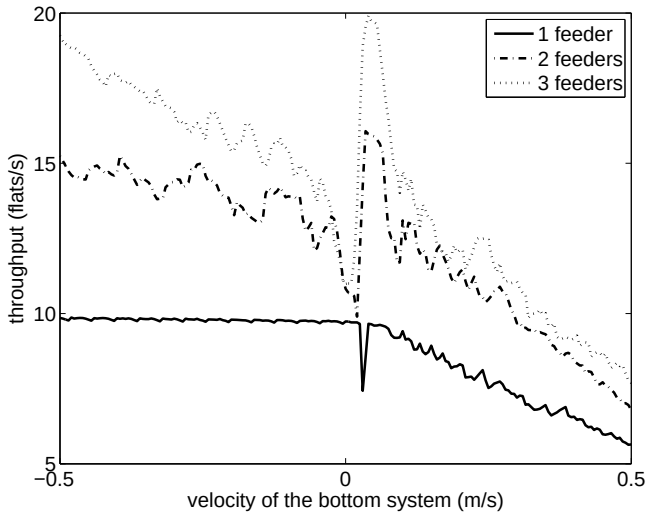


Figure 3.2: Throughput versus v^{bottom} when $v^{\text{top}} = 1\,\text{m/s}$ and v^{bottom} varies between, e.g., $-0.5\,\text{m/s}$ and $0.5\,\text{m/s}$ (v^{bottom} is negative when the bottom system has the opposite direction of movement with respect to the top system). The flat sorting machine has 1, 2, and 3 feeders respectively.

3.2 New design

In this section we investigate approaches to increase the throughput of the flat sorting machine. This can be achieved first by making design changes such as augmenting the system with additional feeders and also by moving the bin system to the left or to the right with a given speed.

Motivation

The sorting system sketched in Figure 3.1 can be augmented by adding feeders. However, by increasing the number of feeders only, which can increase the throughput of the machine, one does not necessarily obtain the maximal possible throughput. As example we have illustrated in Figure 3.2 the throughput versus the velocity of the bin system for a typical scenario of 10000 flats. These results have been derived using the event-driven model that will be presented in Section 3.3 and which has been implemented in Matlab. However, since the purpose of these plots is only to motivate the need for more indepth analysis of means to increase the throughput of a flats sorting machine, we will not detail here how we derived these results (the details are presented in Section 3.3).

The general trend of the throughput of a flat sorting machine with 1, 2, or 3 feeders is to decrease when the bottom part of the machine moves in the same direction as the top part and when using a constant speed in the range $0\,\text{m/s}$ to $0.5\,\text{m/s}$. Moreover, the throughput of such a machine can decrease to $0\,\text{flats/s}$ when the bottom part of the machine moves in the same direction as the top part and with the same speed ($1\,\text{m/s}$). This happens since in this case we can drop only the flats that are inserted in boxes positioned on top of the destination

bin with the same identification code as the flat. Furtherore, Figure 3.2 shows that when the sorting machine has one feeder only, and the bottom part of the machine moves with a constant speed, but in opposite direction than the top part, the throughput is about 10 flats/s. This happens since the inserting rate is 10 flats/s (in the considered scenario the width of a box is 0.1 m and the velocity of the top part is 1 m/s, and as a consequence 10 empty[1] boxes pass under the inserting device within 1 second). The very small differences in throughput in this case are only influenced by the time instant when the last inserted flat is dropped. So, the larger the stream of flats to be sorted, the larger the total sorting time, and consequently, the smaller the differences in throughput for this case. The results illustrated in Figure 3.2 also indicate that augmenting the flat sorting machine with more feeders increases the throughput, and that typically increasing the relative speed between the top and the bottom system also increases the throughput. However, the evolution of the throughput versus the velocity of the bottom part of the sorting machine is nonlinear and nonsmooth. However, the peaks that appear cannot be predicted, but are dependent on the stream of codes and on the velocity used for the bottom system.

According to the results illustrated in Figure 3.2 (see e.g., the throughput of a flat sorting machine with 2 or 3 feeders) one concludes that the throughput obtained with a static bin system is not always optimal.

Therefore, in the new set-up, the bottom system transporting the bins is also able to move clockwise or counterclockwise (top view) with varying speed. Moving the bottom part of the flat sorting machine, but with a constant speed, is currently already being implemented and operational.

Description

In order to increase the throughput of the flat sorting machine, we propose the new set-up illustrated in Figure 3.3. The preparation of the flats is identical to the one described in Section 3.1.1. But now we want to simplify the previous sketch. Therefore, instead of the feeding device and the preparation phase, we now consider a buffer of flats with known identification codes. Note that in this work we consider that the delivery information is always acquired on-line. The top system transporting the boxes moves as usual, with a constant speed. The bottom system transporting the bins can now move clockwise or counterclockwise with varying speed. The reason for this is to increase the number of empty transport boxes and, hence, increase their availability.

New control problems

With this new set up, a new control problem arises: how to adjust the speed of the bottom system, so that the throughput is maximized. In order to ensure the optimal movements we will implement advanced model-based control methods, namely optimal control and model predictive control (MPC). A detailed presentation of these methods can be found in Section 3.5.

[1] The boxes that pass under the inserting device of a flat sorting machine with 1 feeder are always empty when the bottom part moves in opposite direction than the top part since the flat in a box will always be dropped before that box passes again under the inserting device.

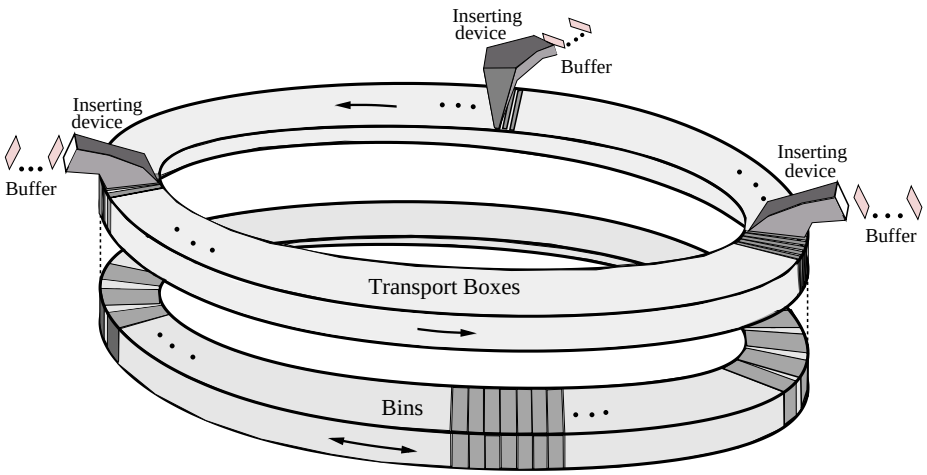


Figure 3.3: New set-up for the flat sorting machine: more feeders, moving destination bins

3.3 Event-based model

Later on we will use the model of a flat sorting system for model-based control. Therefore, in order to have a trade-off between a detailed model that requires large computation time and a fast simulation, in this section we present the simplifying assumptions and the continuous-time event-driven model to be used to determine the optimal speed of the destination bins.

3.3.1 Assumptions

Consider the simplified process depicted in Figure 3.3 of a flat sorting system with F feeders. Accordingly, we consider F FIFO (First In First Out) buffers of flats. Note that the sequences and the streams defined throughout this chapter will be represented by (column) vectors.

Let N^{boxes} and N^{bins} be respectively the number of boxes and the number of destination bins of the sorting machine. The width of a box will be denoted by w^{box} and the width of a bin will be denoted by w^{bin}.

To model the flat sorting system we make the following assumptions:

$\mathbf{A}_1$: The width of the gaps between the boxes is assumed to be negligible. We assume the following relation satisfied: $w^{\text{box}}N^{\text{boxes}} = w^{\text{bin}}N^{\text{bins}} = l^{\text{total}}$ with l^{total} the total length of the sorting part of the flat sorting machine.

$\mathbf{A}_2$: The top system moves with a constant speed v^{top}.

$\mathbf{A}_3$: The speed of the bottom system is piecewise constant.

$\mathbf{A}_4$: The flat sorting machine has F inserting devices that are positioned equidistantly. These inserting devices correspond to the F feeders and are denoted by $\mathrm{I}_1, \ldots, \mathrm{I}_F$.

$\mathbf{A}_5$: Each inserting device has a finite buffer of flats, with codes that are known in advance.

$\mathbf{A}_6$: When using a sorting machine with F feeders, the stream of codes $\mathbf{s} = [s_1\ s_2 \dots\ s_{N^{\text{flats}}}]^\top$ — where N^{flats} is the number of flats to be sorted during a sorting round —, is split into F new streams $\mathbf{s}_1 = [s_1\ s_2\ \dots\ s_f]^\top$, $\mathbf{s}_2 = [s_{f+1}\ \dots\ s_{2f}]^\top, \dots,$ $\mathbf{s}_F = [s_{(F-1)f+1}\ \dots\ s_{N^{\text{flats}}}]^\top$ with $f = \lfloor \frac{N^{\text{flats}}}{F} \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

$\mathbf{A}_7$: The correct dropping (and consequently the correct stacking) of a flat into a bin is controlled by a low-level controller. In the model that we determine, a flat can be dropped when the box carrying it is positioned on top of its destination bin, see e.g., Figure 3.4. Moreover, the dropping of a flat into a bin is assumed to be performed in a negligible time span.

Figure 3.4: Positioning when the box transports the flat to be dropped in the bin below and the dropping is still allowed.

$\mathbf{A}_8$: A full bin is replaced with a new one in a negligible time span.

Next we will discuss each of the assumptions above, stating why the assumption is required and whether or not it is (very) restrictive:

$\mathbf{A}_1$: This assumption has been made without loss of generality. Its purpose is to simplify the explanation of the event-based model.

$\mathbf{A}_2$: This assumption corresponds to state-of-the-art flat sorting machines.

$\mathbf{A}_3$: Recall from Chapter 2 that the continuous adjustment of the velocity of the bottom system so as to maximize the throughput J of a flat sorting machine is not possible using digital control. Hence, the assumption that v^{bottom} is piecewise constant is necessary. Note that $\mathbf{A}_3$ is not a very restrictive assumption since one can always approximate an arbitrary speed profile arbitrarily well by a piecewise constant speed profile.

$\mathbf{A}_4$: The assumption that the F inserting devices are positioned equidistantly is not restrictive, in the sense that other positions for the inserting devices are also allowed. However, this positioning will influence the throughput of the sorting system. In practice flat sorting machines with $F = 2$ or $F = 4$ already exist and are commercially available.

$\mathbf{A}_5$: This assumption corresponds to state-of-the-art flat sorting machines due to the following reason. During the preparation phase, after extracting from each flat its destination address and postal code, an identification code will be assigned to it. Hence, at any time instant there will be a buffer of flats transported by conveyor systems, the codes of which are known.

$\mathbf{A}_6$: This assumption is necessary in order to later on understand some of the simulation results. The assumption is not very restrictive in the sense that other ways of splitting

the initial stream of codes can also be allowed, but then the optimal speed profile would be different.

A_7: Taking into account the purpose of this work (i.e. to develop and compare control approaches for increasing the throughput of the sorting machine), the correct dropping of a flat into a bin is considered to be performed by a low-level controller already present in the system.

A_8: In practice, a full bin cannot be replaced with a new one in a negligible time span. But, one can design an intermediary pocket on top of each bin which can store a limited number of items when that specific bin is full. This yields then a small delay in dropping, and then the full bin can be replaced with an empty one. Note that automated bin replacement has already been developed and is currently operational.

3.3.2 Model

There are three types of events that can occur:

- inserting a new flat into the sorting section of the system,
- dropping the flats that meet the corresponding bin,
- updating the speed of the bottom system.

We model the flat sorting system as an event-driven model consisting of a continuous part, viz. the movement of the transport boxes and bins, and of the discrete events listed above. The following situation has been assumed: given a velocity sequence $\mathbf{v}^{\text{bottom}} = [v_0^{\text{bottom}}\ v_1^{\text{bottom}} \dots v_N^{\text{bottom}}]^\top$ and a sequence of time interval lengths $\boldsymbol{\tau} = [\tau_0\ \tau_1 \dots \tau_N]^\top$, on each time interval $[t_k, t_{k+1})$, $k = 0, 1, \dots, N$, with $t_{k+1} = t_k + \tau_k$ and t_0 the time instant when we start sorting, the velocity of the bottom system equals v_k^{bottom} as illustrated in Figure 3.5.

The model of the flat sorting system is captured by **Algorithm 1** where $\tau^{\text{sort,max}} \geq 0$ is the maximum time period that we allow for sorting. Moreover, according to the model, for each flat i, for $i = 1, 2, \dots, N^{\text{flats}}$ that has to be sorted, the time instant when the flat i is inserted into a box (t_i^{insert}) and the time instant when the flat i is dropped (t_i^{drop}) are computed. Consequently, the model of the flat sorting machine is denoted by $\mathbf{t} = \mathscr{M}(\mathbf{x}(t_0), \mathbf{v}^{\text{bottom}}, \boldsymbol{\tau})$, where $\mathbf{t} = [t_1^{\text{insert}} \dots t_{N^{\text{flats}}}^{\text{insert}}\ t_1^{\text{drop}}\ \dots\ t_{N^{\text{flats}}}^{\text{drop}}]^\top$ and $\mathbf{x}(t_0)$ is the initial state of the flat sorting system. Note that if during the time period $[t_0, t_0 + \tau^{\text{sort,max}})$ not all the flats have been

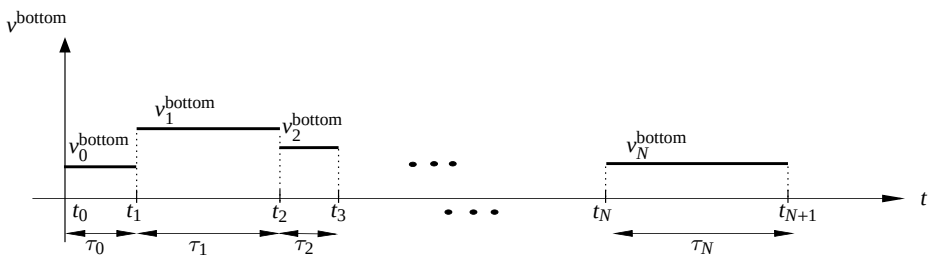


Figure 3.5: Speed evolution of the bin system.

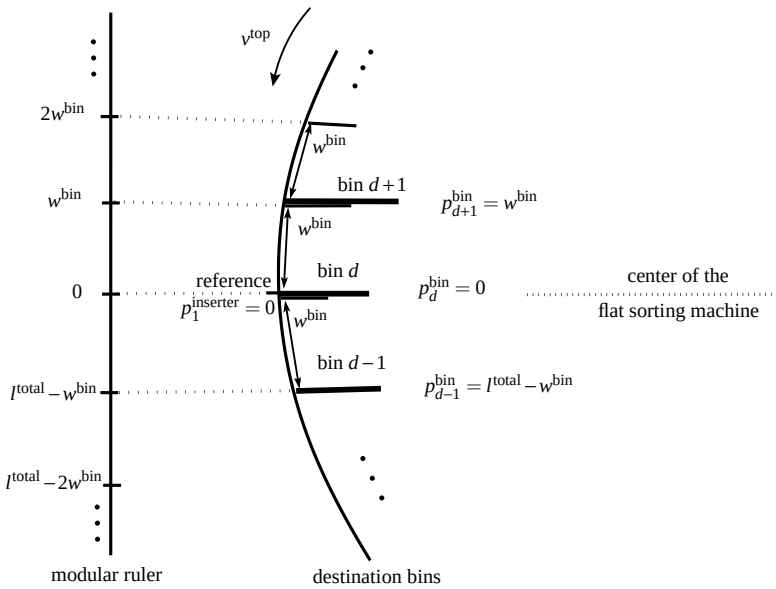


Figure 3.6: Position of bins relative to the position of the first inserting device.

sorted, then **t** consists of the time instants when we insert and drop each of the flats up to the time instant $t_0 + \tau^{\text{sort,max}}$.

For the sake of simplicity and without loss of generality, assume that the destination bins have assigned the following identification codes: $1, 2, \ldots, N^{\text{bins}}$. Note that from now on we refer to the bin with identification code d with $d \in \{1, \ldots, N^{\text{bins}}\}$ as bin d. Then we assign the identification codes as follows: the bin positioned at the left-hand side of bin d with $d \geq 2$ is bin $d+1$ and the bin positioned at the right-hand side of bin d is bin $d-1$ — the bin positioned at the right-hand side of bin 1 is bin N^{bins}. Then in order to identify the position of a bin or of a box we only refer to its right-hand side relative to the right-hand side of the first inserting device[2]. The position of the first inserting device is denoted by p_1^{inserter}. Let us set $p_1^{\text{inserter}} = 0$ as reference. As an example, Figure 3.6 illustrates the position of bin d with $d \in \{1, \ldots, N^{\text{bins}}\}$ and the positions of the bins in front and after bin d. Then we denote the position of the bin with identification code d by p_d^{bin}. Similarly, we denote the position of box m with $m \in \{1, \ldots, N^{\text{boxes}}\}$ by p_m^{box}. Note that the positions of boxes and bins are determined using modular arithmetic since these positions are expressed as variables that are larger than or equal to 0 and smaller than l^{total} with l^{total} the total length of the sorting part of the flat sorting machine ($l^{\text{total}} = w^{\text{box}} N^{\text{boxes}} = w^{\text{bin}} N^{\text{bins}}$ according to assumption $\mathbf{A}_1$).

The **state** of the sorting system consists of the positions of all the boxes and bins, the state of the box (loaded or empty), the time instant when we had the last dropping event for each box, the number of flats dropped till now for each destination, and the streams of codes $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_F$ that still have to be inserted into the machine. Note that the positions of the transport boxes and bins are determined relative to fixed points (e.g. position of the first

[2]The inserting devices have fixed positions with respect to the frame of the machine.

inserting device). Furthermore, the **input** of the system consists of the streams of codes to be sorted $\mathbf{s}_1,\ldots,\mathbf{s}_F$, the piecewise constant velocity profile for the destination bins $\mathbf{v}^{\text{bottom}}$ with $\mathbf{v}^{\text{bottom}} = [v_0^{\text{bottom}}\, v_1^{\text{bottom}} \ldots v_N^{\text{bottom}}]^\top$, and the time interval lengths $\boldsymbol{\tau}$ on which the velocity of the destination bins is constant with $\boldsymbol{\tau} = [\tau_0\, \tau_1 \ldots \tau_N]^\top$. The **control variable** is the velocity of the bottom part of the flat sorting machine and is part of the input. Finally, the **output t** consists of the time instants when we insert and drop each of the flats to be handled.

The variable τ_j^{insert} with $j \in \{1,\ldots,F\}$ of **Algorithm 1** presented below is the time that will pass until the first empty transport box will be positioned under the inserting device I_j. The variable τ_m^{drop} with $m \in \{1,\ldots,N^{\text{boxes}}\}$ is the time that will pass until the next mail item will be dropped from the box m into its destination bin.

Algorithm 1. Model of the flat sorting system

Input: streams of codes to be sorted $\mathbf{s}_1,\ldots,\mathbf{s}_F$, piecewise constant velocity profile for the destination bins $\mathbf{v}^{\text{bottom}}$, time interval lengths $\boldsymbol{\tau}$, and the initial state of the system

$k \leftarrow 0$
$t^{\text{crt}} \leftarrow t_0$
while $t^{\text{crt}} < t_0 + \tau^{\text{sort,max}}$ **do**
  for $j = 1$ to F **do**
    $\tau_j^{\text{insert}} \leftarrow$ time that will pass until the next flat inserting event at I_j
  end for
  for $m = 1$ to N^{boxes} **do**
    $\tau_m^{\text{drop}} \leftarrow$ time that will pass until the next dropping event for the box m
  end for
  $\tau^{\min} \leftarrow \min\Big(\min\limits_{j=1,\ldots,F} \tau_j^{\text{insert}}, \min\limits_{m=1,\ldots,N^{\text{boxes}}} \tau_m^{\text{drop}}, \tau_k\Big)$
  $t^{\text{crt}} \leftarrow t^{\text{crt}} + \tau^{\min}$
  for $m = 1$ to N^{boxes} **do**
    $p_m^{\text{box}} \leftarrow \left(p_m^{\text{box}} - v^{\text{top}}\tau^{\min}\right) \bmod l^{\text{total}}$
  end for{update the position of the box system}
  for $i = 1$ to N^{bins} **do**
    $p_i^{\text{bin}} \leftarrow \left(p_i^{\text{bin}} - v_k^{\text{bottom}}\tau^{\min}\right) \bmod l^{\text{total}}$
  end for{update the position of the bin system}
  $\Gamma \leftarrow$ set of flats to be inserted in the boxes next
  $\Delta \leftarrow$ set of flats to be dropped next
  update the streams $\mathbf{s}_j$ for $j = 1,2,\ldots,F$ {remove from each stream the codes that have been inserted in transport boxes}
  for all $j \in \Gamma$ **do**
    $t_j^{\text{insert}} \leftarrow t^{\text{crt}}$
  end for
  for all $m \in \Delta$ **do**
    $t_m^{\text{drop}} \leftarrow t^{\text{crt}}$
  end for
  $\tau_k \leftarrow \tau_k - \tau^{\min}$
  if $\tau_k = 0$ **then**
    $k \leftarrow k+1$
  end if

31: **end while**
Output: t

Next we present how we compute the variables τ_j^{insert} with $j \in \{1,\ldots,F\}$ and τ_m^{drop} with $m \in \{1,\ldots,N^{\text{boxes}}\}$ involved in determining the event-based model of **Algorithm 1**.

The variable τ_j^{insert} with $j \in \{1,2,\ldots,F\}$ is determined as follows.

- If the stream $\mathbf{s}_j$ is empty then we set $\tau_j^{\text{insert}} = \infty$ (note that if $\tau_j^{\text{insert}} = \infty$ then τ_j^{insert} will never be selected at step 10).
- Otherwise, we search for the next empty box transported towards the inserting device I_j and so that its position is larger than or equal to p_j^{inserter} and smaller than $p_{j+1}^{\text{inserter}}$ (note that if $j = F$ then we consider $p_{j+1}^{\text{inserter}} = l^{\text{total}}$).

  Note that a box can be either empty or loaded. Then we distinguish two cases:

  all boxes are loaded: If all the boxes are loaded, then no feeding event can occur, and therefore we set $\tau_j^{\text{insert}} = \infty$.

  there is an empty box: If such an empty box exists, then let this box be indexed by m. Furthermore, according to the operational constraint $\mathbf{C}_3$ that will be presented on page 32, in order to avoid executing the actions insert, drop, insert at the same time instant for the same box, a minimum period of $w^{\text{box}}/v^{\text{top}}$ time instants has to pass between two consecutive inserting events for box m with $m \in \{1,\ldots,N^{\text{boxes}}\}$. Then the time that will pass until the first flat inserting event from the buffer corresponding to stream $\mathbf{s}_j$ is determined as follows:

  feeding allowed for box m: If the last time when we had dropping from box m is larger than or equal to $t^{\text{crt}} - w^{\text{box}} v^{\text{top}}$ (so operational constraint $\mathbf{C}_3$ on page 32 is satisfied), then

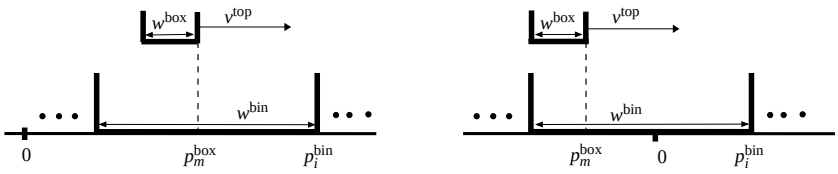
  $$\tau_j^{\text{insert}} = \frac{p_m^{\text{box}} - p_j^{\text{inserter}}}{v^{\text{top}}}$$

  where p_m^{box} is the position of box m, p_j^{inserter} is the position of the inserting device I_j, and t_m^{drop} is the time instant when we had the last dropping from box m.

  feeding not allowed for box m: If we are not allowed to feed box m due to constraint $\mathbf{C}_3$, then we set $\tau_j^{\text{insert}} = \infty$.

The variable τ_m^{drop} with $m \in \{1,2,\ldots,N^{\text{boxes}}\}$ is determined as follows. Assume that box m carries the flat that has to be dropped in destination bin i. Then we compute τ_m^{drop} as follows.

- If the box m is positioned on top of the bin i — i.e. $0 \le p_m^{\text{box}} - p_i^{\text{bin}} \le w^{\text{bin}} - w^{\text{box}}$ **or** $p_m^{\text{box}} + l^{\text{total}} - p_i^{\text{bin}} \le w^{\text{bin}} - w^{\text{box}}$ (see Figure 3.7) — then the flat can be immediately dropped. So, in this case we set $\tau_m^{\text{drop}} = 0$.
- Otherwise, we distinguish three cases as presented next. Note that we can drop as soon as box m is on top of bin i.



(a) $0 \leq p_m^{box} - p_i^{bin} \leq w^{bin} - w^{box}$ (b) $p_m^{box} + l^{total} - p_i^{bin} \leq w^{bin} - w^{box}$

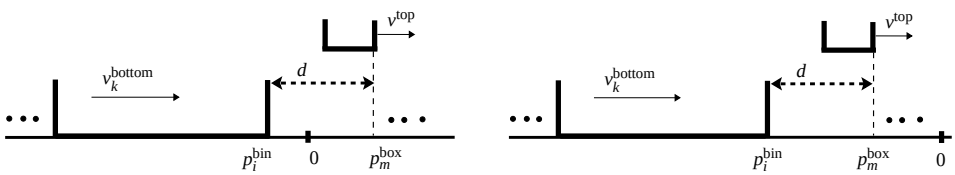
Figure 3.7: Positioning when the box transports the flat to be dropped in the bin below and the dropping is still allowed. We consider the positions of bins and the positions of boxes relative to the position of the first inserting device as illustrated in Figure 3.6.

case 1: $v_k^{bottom} > v^{top}$

In this case $\tau_m^{drop} = \dfrac{d}{v^{top} - v_k^{bottom}}$ where d is the shortest distance that the bin i and box ι have to travel at constant speed until they meet and dropping is allowed. So, d is the distance between the right-hand side of box m and the right-hand side of bin i.

The distance d is defined as follows (see also Figure 3.8):

$$d = \begin{cases} p_i^{bin} - p_m^{box} & \text{if } p_m^{box} \leq p_i^{bin} \\ l^{total} - p_m^{box} + p_i^{bin} & \text{otherwise.} \end{cases}$$

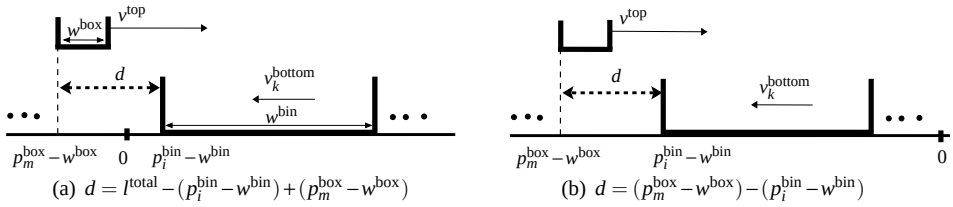


(a) $d = l^{total} - p_m^{box} + p_i^{bin}$ (b) $d = p_i^{bin} - p_m^{box}$

Figure 3.8: Distance to be traveled between the position of bin i and the position of box m when $v_k^{bottom} > v^{top}$.

case 2: $v_k^{bottom} < v^{top}$

In this case $\tau_m^{drop} = \dfrac{d}{v^{top} - v_k^{bottom}}$ where v_k^{bottom} is the velocity of the bottom system at the current time instant and d is the shortest distance that bin i and box ι have to travel at speed until they meet and dropping is allowed. So, d is the distance between the left-hand side of box m and the left-hand side of bin i and is defined as follows (for a more intuitive



(a) $d = l^{\text{total}} - (p_i^{\text{bin}} - w^{\text{bin}}) + (p_m^{\text{box}} - w^{\text{box}})$ (b) $d = (p_m^{\text{box}} - w^{\text{box}}) - (p_i^{\text{bin}} - w^{\text{bin}})$

Figure 3.9: Distance to be traveled between the position of bin i and the position of box m when $v_k^{\text{bottom}} < v^{\text{top}}$.

explanation see also Figure 3.9):

$$d = \begin{cases} l^{\text{total}} - (p_i^{\text{bin}} - w^{\text{bin}}) + (p_m^{\text{box}} - w^{\text{box}}) & \text{if } p_m^{\text{box}} - w^{\text{box}} \leq p_i^{\text{bin}} - w^{\text{bin}} \\ (p_m^{\text{box}} - w^{\text{box}}) - (p_i^{\text{bin}} - w^{\text{bin}}) & \text{otherwise} \end{cases}$$

where $p_m^{\text{box}} - w^{\text{box}}$ is the position of the left-hand side of box m and $p_i^{\text{bin}} - w^{\text{bin}}$ is the position of the left-hand side of bin i.

case 3: $v_k^{\text{bottom}} = v^{\text{top}}$

Since the box m is not positioned on top of bin i (otherwise we would not have reached this case) we set $\tau_m^{\text{drop}} = \infty$.

3.4 Constraints and control objective

The operational constraints derived from the mechanical and design limitations of the machine are the following:

C_1: the velocity of the bottom system is bounded between $-v^{\text{bottom,max}}$ (the bottom part moves in opposite direction than the top system at speed $v^{\text{bottom,max}}$) and $v^{\text{bottom,max}}$,

C_2: $\tau_k \geq \tau^{\text{ct}}$ for $k = 0, 1, \ldots, N$ with τ^{ct} the minimum time period for which the velocity of the bottom system has to stay constant,

C_3: the three actions insert, drop, insert cannot happen at the same time instant for the same box. Therefore, if at the same time instant we inserted a flat into, e.g., box m and we could immediately drop that flat, then box m can be fed again only after a minimum time period of $w^{\text{box}}/v^{\text{top}}$ time instants.

These constraints are denoted by $\mathscr{C}(\mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}) \leq 0$.

Recall that our goal is to increase the throughput of the flat sorting machine. Hence, let the control objective function J be defined as the throughput of the flat sorting system. For a given scenario and for the current state of the system, J depends on v^{bottom} and $\boldsymbol{\tau}$.

3.5 Control methods

In order to determine the speed of the bottom system that maximizes the throughput of the flat sorting system, we propose two model-based control approaches namely optimal control and model predictive control.

3.5.1 Optimal control

We now propose different variants of optimal control with gradually decreasing complexity such as optimal control with a piecewise constant speed on time intervals of variable length, optimal control with a piecewise constant speed on time intervals of constant length, and optimal control with a constant speed.

Optimal control with a piecewise constant speed on time intervals of variable length

One may first divide the period $[t_0, t_0 + \tau^{\text{sort,max}})$ into $N+1$ time intervals of variable length $\tau_0, \tau_1, .., \tau_N$ such that $\sum_{k=0}^{N} \tau_k = \tau^{\text{sort,max}}$. Define the time instants $t_{k+1} = t_k + \tau_k$ with $k \geq 0$. Then the piecewise constant control law $\mathbf{v}^{\text{bottom}} = [v_0^{\text{bottom}}\ v_1^{\text{bottom}} \dots v_N^{\text{bottom}}]^\top$, and the intervals τ_0, τ_1, … , τ_N on which v_0^{bottom}, v_1^{bottom}, …, v_N^{bottom} are constant have to be computed such that the throughput J is maximized. As a consequence, the optimal control problem is defined as follows:

$$
\begin{aligned}
\mathrm{P}_1: \quad & \max_{\mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}} \ J(\mathbf{x}(t_0), \mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}) \\
& \text{subject to} \\
& \quad \mathbf{t} = \mathscr{M}(\mathbf{x}(t_0), \mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}) \\
& \quad \mathscr{C}(\mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}) \leq 0
\end{aligned}
$$

Optimal control with a piecewise constant speed on time intervals of constant length

One may further simplify the problem P_1 by considering τ_k for $k = 0, 1, \dots, N-1$ equal to a fixed sampling time τ_{s}, and $\tau_N = \tau^{\text{sort,max}} - \sum_{k=0}^{N-1} \tau_{\text{s}}$. So, $\boldsymbol{\tau} = [\tau_{\text{s}}\ \tau_{\text{s}}\ \dots\ \tau_{\text{s}}\ \tau^{\text{sort,max}} - \sum_{k=0}^{N-1} \tau_{\text{s}}]^\top$. Accordingly, we define the optimal control problem with a piecewise constant speed on time intervals of constant length as follows:

$$
\begin{aligned}
\mathrm{P}_2: \quad & \max_{\mathbf{v}^{\text{bottom}}} \ J(\mathbf{x}(t_0), \mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}) \\
& \text{subject to} \\
& \quad \mathbf{t} = \mathscr{M}(\mathbf{x}(t_0), \mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}) \\
& \quad \mathscr{C}(\mathbf{v}^{\text{bottom}}, \boldsymbol{\tau}) \leq 0
\end{aligned}
$$

In both P_1 and P_2, the throughput increases monotonically with a smaller τ_{s} or with an increasing N until the best achievable throughput is reached. However, this comes at the cost of a higher computation time.

Optimal control with a constant speed

Now consider the simplest case of P_1 and P_2. For the entire stream of flats entering the system in one sorting round, the constant speed u^{ct} that maximizes the throughput is computed. This optimal control problem can be defined as follows:

$$
\begin{aligned}
\mathrm{P}_3: \quad & \max_{u^{\mathrm{ct}}} J(\mathbf{x}(t_0), u^{\mathrm{ct}}, \tau^{\mathrm{sort,max}}) \\
& \text{subject to} \\
& \quad \mathbf{t} = \mathscr{M}(\mathbf{x}(t_0), u^{\mathrm{ct}}, \tau^{\mathrm{sort,max}}) \\
& \quad \mathscr{C}(u^{\mathrm{ct}}, \tau^{\mathrm{sort,max}}) \leq 0
\end{aligned}
$$

The throughput obtained when solving P_3 is in general smaller than the one obtained when using optimal control with piecewise constant speed (described by P_1 and P_2), but the total computation time also decreases significantly.

3.5.2 Centralized MPC

In order to obtain a trade-off between the optimality of the throughput and the time required to compute the optimal velocity sequence of the bottom system, model predictive control (MPC) is introduced.

The MPC optimization problem can be solved by using a modified version of optimal control with a piecewise constant speed where at step k, the control sequence to be computed is v_k^{bottom}, $v_{k+1}^{\mathrm{bottom}}$, ..., $v_{k+N_{\mathrm{c}}-1}^{\mathrm{bottom}}$, and where τ_{k+j} is constant ($\tau_{k+j} = \tau_{\mathrm{s}}$) for $j = 0, 1 \ldots, N_{\mathrm{p}} - 1$. Then for $\boldsymbol{\tau}_k = [\tau_k \ \tau_{k+1} \ \ldots \ \tau_{k+N_{\mathrm{p}}-1}]^\top$, the MPC optimization problem can be written as follows:

$$
\begin{aligned}
\mathrm{P}_4: \quad & \max_{v_k^{\mathrm{bottom}}, v_{k+1}^{\mathrm{bottom}}, \ldots, v_{k+N_{\mathrm{c}}-1}^{\mathrm{bottom}}} J_{k,N_{\mathrm{c}},Np}(\mathbf{x}(t_k), [v_k^{\mathrm{bottom}} \ v_{k+1}^{\mathrm{bottom}} \ldots v_{k+N_{\mathrm{p}}-1}^{\mathrm{bottom}}]^\top, \boldsymbol{\tau}_k) \\
& \text{subject to} \\
& \quad \mathbf{t} = \mathscr{M}(\mathbf{x}(t_k), [v_k^{\mathrm{bottom}} \ v_{k+1}^{\mathrm{bottom}} \ldots v_{k+N_{\mathrm{p}}-1}^{\mathrm{bottom}}]^\top, \boldsymbol{\tau}_k) \\
& \quad \mathscr{C}([v_k^{\mathrm{bottom}} \ v_{k+1}^{\mathrm{bottom}} \ldots v_{k+N_{\mathrm{p}}-1}^{\mathrm{bottom}}]^\top, \boldsymbol{\tau}_k) \leq 0 \\
& \quad v_{k+j}^{\mathrm{bottom}} = v_{k+N_{\mathrm{c}}-1}^{\mathrm{bottom}} \qquad \text{for } j \geq N_{\mathrm{c}}
\end{aligned}
$$

where $J_{k,N_{\mathrm{c}},Np}$ is the throughput of the flat sorting system computed at time step k over the prediction horizon $[k\tau_{\mathrm{s}}, (k+N_{\mathrm{p}})\tau_{\mathrm{s}})$, and for a control horizon N_{c}.

The prediction horizon is determined using the following procedure[3]. Assume that one wants to determine in advance the optimal speed profile for sorting b flats — the variable b is defined as $b = \sum_{j=1}^{F} b_j$ where b_j is the number of flats that are actually in the buffer at the inserting device index j for $j = 1, 2, \ldots, F$, $0 \leq b_j \leq b_j^{\max}$ with $b_j^{\max}$ the maximum number of flats that can be in the buffer corresponding to the inserting device I_j. Then for $N_{\mathrm{c}}\tau_{\mathrm{s}}$ time units the velocity sequence v_k^{bottom}, $v_{k+1}^{\mathrm{bottom}}$, ..., $v_{k+N_{\mathrm{c}}-1}^{\mathrm{bottom}}$ is used. Each velocity $v_{k+j}^{\mathrm{bottom}}$, with $j = 0, 1, \ldots, N_{\mathrm{c}} - 1$, is applied during a sampling period of length τ_{s}. All this results in $b_1 \leq b$

[3] In this variant of MPC the prediction horizon depends on the number of flats to be sorted in advance.

flats being sorted in the period $[t_0, t_0 + N_c\tau_s)$. For sorting the rest of $b - b_1$ flats, the velocity of the bottom system is kept constant, equal to $v^{\text{bottom}}_{k+N_c-1}$. Consequently, the time τ^{horizon} needed to sort the b flats divided by τ_s determines the prediction horizon as follows (τ^{horizon} is determined via simulation): $N_p = \lceil \frac{\tau^{\text{horizon}}}{\tau_s} \rceil$ where $\lceil x \rceil$ denotes the smallest integer larger than or equal to x.

One of the advantages of MPC over optimal control with variable speed is given by a smaller computation time since $N_p\tau_s \leq \tau^{\text{sort,max}}$. However, this happens at the cost of a suboptimal throughput.

Optimization methods

In order to solve the optimization problems presented above in this subsection, and hence, to determine the optimal speed of the bottom system of a flat sorting machine, one may use the following Matlab functions: `fmincon` incorporated in the Optimization Toolbox, or `patternsearch` and `ga`, incorporated in the Genetic Algorithm and Direct Search Toolbox.

The `fmincon` function finds a local minimum of a smooth function based on gradient methods, while `patternsearch` and `ga` determine a local minimum of a non-smooth objective function.

3.6 Case study

In this section we compare the proposed control methods based on simulation examples. We will first detail the scenarios to be used for this comparison. Next we analyze the obtained results.

3.6.1 Scenarios

Recall that the velocity v^{top} of the top system is constant. Assume v^{top} to be equal to $1\,\text{m/s}$, while the velocity v^{bottom} of the bottom system is allowed to vary between $-0.5\,\text{m/s}$ and $0.5\,\text{m/s}$. It is also assumed that the width of the bin is four times the width of the box. The examples in the following sections involve $N^{\text{bins}} = 100$ bins and $N^{\text{boxes}} = 400$ boxes, while the length N^{flats} of the stream of flats that enter the sorting system equals 24000.

Furthermore, we assume that each destination bin has a unique identification code. Let these identification codes be: $1, 2, \ldots, N^{\text{bins}}$. At time instant t_0 we assign the identification codes to the destination bins so that $p_1^{\text{bin}} = 0\,\text{m}$, $p_2^{\text{bin}} = 0.4\,\text{m}$, $\ldots$, $p_{N^{\text{bins}}}^{\text{bin}} = l^{\text{total}} - 0.4\,\text{m}$. Also, the initial state of all the boxes is "empty" and the time when we had the last dropping event is initialized with $-\infty$ for all boxes (this initialization allows the system to insert the first flat into a box as soon as the box arrives in front of an inserting device).

Finally, we consider several scenarios for the same initial state of the system described above. According to these scenarios, the streams of codes consists of:

scenario 1: ordered codes (i.e., with the same order as the order of codes allocated to the bins). Since the width of the bin is four times the width of the box, a perfectly ordered stream of flats is a stream of the form e.g. $1, 1, 1, 1, 2, 2, 2, 2, \ldots, N^{\text{bins}}, N^{\text{bins}}, N^{\text{bins}}, N^{\text{bins}}, 1, 1, 1, 1, \ldots$. The order of the N^{bins} bins passing under

the first inserting device when the bottom system moves to the right is in this case $1,2,3,\ldots,N^{\text{bins}}$.

scenario 2: alternating sequences of random, and respectively ordered codes, e.g., $\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_m$ where if $\boldsymbol{\sigma}_j$ is a sequence of random codes, $1 \leq j < m$, then $\boldsymbol{\sigma}_{j+1}$ is a sequence of ordered codes and vice versa. The length of each sequence $\boldsymbol{\sigma}_j$ for $j = 1,2,\ldots,m$ is also chosen randomly. This scenario has been chosen due to the fact that the mail may be partially presorted;

scenario 3: completely random codes.

3.6.2 Results

For the scenarios above, the throughput of the flat sorting machine with a static bottom system (i.e., $v^{\text{bottom}} \equiv 0\,\text{m/s}$) has been listed in Table 3.1.

Table 3.1: The throughput $\text{J}^{\text{opt}}\,(\text{flats/s})$ *for the scenarios considered, when the bottom system of the sorting machine is static (*$v^{\text{bottom}} = 0\,\text{m/s}$*).*

Scenario	1 feeder	2 feeders
1	9.90	13.19
2	9.85	13.14
3	9.85	13.07

According to these results, one can notice that adding the second feeding device increases with about 33% the throughput of the flat sorting machine with a feeder only. Next we will determine whether using the proposed new design of the flat sorting machine and the proposed optimal control and model predictive control approaches increase the throughput even more.

Optimal control with a constant speed

In this section we compute the constant speed of the bottom system that optimizes the throughput for all the flats that enter the system in one sorting round. So, we solve the optimization problem P_3.

Figure 3.2 shows the throughput versus the velocity of the bottom system by discretizing the velocity with the sampling step of, e.g., $0.01\,\text{m/s}$. One may notice many variations of the throughput's amplitude. Hence, in order to optimize the throughput, a global or multi-start

Table 3.2: Comparison of throughput and computation time obtained by using the Matlab functions `fmincon`*,* `patternsearch`*, and* `ga` *for the set-up with two feeders.*

	$\text{J}^{\text{opt}}\,(\text{flats/s})$			computation time (s)		
Scenario	`fmincon`	`patternsearch`	`ga`	`fmincon`	`patternsearch`	`ga`
1	15.68	15.68	15.68	$2.39 \cdot 10^3$	$6.32 \cdot 10^2$	$1.74 \cdot 10^3$
2	15.81	15.84	15.84	$4.63 \cdot 10^3$	$1.20 \cdot 10^3$	$4.01 \cdot 10^3$
3	15.79	15.88	15.77	$5.27 \cdot 10^3$	$1.22 \cdot 10^3$	$4.20 \cdot 10^3$

local optimization method is required. Therefore, when solving the optimization problems P_1, P_2, P_3, P_4 corresponding to (1) optimal control with a piecewise constant speed on time intervals of variable length, (2) optimal control with a piecewise constant speed on time intervals of constant length, (3) optimal control with a constant speed, and (4) MPC with a piecewise constant speed on time intervals of constant length respectively, we have used as optimization algorithms:

- the *pattern search* algorithm incorporated in the Matlab optimization toolbox *Genetic Algorithm and Direct Search* implemented via the function `patternsearch` with multiple initial points,
- the *sequential quadratic programming* method, incorporated in the Matlab *Optimization* toolbox implemented via the function `fmincon` with multiple initial points.
- the *genetic* algorithm, incorporated in the Matlab optimization toolbox *Genetic Algorithm and Direct Search* implemented via the function `ga` with multiple runs.

In Table 3.2 we have listed the throughput and the corresponding computation[4] time obtained when solving P_3 by using the Matlab functions `fmincon` and `patternsearch` with three random initial points and when running the Matlab function `ga` three times. Note that we have used for this comparison only three initial points in order to keep the computation time low. Then, by comparing the throughput attained for each of the three optimization routines and the corresponding computation time, one may note that the `patternsearch` function gives the best results, i.e., the maximal throughput and the lowest computation time. Consequently, this optimization technique will be further used for solving the optimization problems.

The results listed in Table 3.2 also indicate that for the case where the flats are ordered (scenario 1), the obtained throughput is smaller than the throughput corresponding to scenario 2 and 3, where the flats have also random codes. This situation appears since both feeders feed the transport boxes with flats that have the following order of codes 1, 1 ,1, 1, 2, 2 ,2, 2, ..., 100, 100 ,100, 100 repeated 30 times (due to our initialization, see page 35). Furthermore, due to assumption $\mathbf{A}_4$ and due to our initialization, see page 35, the inserting devices corresponding to these two feeders are positioned over boxes aligned (at t_0) with the destination bins that have identification codes 1 and 51 respectively. Hence, the order that we use in scenario 1 does not help the sortation, but in contrary, it yields more loaded boxes passing under the inserting devices. Then the inserting rate will be lower than when considering random codes, and consequently, the throughput is also lower.

The improvement of the throughput obtained when using optimal control with a constant speed is defined with respect to the throughput obtained when the bottom system of the flat sorting machine is static. Simulations indicate that for a set-up with one and two feeders the improvement is about 1% and 20%, respectively. Hence, since the improvement obtained by using the proposed set-up with only one feeder is not substantial, only the system with two feeders is further considered. The results obtained when using optimal control with a constant speed are shown in Table 3.3.

Optimal control with a piecewise constant speed

Table 3.3 also lists the throughput obtained when solving P_2 corresponding to optimal control with a piecewise constant speed on time intervals of constant length $\tau_s = 3\,s$. Simulations show that using a smaller sampling interval does not further increase the throughput. Also, if one solves P_2 for $N > 4$ control variables, and P_1 corresponding to optimal control with a piecewise constant speed on time intervals of variable length, for $2N$ control variables, the resulted throughput is the same within an accuracy of 10^{-3}. However, the computation time when solving P_1 is much larger than the one required when solving P_2. So, optimal control with a piecewise constant speed on time intervals of constant length outperforms the optimal control with a piecewise constant speed on time intervals of variable length.

Model predictive control

When applying MPC, the smaller τ_s is chosen the bigger N_c has to be set in order to maximize the performance. If one does not want to increase N_c, blocking[5] can be also used. First we consider improving the performance and afterwards, we also take into account the computational effort.

To this aim we consider three cases:

- First, we try to obtain the best throughput. Therefore, we select a maximal prediction horizon — $b_j^{\max}$ for $j = 1,2,\ldots,F$ equals the length of the stream $\mathbf{s}_j$ that will be fed to the system using inserting device I_j — while N_c is set equal to N_p. To this aim, we compute the period length $\tau^{\text{sort,max}}$ needed to sort the entire stream of flats using optimal control with a constant speed. Accordingly, the prediction horizon is set to $\left\lceil \frac{\tau^{\text{sort,max}}}{\tau_s} \right\rceil$, while $N_c = N_p$. We have considered various lengths τ_s of the sampling time period. Based on simulation results, it has been noticed that for $\tau_s \ll 5\,s$ the resulting values of the throughput remain the same within an accuracy of 10^{-3}. This choice gives high performance, but is not feasible due to the high computational effort required.

- Secondly, we consider a prediction horizon determined by a buffer of $b_j^{\max} = 120$ flats for $j = 1,2,\ldots,F$, $N_c = N_p$, and $\tau_s = 5\,s$. Simulations indicate that applying MPC

Table 3.3: Comparison of throughput (expressed in (flats/s)*) obtained by using the proposed control methods for a set-up with two feeders.*

Scenario	Optimal control with piecewise constant speed	Optimal control with constant speed	MPC ($\tau_s = 5\,s$ $N_c = N_p$)	MPC ($\tau_s = 10\,s$ $N_c = 1$)
1	15.78	15.68	15.73	15.71
2	15.89	15.84	15.87	15.63
3	15.91	15.88	15.83	15.73

[4]The simulations were performed on a 3.0 GHz P_4 with 1 GB RAM.

[5]Instead of making the input to be constant beyond the control horizon only, one can force the input to remain constant during some predefined intervals.

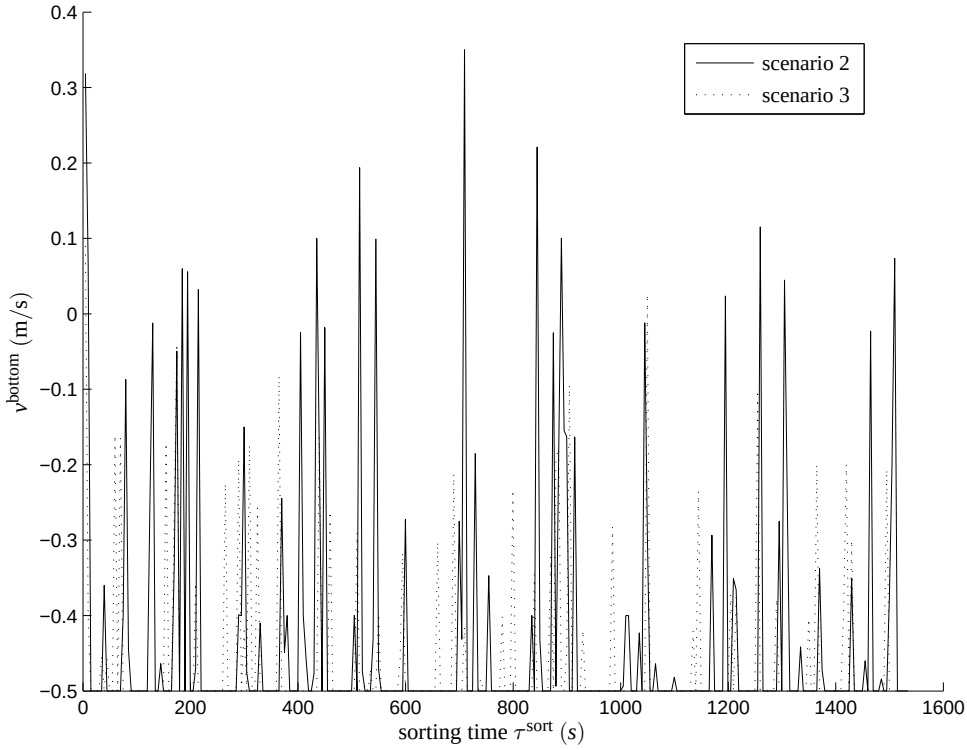


Figure 3.10: Speed evolution for the bottom part of the flat sorting machine (v^{bottom} is negative when the bottom system has the opposite direction of movement with respect to the top system). The profile has been determined when applying MPC with $N_{\text{p}} = N_{\text{c}}$, N_{p} determined by a buffer of 120 flats, and $\tau_{\text{s}} = 5\,\text{s}$.

with these characteristics gives already the throughput within 1% deviation of the throughput achieved when applying optimal control with a piecewise constant speed (see Table 3.3). Nevertheless, the computation time is still high.

- Finally, we consider a prediction horizon determined by a buffer of $b_j^{\max} = 120$ flats for $j = 1, 2, \ldots, F$, $N_{\text{c}} = 1$, and $\tau_{\text{s}} = 10\,\text{s}$. This choice produces real-time, but suboptimal results.

For the second case that we have considered when assessing the performance of MPC, we have also plotted the velocity profile of the bottom part of a flat sorting machine with two feeders, see Figure 3.10. This profile has been determined when applying MPC with $N_{\text{p}} = N_{\text{c}}$, N_{p} determined by a buffer of 120 flats, and $\tau_{\text{s}} = 5\,\text{s}$, for scenario 2 (with alternating sequences of random and ordered codes) and for scenario 3 (random codes only) respectively. According to the profile illustrated in Figure 3.10, during the entire sorting period ($\tau^{\text{sort,max}} = 1535\,\text{s}$ for scenario 2 and $\tau^{\text{sort,max}} = 1525\,\text{s}$ for scenario 3) the velocity of the bottom part varies between $0\,\text{m/s}$ and $0.5\,\text{m/s}$ when the bottom part has the opposite direction of movement with respect to the top part and between $0\,\text{m/s}$ and $0.4\,\text{m/s}$ when the bottom system has the same direction of movement as the top part. These results also

Table 3.4: Comparison of the average throughput and the average computation time of the proposed control methods for a flat sorting machine with two feeders.

Control method	total CPU time (s)	relative performance (%)
optimal control with piecewise constant speed on time intervals of variable length	$2.4 \cdot 10^6$	100
optimal control with piecewise constant speed on time intervals of constant length	$8.6 \cdot 10^5$	100
optimal control with constant speed	$1.0 \cdot 10^3$	99.60
MPC with $N_c = N_p$ and $\tau_s = 5\,\mathrm{s}$	$1.3 \cdot 10^4$	99.68
MPC with $N_c = 1$ and $\tau_s = 10\,\mathrm{s}$	$1.5 \cdot 10^3$	98.93
static bottom system	0	82.80

confirm that the flat sorting machine with a static bottom part is not optimal, but neither the one where $v^{\mathrm{bottom}} = -0.5\,\mathrm{m/s}$ is (in Section 3.6.4 we will also show that only increasing the relative speed between the top and the bottom part of a flat sorting machine does not maximize the throughput).

3.6.3 Discussion

Table 3.4 summarizes the results obtained when using the proposed optimal control approaches and model predictive control.

We have assumed that the maximal achievable throughput is obtained by using the optimal control with piecewise constant speed on time intervals of variable length. The performance of the other approaches was computed relative to this maximum. But, for each of the control methods to be compared, the throughput corresponding to the chosen scenarios varies. Therefore, in order to summarize the results of Table 3.3, the average throughput, $J = \frac{\sum_{j=1}^{N^{\mathrm{scenarios}}} J_j^{\mathrm{scenario}}}{N^{\mathrm{scenarios}}}$, is used in calculating the relative performance. The computation time is also averaged over all the considered scenarios.

The simulation results show that applying MPC gives a good trade-off between the computation time and the maximal achievable throughput. Also, the optimal control approach is not feasible, in the sense that in reality the entire stream of flats that enter the system in one sorting round is not known in advance. Only a finite buffer b of codes is known beforehand, with b depending on the maximal time allowed to prepare the flats for sorting. Therefore, in practice, MPC is the most suitable for determining the optimal velocity sequence of the bottom system. Moreover, note that the total sorting time satisfies the relation $1500\,\mathrm{s} \leq \tau^{\mathrm{sort,max}} \leq 1550\,\mathrm{s}$ for all proposed methods and for all considered scenarios. Hence, out of the MPC variants that we consider, only the one where $N_c = 1$ and $\tau_s = 10\,\mathrm{s}$ offers real-time results. However, also note that one can easily gain several orders of magnitude in the total computation time of MPC by using parallel computation when solving an optimization problem, better implementation, object coded programming languages instead of Matlab, or dedicated optimization algorithms. In future work also other control meth-

ods will be considered such as fast rule-based approaches, neural networks, see, e.g., [36], and fuzzy-based approaches, see, e.g., [63]. These approaches will then use the receding horizon principle (for more details see Section 5.3).

3.6.4 Influence of structural changes

In this subsection we analyze how the number of feeders, their position, and the velocity of the top system influence the throughput of a flat sorting machine.

Number of feeding devices

Consider a flat sorting machine with F inserting devices positioned symmetrically, $F \leq N^{\text{boxes}}$. Assume that both the transport boxes and the destination bins move with constant speed.

First we determine analytically the maximum throughput when considering two cases: the flats that enter the sorting part of the flat sorting machine have (1) perfectly ordered codes and (2) perfectly random codes. For both cases we compute the maximum throughput when the sorting system is in steady[6] state. Moreover, note that we want to compute the maximum throughput for a system where the velocity of the top system is fixed, v^{top}, while the velocity of the bottom system is bounded ($-v^{\text{bottom,max}} \leq v^{\text{bottom}} \leq v^{\text{bottom,max}}$ with $v^{\text{bottom,max}}$ the maximum speed that the bottom part can use).

Next we compute (via simulation) the throughput of a flat sorting machine where the top part moves with a constant speed and the bottom part moves with an optimal constant speed that is determined using optimal control with a constant speed.

Finally, we illustrate the maximum throughput determined analytically or via simulation for the case study of Section 3.6.1.

Perfectly ordered codes We now consider the case where the flats that enter the sorting part of the flat sorting machine have perfectly ordered codes. This means that the flats are ordered in such a way that once they enter the system, the time that they spend in the box is negligible.

Note that since the flats are dropped as soon as they enter the system, the maximum throughput is in fact bounded by the maximum feeding rate of the flat sorting machine. Let $\zeta^{\text{feed,max}}$ denote the maximum feeding rate. Note that $\zeta^{\text{feed,max}}$ is bounded due to the operational constraint $\mathbf{C}_3$ on page 32 — in order to avoid executing the actions insert, drop, insert at the same time instant for the same box, a minimum period of $w^{\text{box}}/v^{\text{top}}$ time instants has to pass between two consecutive inserting events for any box m with $m \in \{1,\ldots,N^{\text{boxes}}\}$. As a consequence, $\zeta^{\text{feed,max}}$ is bounded by the speed that the top system can use and by the width of the box. Assume that the transport boxes move with the constant speed v^{top}, then $\zeta^{\text{feed,max}} = \frac{v^{\text{top}}}{w^{\text{box}}}$.

Let $J^{\text{max,ordered}}$ be the maximal throughput when dealing with perfectly ordered codes to be fed into the machine. Then

$$J^{\text{max,ordered}} = F\zeta^{\text{feed,max}} = F\frac{v^{\text{top}}}{w^{\text{box}}}.$$

[6] In this thesis we say that a system is in its steady state if the system is at its equilibrium (the system is working in a regular and constant mode).

Example As example, consider a flat sorting machine where the top part moves with $1\,\mathrm{m/s}$, and where the width of a box is $0.1\,\mathrm{m}$. Suppose that there are $F = 4$ inserting devices, then the maximal throughput is

$$J^{\max,\text{ordered}} = 4 \cdot \frac{1}{0.1}\,\text{flats/s} = 40\,\text{flats/s}.$$

Perfectly random codes Next we determine a tighter upper bound for the case in which each of the F inserting devices feeds the system with perfectly random sequences of codes with respect to the destination bins. Assume the system in steady state. This computation goes as follows. In order to make the explanation more clear, we first assume that each of the F inserting devices feeds the system at a constant rate ζ_i (flats/s) with $i \in \{1,2,\ldots,F\}$. Then, since we consider perfectly random sequences of codes with respect to the identification codes of the destination bins, the flats are dropped uniformly along the bins. Let the part of the sorting system between two inserting points be called a "segment". Then, $\frac{\zeta_i}{F}$ flats per second are dropped along each segment of the sorting machine for $i = 1,\ldots,F$. As example we have illustrated in Figure 3.11 the feeding rate and the dropping rate along each segment of the sorting machine (solid arrows going in and out of the ellipse illustrate the feeding and dropping rates that correspond to inserting device I_1, dashed arrows for inserting device I_2, and so on).

Consider a point P fixed to the bottom system. For the simplicity of the explanation assume that this point is situated on the segment between I_F and I_1. However, note that

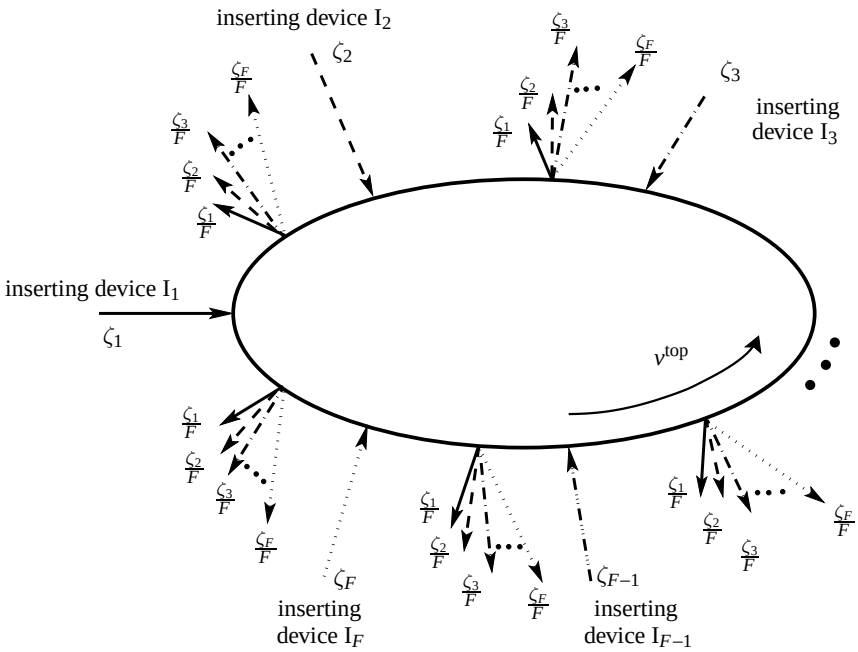


Figure 3.11: Top part of a flat sorting machine. Each of the F feeders feeds the system at rate ζ_i with $i \in \{1,\ldots,F\}$. The flats are dropped uniformly. Then $\sum_{i=1}^{F} \frac{\zeta_i}{F}$ flats per second are dropped between each two consecutive inserting points.

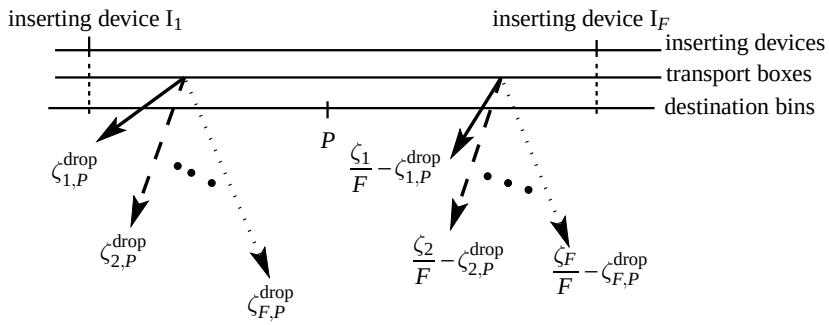


Figure 3.12: The point P is positioned between the inserting device I_1 and inserting device I_F. Then $\sum_{i=1}^{F} \zeta_i^{\text{drop}}$ flats per second are dropped between P and I_F.

a similar reasoning holds for any positioning of P along the destination bins. Moreover, assume that the bottom system moves in opposite direction than the top system. Note, however, that the reasoning that we make below is similar for the case where the bottom system moves in the same direction as the top system. But since we want to have a maximum relative speed between the top and the bottom system, see page 44, the second case is not relevant.

The boxes that pass a point P are either loaded or empty. These boxes are either transporting flats inserted into the system by feeder 1 up to feeder F or have been emptied somewhere between the inserting device I_1 and P, see Figure 3.12. Then, one can make the following remarks:

- The inserting device I_1 feeds the system in steady state with ζ_1 flats per second. Then, assuming that $\zeta_{1,P}^{\text{drop}}$ flats per second are dropped between the inserting device I_1 and P, $\zeta_1 - \zeta_{1,P}^{\text{drop}}$ loaded boxes per second will pass point P transporting flats inserted by the inserting device I_1.

- The inserting device I_2 feeds the system with ζ_2 flats per second. But $\frac{\zeta_2}{F}$ flats per second have been dropped along the segment bounded by the inserting devices I_2 and I_1. Assume that $\zeta_{2,P}^{\text{drop}}$ flats per second are dropped between I_1 and P. Then, $\zeta_2 - \frac{\zeta_2}{F} - \zeta_{2,P}^{\text{drop}} = (F-1)\frac{\zeta_2}{F} - \zeta_{2,P}^{\text{drop}}$ loaded boxes per second will pass point P transporting flats inserted by the inserting device I_2.

- ...

- The inserting device I_F feeds the system with ζ_F flats per second. But $(F-1)\frac{\zeta_F}{F}$ flats per second have been dropped along the segment bounded by the inserting devices I_F and I_1. Assume that $\zeta_{F,P}^{\text{drop}}$ flats per second are dropped between I_1 and P. Then $\frac{\zeta_F}{F} - \zeta_{F,P}^{\text{drop}}$ loaded boxes per second will pass point P transporting flats inserted by the inserting device I_F.

Hence one can write:

$$\zeta_1 + (F-1)\frac{\zeta_2}{F} + \cdots + 2\frac{\zeta_{F-1}}{F} + \frac{\zeta_F}{F} - \sum_{i=1}^{F} \zeta_{i,P}^{\text{drop}} = \varrho_P^{\text{box,loaded}}$$

where $\varrho_P^{\text{box,loaded}}$ is the number of loaded boxes that pass point P per second.

Now assume $\zeta_i = \zeta$ for $i = 1, 2, \ldots, F$. And since

$$1 + 2 + \cdots + F = \frac{F(F+1)}{2},$$

we can derive the relation:

$$\frac{\zeta}{F}\frac{F(F+1)}{2} = \sum_{i=1}^{F} \zeta_{i,P}^{\text{drop}} + \varrho_P^{\text{box,loaded}}$$

But

$$\sum_{i=1}^{F} \zeta_{i,P}^{\text{drop}} + \varrho_P^{\text{box,loaded}} = \varrho_P^{\text{box}}$$

where ϱ_P^{box} the number of boxes (loaded or empty) that pass point P per second. Then

$$\zeta = \frac{2}{F+1}\varrho_P^{\text{box}}.$$

Note that, in steady state, ζ is not only the feeding rate per inserting device, but also the actual throughput corresponding to that feeder. Hence, the actual throughput ($F\zeta$) of a flat sorting machine obtained for F streams of flats, the codes of which are uniformly distributed, is given by:

$$F\zeta = \frac{2F}{F+1}\varrho_P^{\text{box}}.$$

Next we determine the maximum feeding rate (and as a consequence the maximum throughput) that we can obtain, in steady state, for streams of uniformly distributed codes. We have two bottlenecks:

1. the feeding rate is bounded due to the operational constraint $\mathbf{C}_3$,
2. the dropping rate depends on the relative speed between the top and the bottom part of the flat sorting system.

As explained on page 41, the maximum feeding rate is

$$\zeta^{\text{feed,max}} = \frac{v^{\text{top}}}{w^{\text{box}}}.$$

Next we maximize the dropping rate. Note that the speed of the bottom system is bounded. As a consequence, in order to maximize the dropping rate, we have to set $v^{\text{bottom}} = -v^{\text{bottom,max}}$ (the bottom part moves with constant speed $v^{\text{bottom,max}}$ in opposite direction than the top system). Then the dropping rate will depend on the relative speed $v^{\text{rel,max}} = v^{\text{top}} + v^{\text{bottom,max}}$.

The maximum dropping rate is in fact the maximum number of boxes ($\varrho_P^{\text{box,max}}$) that pass a point P fixed on the bottom part of the machine per second, and is given by

$$\zeta^{\text{drop,max}} = \varrho_P^{\text{box,max}} = \frac{v^{\text{rel,max}}}{w^{\text{box}}}.$$

Let $J^{\text{max,rand}}$ denote the maximum throughput that we can obtain in this case. Then $J^{\text{max,rand}}$ is given by

$$J^{\text{max,rand}} = \min\left(F\zeta^{\text{feed,max}}, \frac{2F}{F+1}\zeta^{\text{drop,max}}\right) = \min\left(F \cdot \frac{v^{\text{top}}}{w^{\text{box}}}, \frac{2F}{F+1} \cdot \frac{v^{\text{top}} + v^{\text{bottom,max}}}{w^{\text{box}}}\right).$$

Example As an example, according to our case study the bottom system can move in both directions (clockwise and counterclockwise with maximum speed of $0.5\,\text{m/s}$). The velocity of the top part of the machine is $v^{\text{top}} = 1\,\text{m/s}$. The width of a box is $w^{\text{box}} = 0.1\,\text{m}$. So, $\zeta^{\text{feed,max}} = 10\,\text{flats/s}$. The maximum number of boxes that pass per second a point associated to the bottom part of the machine is obtained for the relative speed $v^{\text{rel,max}} = 1.5$ m/s. Then $\zeta^{\text{drop,max}} = \varrho^{\text{box,max}} = 15$ flats/s. Accordingly, when dealing with F inserting devices, we have

$$J^{\text{max,rand}} = \min\left(F \cdot 10\,\text{flats/s}, \frac{2F}{F+1} \cdot 15\,\text{flats/s}\right).$$

Assume $F = 1$ then $J^{\text{max,rand}} = 10\,\text{flats/s}$. Assume $F = 4$ then $J^{\text{max,rand}} = \frac{8}{5} \cdot 15\,\text{flats/s} = 24\,\text{flats/s}$.

Simulation results Next we compute via simulation the maximum throughput of a flat sorting machine where the top part of the flat sorting machine moves with $v^{\text{top}} = 1\,\text{m/s}$ and the bottom part of the machine moves with the constant speed determined by using optimal control with a constant speed. We consider optimal control with a constant speed to determine the optimal velocity of the bottom system due to the following reasons:

- We can now solve the optimization problem off-line.
- We are mainly interested in obtaining a high throughput, and not anymore in the tractability of the control method.
- Optimal control with a constant speed gives better results than MPC, but still with relatively low computation time (see Table 3.4).

Figure 3.13 illustrates the throughput achieved by using optimal control with a constant speed to determine the optimal velocity of the bottom part when dealing with the scenarios presented in Section 3.6.1. This throughput is plotted versus the number of feeders of the flat sorting machine, $F = 1, 2, \ldots, 50$. The maximum throughput obtained for scenario 1 (that consists of ordered codes) has been denoted by $J^{\text{max,sim,1}}$, the maximum throughput obtained for scenario 2 (that consists of alternating sequences of random and ordered codes) has been denoted by $J^{\text{max,sim,2}}$, and the maximum throughput obtained for scenario 3 (that consists of random codes) has been denoted by $J^{\text{max,sim,3}}$. Note that the throughput first increases with the number of feeders, but levels off around twenty feeders. Also, adding more feeders makes the system more complex and more expensive.

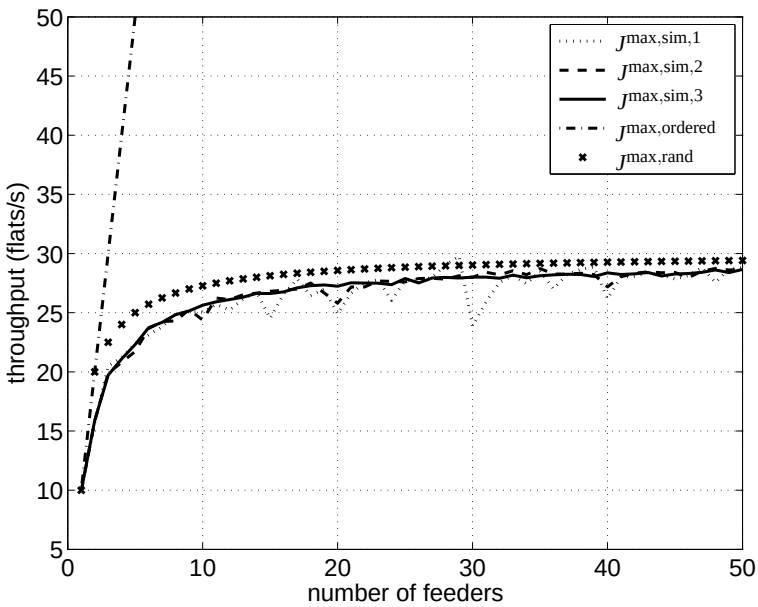


Figure 3.13: Throughput versus number of feeders.

Position of inserting devices

Next we analyze how the position of the inserting devices influences the throughput of a flat sorting machine. To this aim, we consider a flat sorting machine with F feeders. Then we have to determine the optimal positions of the $F-1$ inserting devices corresponding to the $F-1$ feeders when the position of the inserting device of the first feeder is fixed. This is then the problem of optimizing the sequence of distances $\mathbf{d} = [d_1\ d_2\ \ldots\ d_{F-1}]^\top$, where d_j, $j = 1,\ldots,F-1$, is the distance between the jth and the $(j+1)$st inserting device. This optimization problem can be formulated as follows:

$$
\begin{aligned}
\mathrm{P}_5\colon\quad & \max_{\mathbf{d}}\ J^{\mathrm{new}}(\mathbf{x}(t_0), u^{\mathrm{ct},*}, \mathbf{d}) \\
& \quad \text{subject to} \\
& \qquad \mathbf{t} = \mathscr{M}(\mathbf{x}(t_0), u^{\mathrm{ct},*}, \tau^{\mathrm{sort,max}}) \\
& \qquad \mathscr{C}(u^{\mathrm{ct},*}, \tau^{\mathrm{sort,max}}) \leq 0
\end{aligned}
$$

where J^{new} is the throughput of the flat sorting machine obtained when using the optimal constant velocity of the bottom system $u^{\mathrm{ct},*}$ for configuration $\mathbf{d}$ and for the given scenario, and $\mathbf{d}$ is the sequence of distances between the feeders defined above.

We consider a flat sorting machine with 3 feeders since this machine offers a good trade-off between the performance and the costs of the system. Accordingly, we consider three layouts for positioning the inserting devices of this flat sorting machine, described as follows (see also Figure 3.14):

case 1: the three existing feeders of the flat sorting machine have their inserting devices

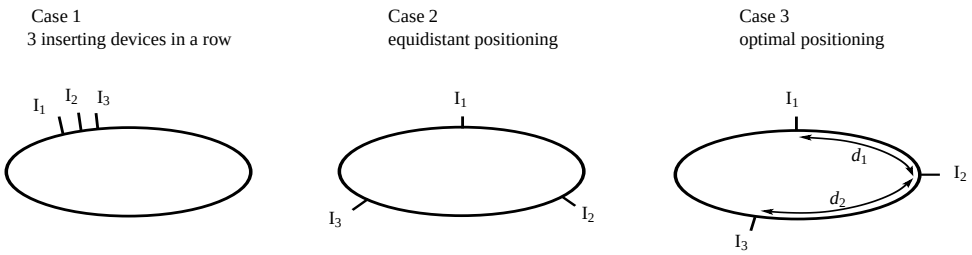


Figure 3.14: Layouts.

positioned one next to the other,

case 2: the inserting devices are positioned in an equidistant way,

case 3: the inserting devices are positioned at distances d_1 and d_2 computed by solving P_5 for the optimal constant speed of the bottom system for that specific scenario.

The throughput obtained for each of these configurations is illustrated in Table 3.5. One may notice that the throughput obtained in Table 3.5 for the first case (where the flat sorting machine has 3 feeders in a row) is very close to the throughput obtained by a sorting machine with only one feeder (see, e.g., Table 3.1 or Figure 3.2). Hence, the simulation results indicate that, e.g., 3 feeders in a row perform like if there is only one feeder, whereas by positioning them in an equidistant way one obtains the throughput within 1 % of the one obtained when solving P_5.

To conclude the analysis regarding the influence of the position of the inserting devices over the throughput of the flat sorting machine, we can generalize these results and note that positioning the inserting devices along the transport boxes equidistantly is a balanced arrangement. To support this conclusion we note that the throughput obtained for this configuration is typically very close to the one obtained when solving the optimization problem P_5.

Relative velocity between the transport boxes and the destination bins

In this section we analyze how an increase in the velocity of the top system, and consequently the increase of the relative velocity between the transport boxes and the destination bins, influences the throughput of a flat sorting machine. We consider a flat sorting machine with 1, 2, and 3 feeders respectively. The inserting devices are positioned at equidistant distances along the transport boxes. Assuming that the process of inserting a flat into a transport box can be performed also when the top system moves with higher speed, in this

Table 3.5: J^{opt} (flats/s) *obtained when positioning the feeders according to the considered layouts.*

Scenario	Case 1	Case 2	Case 3
1	9.98	20.49	20.49
2	9.97	19.87	19.94
3	9.94	19.71	19.85

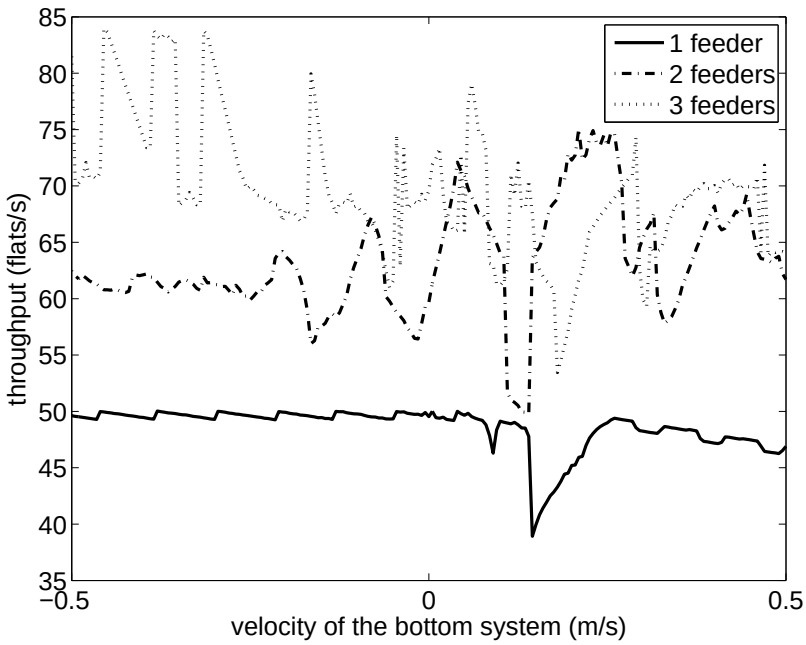


Figure 3.15: Throughput versus v^{bottom} when $v^{\text{top}} = 5\,\text{m/s}$ for a flat sorting machine with 1, 2, and 3 feeders respectively. The inserting devices are positioned at equidistant distances along the transport boxes.

section we consider $v^{\text{top}} = 5\,\text{m/s}$. In Figure 3.15, the throughput of the flat sorting machine is plotted versus the velocity of the bottom system, for a velocity range between -0.5 and $0.5\,\text{m/s}$ and a discretization sampling step of $0.01\,\text{m/s}$. We have plotted only the curves for the second scenario, since the plots for the first and third scenario are similar.

The plots illustrated in Figure 3.15 are nonlinear and nonsmooth, with many variations of the throughput's amplitude. Also, the simulation results indicate that the amplitude of the variation of the throughput's amplitude increases with the number of feeders of the flat sorting machine.

Hence, one can conclude with the following remark: only increasing the relative speed between the top and the bottom system does not maximize the thoughput. As a consequence, implementing advanced control methods to compute the optimal velocity of the bottom system is still required.

3.7 Summary

In this chapter we have considered sorting machines in mail sorting centers designed to handle large mail items such as newspapers, catalogs, and large letters. We have given a brief description of how flat sorting machines currently work. Afterwards, a new set-up has been proposed by making minor design changes, i.e., adding extra feeders and moving the bottom bin system. An event-driven model of the process has been determined, and

advanced control methods have been implemented so as to ensure the optimal speed of the bin movements.

We have also analyzed how the number of feeders, their position, and the velocity of the top system influence the throughput of the automated flat sorting machine. The simulation results show that just increasing the speed of the top system, and hence, the relative speed between the top and bottom system, does not have as immediate consequence an increase in the throughput. Hence, determining the optimal bottom velocity is still required so as to maximize the efficiency of the flat sorting machine. The results indicate that model predictive control is the most suitable control method to determine the velocity of the bottom system for the proposed flat sorting machine. To support this conclusion we make two remarks: (1) the simulation results show that applying MPC gives a good trade-off between the computation time and the maximal achievable throughput, (2) the optimal control approach is not feasible in the sense that in reality the entire stream of flats that enter the system in one sorting round is not known in advance (only a finite buffer of codes is known beforehand, the length of this buffer depends on the maximal time allowed to prepare the flats for sorting).

In future work also other control methods will be considered such as fast heuristic approaches, fuzzy control, case-based control, etc. In future work we will also include more complex dynamics of the system than those considered in this chapter (acceleration and deceleration of the speed at which the destination bins move).

Chapter 4

Baggage handling

In this chapter we consider state-of-the-art baggage handling systems in large airports. The chapter is structured as follows. In Section 4.1 we describe the automated baggage handling process and the current problems of a baggage handling system. Afterwards, in Section 4.2 we present the simplifying assumptions made in order to obtain a fast simulation, and the resulting nonlinear event-driven model of the DCV-based baggage handling system. This model will be later on used for model-based control. Next, in Section 4.3, we describe the operational constraints together with the control objective. Furthermore, in Section 4.4, we propose several control approaches for determining the route choice of bags through the baggage handling system. First we develop and compare centralized, decentralized, and distributed predictive methods that could be used to optimize the performance of the system. This results in a nonlinear, nonconvex, mixed integer optimization problem that is very expensive to solve in terms of computational effort. Therefore, we also propose an alternative approach for reducing the complexity of the computations by simplifying the nonlinear optimization problem and writing it as a mixed integer linear programming (MILP) optimization problem for which solvers are available that allow us to efficiently compute the global optimal solution. Finally, in order to reduce the computational requirements, we also propose two heuristic methods and a hierarchical control framework. The analysis of the simulation results and the comparison of the proposed control methods and control frameworks are elaborated in Section 4.5. Finally, in Section 4.6, we draw the conclusions of this chapter and we present possible directions for future research.

Parts of this chapter have been published in [76–78, 80–83, 85–89].

4.1 State-of-the-art solutions

In this section we briefly present the state-of-the-art in baggage handling systems and their control problems.

4.1.1 Process description

The main tasks of a baggage handling system are the following:

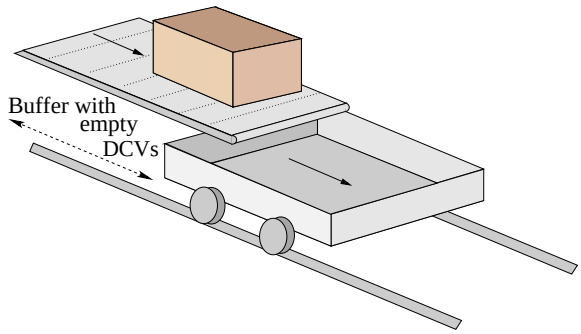


Figure 4.1: Loading a DCV.

- to transport baggage from the check-in area to the appropriate end point[1] and from there to the plane or between two different cargo terminals in case of transfer luggage,
- to transport baggage from a check-in desk or from a certain gate during transfers and to store them in the temporarily storage area (if the person checked in too early or the different flights have more than two hours waiting period),
- to transport baggage from the arrival gate to the baggage claim area.

The state-of-the-art technology used by baggage handling systems at airports to transport the bags in an automated way incorporates:

1. scanners that scan the (electronic) baggage tags on each piece of luggage,
2. baggage screening equipment for security scanning,
3. networks of conveyors equipped with junctions that route the bags through the system,
4. destination coded vehicles (DCVs). These vehicles are used in large airports only, where the distances between the check-in desks and the end points towards which the baggage has to be transported are too large (for these airports the conveyor systems are too slow, and therefore, a faster carrier is required for each bag).

As illustrated in Figure 4.1, a DCV is a metal cart with a plastic tub on top. These carts are propelled by linear induction motors mounted on the tracks. The DCVs transport the bags at high speed on a network of tracks. The nodes via which the DCVs enter the track network are called loading stations, the nodes via which the DCVs exit the network are called unloading stations, while all the other nodes in the network are called junctions. The section of track between two nodes is called link.

In this thesis we consider the general DCV-based baggage handling system sketched in Figure 4.2. This baggage handling system operates as follows: given a demand of bags (identified by their unique code) together with their arrival times at the loading stations, and the network of single-direction tracks, the route of each DCV in the network has to be computed subject to the operational and safety constraints presented in Section 4.3, such that all the bags to be handled arrive at their end points within given time windows. The

[1] An end point of the baggage handling system is the final part of the system where the bags are lined up, waiting to be loaded into containers and from there to the plane.

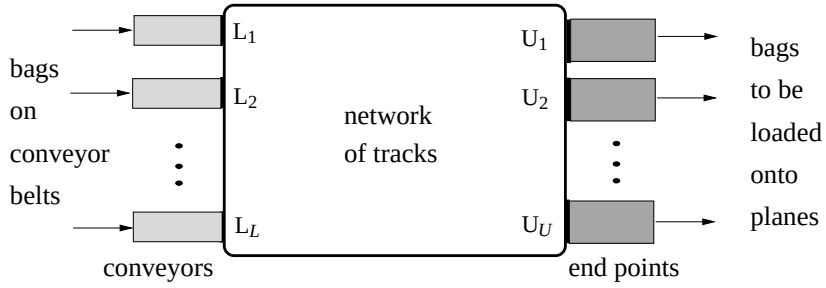


Figure 4.2: Baggage handling system using DCVs.

bags unloaded outside their end points' time window are then penalized as presented in Section 4.3.

We consider a system with L loading stations L_1, L_2, ..., L_L, U unloading stations U_1, U_2, ..., U_U, and S junctions S_1, S_2,..., S_S. Let us index the bags loaded onto DCVs at station L_ι with $\iota \in \{1,\ldots,L\}$ as $b^{\text{load}}_{\iota,1},\ldots,b^{\text{load}}_{\iota,N^{\text{load}}_\iota}$ with N^{load}_ι the number of bags that will be loaded at station L_ι during the entire simulation period. Then let $t^{\text{arrival}}_{\iota,j}$ denote the time instant when bag $b^{\text{load}}_{\iota,j}$ actually arrives at loading station L_ι ($t^{\text{arrival}}_{\iota,j} < t^{\text{arrival}}_{\iota,j+1}$ for $j = 1,\ldots,N^{\text{load}}_\iota - 1$). Then we define the L-tuple $\mathscr{T} = (\mathbf{t}^{\text{arrival}}_1, \mathbf{t}^{\text{arrival}}_2, \ldots, \mathbf{t}^{\text{arrival}}_L)$ that comprises the vectors of bag arrival times $\mathbf{t}^{\text{arrival}}_\iota = [t^{\text{arrival}}_{\iota,1} \ldots t^{\text{arrival}}_{\iota,N^{\text{load}}_\iota}]^\top$ with $\iota \in \{1,2,\ldots,L\}$.

4.1.2 Control problems

One can describe the control problems of a DCV-based baggage handling system in a hierarchical framework. At the lowest level the control problems are coordination and synchronization when loading a bag onto a DCV and when unloading it at its end point (in order to avoid damaging the bags or blocking the system). We assume the low-level controllers already present in the system. These low-level controllers are typically PID controllers and logic controllers that can stop the DCV when necessary. The velocity control of each DCV can be seen as a medium-level control problem. In this thesis we assume that each DCV has a medium-level speed controller on board. This controller ensures a minimum safety distance between DCVs and also holds DCVs at switching points, if required. So, we assume that the velocity of each DCV is always at its maximum, $v^{\max} = 20\,\mathrm{m/s}$, unless overruled by the local on-board collision avoidance controller. Finally, the higher-level control problems are route assignment for each DCV transporting a bag (and implicitly the switch control of each junction), line balancing (i.e., assignment of loading stations for each empty DCV such that all the loading stations have enough empty DCVs at any time instant), empty cart management (route assignment for each empty DCV), and prevention of buffer overflows.

In this chapter we focus on the higher-level control problem of determining the route choice for each DCV transporting a bag. Currently, the DCVs are routed through the system using routing schemes based on preferred routes. These routing schemes respond to the occurrence of predefined events as follows. Each junction has a logic controller and a lookup table storing preferred routes from that junction to all unloading stations. If the currently preferred route is blocked due to e.g. jams or buffer overflows, then the next to-be-preferred-

route of the lookup table is chosen and the switch out of that junction is toggled accordingly. In the research we conduct we do not consider such predefined preferred routes. Instead we develop advanced control methods to determine the optimal routing in case of dynamic demand.

In the literature, the route assignment problem has been addressed to a large extent for automated guided vehicles (AGVs). The algorithms developed for routing AGVs can be classified in three categories: algorithms for general path topology [22, 33, 39, 45, 47, 50, 73], algorithms for optimizing the path layout [30, 34, 41, 48], and algorithms for specific path topologies [5, 16]. Since we consider general DCV-based baggage handling systems, where the network of tracks is represented as a directed graph, we will look in more detail only to the first category where the path topology is general. These methods can also be classified into three categories:

1. static methods, where an entire path is considered to be occupied until a vehicle completes the tour [22, 33],
2. time-window-based methods, where a path segment can be used by different vehicles during different time windows [39, 45],
3. dynamic methods, where the utilization of any segment of path is dynamically determined using, e.g.,
    - incremental route planning — the next node to travel to (for each vehicle) is selected so that the distance from the vehicle's current position to its destination is minimal — [2, 73, 74],
    - enumeration of transportation plans — dispatching and assigning conflict-free routes— by means of dynamic programming [50].

Traditionally, the AGVs that execute the transportation tasks are controlled by a central server via wireless communication. Hence, the computational complexity of the centralized route choice controller increases with the number of vehicles to be routed. Therefore, [96] presents a decentralized architecture for routing AGVs through a warehouse. However, even for a small number of AGVs to be used for transportation (12 AGVs), the communication requirements are high. But in baggage handling systems the number of DCVs used for transportation is large (typically airports with DCV-based baggage handling systems have more than 700 DCVs). Hence, in practice, designing an on board route choice controller for each DCV is not yet tractable. Also, we do not deal with a shortest-path or shortest-time problem, since, due to the airport's logistics, we need the bags at their end points within given time windows.

The route choice problem for a DCV-based baggage handling system has been presented in [25] where an analogy to data transmission via internet is proposed, and in [38] where a multi-agent hierarchy has been developed. However, the analogy between routing DCVs through a track network and transmitting data over internet has limitations, see [25], while the latter reference, [38], does not focus on control approaches for computing the optimal route of DCVs, but on designing a multi-agent hierarchy for baggage handling systems and analyzing the communication requirements. Moreover, the multi-agent system of [38] is faced with major challenges due to the extensive communication required. Therefore, the goal of our work is to develop and compare efficient control approaches (viz., predictive

control methods and heuristic approaches) for route choice control of each DCV transporting bags to their end points in case of dynamic demand at loading stations. These control approached are developed in a centralized, a decentralized, and a distributed manner. Note that the control approach is said to be *decentralized* if the local control actions are computed without any communication or coordination between the local controllers, while the control approach is said to be *distributed* if additional communication and coordination between neighboring controllers is involved, see e.g. [94], [95].

4.2 Event-based model

In this section we present the simplifying assumptions and the continuous-time event-driven model to be used in order to determine the optimal route choice for DCVs in a baggage handling system.

4.2.1 Assumptions

Later on we will use the model for on-line model-based control. So, in order to obtain a balanced trade-off between a detailed model that requires large computation time and a fast simulation we make the following assumptions:

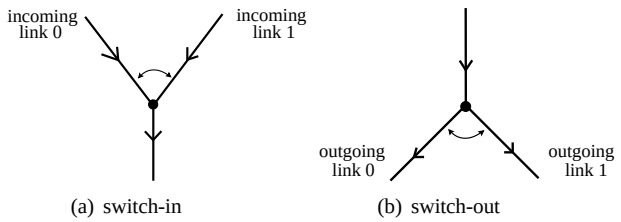
A_1: A sufficient number of DCVs are present in the system so that when a bag is at the loading station there is a DCV ready for transporting it.

A_2: Each junction S_s with $s \in \{1,2,\ldots,S\}$ has maximum 2 incoming links and 2 outgoing links, both indexed by $l \in \{0,1\}$ as sketched in Figure 4.3. If S_s has 2 incoming links then it also has a switch going into the junction (called switch-in hereafter). If S_s has 2 outgoing links then it has also a switch going out of the junction (called switch-out hereafter). Note that a junction can have only a switch-in, only a switch-out, or both a switch-in and a switch-out.

A_3: We assume each loading station to have only one outgoing link and each unloading station to have only one incoming link.

A_4: A route switch at a junction can be performed in a negligible time span.

A_5: The speed of a DCV is piecewise constant.



(a) switch-in (b) switch-out

Figure 4.3: Incoming and outgoing links at a junction.

A_6: The capacity of the end points is large enough that no buffer overflow can occur.

A_7: The flight numbers of the planes to which the bags have to be transported, are allocated to the end points when the process starts.

Next we will discuss for each of the assumptions above what to do if they do not hold:

A_1: Assumption $\mathbf{A}_1$ was made in order to simplify the route choice problem. In practice, the DCV-based baggage handling system does not have an unlimited number of DCVs in the system. Hence, in practice, one has also to efficiently solve the line balancing problem and to optimally assign routes to empty DCVs.

A_2: If one junction would have more than two incoming links or more than two outgoing links, in order to keep using the proposed model and control methods, one could virtually expand such a junction to junctions with maximum 2 incoming links and 2 outgoing links connected via links of length 0, or one could adapt the control methods themself.

A_3: If a loading station would have more than one outgoing link, then one can virtually expand a loading station into a loading station connected via a link of length 0 to a junction with a switch-out. Similarly, one can virtually expand an unloading station with more than one outgoing link.

A_4: Assumption $\mathbf{A}_4$ was made in order to simplify the explanation of the model. If $\mathbf{A}_4$ would not be valid, then one has to take into account the time needed to perform the switch when computing the control actions and the time period until the next event (see Section 4.2.2).

A_5: Assumption $\mathbf{A}_5$ is a fair approximation of the real speed of DCVs on the link segments between any 2 induction motors positioned consecutively.

A_6: The number of bags for each flight is known in advance (due to advance flight booking and historical data). So, one can assume without loss of generality that a sufficient number of end points is associated with a flight, or, if more than one end point is not available, then at the end point assigned to that specific flight there is human force or there are robots to load the bags into containers and from there to the plane.

A_7: During one day several flights are typically assigned to an end point. However, the time window when an end point is available for each of those flights is known beforehand (it is also known from historical data). Hence, this assumption is not restrictive.

So, these approximations are reasonable and give a good approximation of the real baggage handling system.

4.2.2 Model

In order to obtain a fast simulation, we write the model as an event-driven system consisting of a continuous part describing the movement of the individual vehicles transporting bags through the network, and of the following discrete events:

- loading a new bag into the system,
- unloading a bag that arrives at its end point,
- crossing a junction,
- updating the position of the switch-in at a junction,
- updating the position of a switch-out at a junction,
- updating the velocity of a DCV.

Let N^{bags} be the number of bags that the baggage handling system has to handle and let $N^{\text{bags,crt}}$ be the total number of bags that entered the track network up to the current time instant $t^{\text{crt}} < t_0 + \tau^{\text{max\_sim}}$ with t_0 the initial simulation time and $\tau^{\text{max\_sim}}$ the maximum simulation period. Also, let DCV_i denote the DCV that transports the ith bag that entered the track network up to the current time instant, $i \leq N^{\text{bags,crt}}$. Note that if two or more bags are loaded onto DCVs at the same time instant t, we order the DCVs according to the index of the loading stations (DCV_i will then denote the DCV transporting the bag loaded at the loading station with the smallest index, DCV_{i+1} will denote the DCV transporting the bag loaded at the loading station with the next smallest index, and so on).

The **state** of the DCV-based baggage handling system consists of the link on which each of the DCVs travel, their speed and their position on that link, and the position of the switch-in and switch-out at each junction. Furthermore, the **input** of the system consists of the demand of bags together with their arrival times at the loading stations and of the control variables. Note that, depending on the control method, the **control variables** can be the switch positions or the time periods after which we toggle the position of the switch as presented later on. Finally, the **output** consists of the time instants when we load and unload each of the bags to be handled (these time instants will be collected into a vector denoted by **t**, they will be derived via simulation and will be used later on when measuring the performance of the system).

Algorithm 2. Model of the baggage handling system

Input: the demand of bags together with their arrival times at the loading stations, and the initial state of the system

$t^{\text{crt}} \leftarrow t_0$
while $t^{\text{crt}} < t_0 + \tau^{\text{max\_sim}}$ **do**
  for $\iota = 1$ to L **do**
    $\tau_\iota^{\text{load}} \leftarrow$ time that will pass until the next loading event of L_ι
  end for
  for $\upsilon = 1$ to U **do**
    $\tau_\upsilon^{\text{unload}} \leftarrow$ time that will pass until the next unloading event of U_υ
  end for
  for $s = 1$ to S **do**
    $\tau_s^{\text{cross}} \leftarrow$ time that will pass until the next DCV-crosses-S_s event
    $\tau_s^{\text{sw\_in}} \leftarrow$ time that will pass until the next switch-in event at S_s
    $\tau_s^{\text{sw\_out}} \leftarrow$ time that will pass until the next switch-out event at S_s
  end for
  for $i = 1$ to $N^{\text{bags,crt}}$ **do**
    $\tau_i^{\text{v\_update}} \leftarrow$ time that will pass until the next velocity-update event of DCV_i

16: **end for**

17: $\tau^{\min} \leftarrow \min\Big(\min_{\iota=1,2,\ldots,L} \tau_{\iota}^{\text{load}}, \min_{\upsilon=1,2,\ldots,U} \tau_{\upsilon}^{\text{unload}}, \min_{s=1,2,\ldots,S} \tau_{s}^{\text{cross}}, \min_{s=1,2,\ldots,S} \tau_{s}^{\text{sw\_in}}, \min_{s=1,2,\ldots,S} \tau_{s}^{\text{sw\_out}}, \min_{i=1,\ldots,N^{\text{bags,crt}}} \tau_{i}^{\text{v\_update}}\Big)$

18: $t^{\text{crt}} \leftarrow t^{\text{crt}} + \tau^{\min}$

19: take action (i.e. load, unload, cross junction, switch-in update, switch-out update, velocity update)

20: update the state of the system

21: **end while**

Output: t

Note that if multiple events occur at the same time, then we take all these events into account when updating the state of the system at step 20.

Next we describe the variables involved in determining the event-based model of **Algorithm 2**. This goes as follows, where $\iota \in \{1,2,\ldots,L\}$, $\upsilon \in \{1,2,\ldots,U\}$, $s \in \{1,2,\ldots,S\}$, and $i \in \{1,2,\ldots,N^{\text{bags,crt}}\}$:

$\tau_{\iota}^{\text{load}}$**:** If there is no bag coming towards loading station L_{ι}, then $\tau_{\iota}^{\text{load}} = \infty$. Otherwise, a conveyor transports bags towards loading station L_{ι}. Recall that we assume that there are sufficient DCVs present in the system so that when a bag is at the loading station there is a DCV ready for transporting it. Then, for the current state of the system at time instant t^{crt}, the time period $\tau_{\iota}^{\text{load}}$ is equal to $\max\left(t_{\iota,j}^{\text{arrival}} - t^{\text{crt}}, t_{\iota,j,i}^{\text{safe}}\right)$ where $t_{\iota,j}^{\text{arrival}}$ denotes the time instant when bag $b_{\iota,j}^{\text{load}}$ actually arrives at loading station L_{ι}, $j-1$ is the number of bags that have been already loaded from L_{ι} (so, the next bag to be loaded at L_{ι} has local index j), and $\tau_{\iota,j,i}^{\text{safe}}$ expresses the time period that has to pass until it is safe for bag $b_{\iota,j}^{\text{load}}$ to be loaded onto a DCV. This duration is given by:

$$\tau_{\iota,j,i}^{\text{safe}} = \begin{cases} 0 & \text{if } d_{\iota,j-1}^{\text{travel}} \geq d^{\min} \\ \dfrac{d^{\min} - d_{\iota,j-1}^{\text{travel}}}{\max\left(v^{\text{jam}}, v_{\iota,j-1}^{\text{load}}\right)} & \text{otherwise} \end{cases}$$

where $d^{\min}$ is the minimum safe distance between DCVs, $d_{\iota,j-1}^{\text{travel}}$ is the position of the DCV transporting bag $b_{\iota,j-1}^{\text{load}}$ on the outgoing link of loading station L_{ι}, $v_{\iota,j-1}^{\text{load}}$ is the velocity of that DCV, and $v^{\text{jam}} \ll 1\,\mathrm{m/s}$ is the speed to be used in case of jam. The speed v^{jam} is determined based on empirical data.

$\tau_{\upsilon}^{\text{unload}}$**:** The time period that will pass until the next unloading event occurs at unloading station U_{υ} is given by:

$$\tau_{\upsilon}^{\text{unload}} = \frac{d_{\upsilon}^{\text{link}} - d_{\upsilon}^{\text{travel,closest}}}{v_{\upsilon}^{\text{closest}}}$$

where $d_{\upsilon}^{\text{link}}$ is the length of the incoming link of unloading station U_{υ}, $d_{\upsilon}^{\text{travel,closest}}$ is the position of the DCV closest to U_{υ} on the incoming link of U_{υ}, and $v_{\upsilon}^{\text{DCV}}$ is the current speed of this DCV. If there is no DCV on the incoming link of U_{υ}, then $\tau_{\upsilon}^{\text{unload}} = \infty$ by definition.

τ_s^{cross}**:** Consider the switch into junction S_s to be positioned at the current time on the incoming link $l \in \{0,1\}$ of S_s. Then the time that will pass until the next DCV crosses S_s is given by:

$$\tau_s^{\text{cross}} = \begin{cases} \dfrac{d_{s,l}^{\text{link}} - d_{s,l}^{\text{travel,closest}}}{\max\left(v^{\text{jam}}, v_{s,l}^{\text{closest}}\right)} & \text{if there is a DCV on link } l \text{ into } \mathrm{S}_s \\ \infty & \text{otherwise} \end{cases}$$

where $d_{s,l}^{\text{link}}$ is the length of the incoming link l of junction S_s, $d_{s,l}^{\text{travel,closest}}$ is the position of the DCV closest to S_s on the incoming link l of S_s, and $v_{s,l}^{\text{closest}}$ is the velocity of that DCV.

$\tau_s^{\text{sw\_in}}$**,** $\tau_s^{\text{sw\_out}}$**:** Once the toggle command of switch-in and switch-out is given, the position of the switch-in and switch-out is toggled after $\tau_s^{\text{sw\_in}}$ and $\tau_s^{\text{sw\_out}}$ time units respectively. Assume that the toggle commands are given at $t^{\text{sw\_in}} \geq t^{\text{crt}}$ and $t^{\text{sw\_out}} \geq t^{\text{crt}}$. Then $\tau_s^{\text{sw\_in}}$ and $\tau_s^{\text{sw\_out}}$ are given by:

$$\tau_s^{\text{sw\_in}} = \max\left(t^{\text{sw\_in}}, t_s^{\text{sw\_in\_prev}} + \tau^{\text{switch}}\right) - t^{\text{crt}}$$
$$\tau_s^{\text{sw\_out}} = \max\left(t^{\text{sw\_out}}, t_s^{\text{sw\_out\_prev}} + \tau^{\text{switch}}\right) - t^{\text{crt}}$$

where τ^{switch} is the minimum time period after which the switch at a junction can be toggled, and where $t_s^{\text{sw\_in\_prev}}$ and $t_s^{\text{sw\_out\_prev}}$ are respectively the time instants when the switch-in and the switch-out at junction S_s have been toggled last.

$\tau_i^{\text{v\_update}}$**:** We calculate the duration $\tau_i^{\text{v\_update}}$ according to the cases enumerated below — $d^{\min}$ is the minimum safe distance between DCVs and $d_{\text{DCV}_i}^{\text{travel}}$ is the position of DCV_i on the incoming link of S_s.

- Assume DCV_i to be traveling towards junction S_s on link $l \in \{0,1\}$, with no other DCV traveling in front of DCV_i on the same link l. Then we distinguish two situations:
  1. $v_{\text{DCV}_i} < v^{\max}$ and $d_{s,l}^{\text{link}} - d_{\text{DCV}_i}^{\text{travel}} > d^{\min}$.
  2. $v_{\text{DCV}_i} > 0$, $d_{s,l}^{\text{link}} - d_{\text{DCV}_i}^{\text{travel}} \leq d^{\min}$, and the switch-in at S_s is *not* positioned on the incoming link l that DCV_i travels.

  In both cases the velocity of DCV_i has to be updated immediately, $\tau_i^{\text{v\_update}} = 0$. Hence, the velocity of DCV_i will be updated as follows: $v_{\text{DCV}_i} \leftarrow v^{\max}$ for the first case and $v_{\text{DCV}_i} \leftarrow 0$ for the latter.
- Let $\text{DCV}_i^{\text{prev}}$ denote the DCV traveling on the same incoming link l of S_s as DCV_i, in front of DCV_i, with no other DCV between them. Also, let $d_{\text{DCV}_i^{\text{prev}}}^{\text{travel}}$ denote the position of $\text{DCV}_i^{\text{prev}}$ on link l. We distinguish two situations:
  1. $v_{\text{DCV}_i} < v^{\max}$ and $d_{\text{DCV}_i^{\text{prev}}}^{\text{travel}} - d_{\text{DCV}_i}^{\text{travel}} > d^{\min}$
  2. $v_{\text{DCV}_i} > v_{\text{DCV}_i^{\text{prev}}}$ and $d_{\text{DCV}_i^{\text{prev}}}^{\text{travel}} - d_{\text{DCV}_i}^{\text{travel}} \leq d^{\min}$

Then the velocity of DCV_i has to be updated immediately for the first case and after

$$\tau_i^{\text{v\_update}} = \frac{d^{\text{travel}}_{\mathrm{DCV}_i^{\text{prev}}} - d^{\text{travel}}_{\mathrm{DCV}_i} - d^{\min}}{\max\left(v^{\text{jam}}, v^{\max} - v_{\mathrm{DCV}_i^{\text{prev}}}\right)}$$

for the latter one. The velocity of DCV_i will be updated as follows: $v_{\mathrm{DCV}_i} \leftarrow v^{\max}$ for the first case and $v_{\mathrm{DCV}_i} \leftarrow v_{\mathrm{DCV}_i^{\text{prev}}}$ for the latter.

For any other case, we set $\tau_i^{\text{v\_update}} = \infty$.

According to the model, for each bag that has to be handled, we compute the time instants when each bag enters and exits the track network. Let t_i^{load} denote the time instant when the ith bag that entered the track network is loaded onto a DCV (so, this is DCV_i) and let t_i^{unload} denote the time instant when the same bag is unloaded at its end point. Then we denote two models of the baggage handling system which will be used for (1) route control — we determine a route for each DCV, and consequently, the switch will be positioned so that each DCV travels on the assigned route — and (2) switch control — we determine switch positions over the simulation period — respectively:

$$\mathbf{t} = \mathscr{M}^{\text{route\_ctrl}}\left(\mathscr{T}, \mathbf{x}(t_0), \mathbf{r}\right)$$

or

$$\mathbf{t} = \mathscr{M}^{\text{switch\_ctrl}}\left(\mathscr{T}, \mathbf{x}(t_0), \mathscr{U}\right)$$

where:

- $\mathbf{t} = [t_1^{\text{load}} \ldots t_{N^{\text{bags}}}^{\text{load}}\ t_1^{\text{unload}} \ldots t_{N^{\text{bags}}}^{\text{unload}}]^\top$.
- $\mathscr{T} = (\mathbf{t}_1^{\text{arrival}}, \mathbf{t}_2^{\text{arrival}}, \ldots, \mathbf{t}_L^{\text{arrival}})$ defined in Section 4.1.1.
- $\mathbf{x}(t_0)$ is the initial state of the system with t_0 the initial simulation time.
- $\mathbf{r}$ is the route control sequence defined as follows: assume that there is a fixed number R of possible routes from a loading station to an unloading station and that the R routes are numbered $1, 2, \ldots, R$. Let $r(i) \in \{1, 2, \ldots, R\}$ denote the route of DCV_i. Then the route sequence is represented by $\mathbf{r} = [r(1)\, r(2) \cdots r(N^{\text{bags}})]^\top$.
- $\mathscr{U}$ is the switch control input for the entire network defined as $\mathscr{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_S)$ with $\mathbf{u}_s = [u_s^{\text{sw\_in}}(1) \ldots u_s^{\text{sw\_in}}(N^{\text{bags}})\, u_s^{\text{sw\_out}}(1) \ldots u_s^{\text{sw\_out}}(N^{\text{bags}})]^\top$, $s \in \{1, 2, \ldots, S\}$, if junction S_s has both a switch-in and a switch-out, $\mathbf{u}_s = [u_s^{\text{sw\_in}}(1) \ldots u_s^{\text{sw\_in}}(N^{\text{bags}})]^\top$ if junction S_s has only a switch-in, $\mathbf{u}_s = [u_s^{\text{sw\_out}}(1) \ldots u_s^{\text{sw\_out}}(N^{\text{bags}})]^\top$ if junction S_s has only a switch-out.

4.3 Constraints and control objective

In this section we present the safety and operational constraints of a DCV-based baggage handling system, together with the control objective to be used when comparing the proposed control methods.

Operational constraints The operational constraints are derived from the mechanical and design limitations of the system. Such constraints are:

C$_1$**:** A DCV can transport only one bag at the time.

C$_2$**:** A bag can be loaded onto a DCV only if there is an empty DCV under the loading station. This means that if there is a traffic jam at a loading station, then no loading event can occur at that loading station.

These constraints (**C**$_1$ and **C**$_2$) have been already included when modeling the system.

Next we write the set of inequalities describing the operational constraints of a DCV-based baggage handling system as follows:

$$\mathscr{C}(\mathbf{t}) \leq 0 \tag{4.1}$$

Examples of operational constraints described by (4.1) are:

C$_3$**:** A switch at a junction has to wait at least τ^{switch} time units after a switch has occurred (before a new switch can take place), in order to avoid the quick and repeated movement back and forth of the switch which may lead to mechanical damage.

C$_4$**:** The speed of each DCV is bounded between 0 and $v^{\max}$.

Control objective Since the baggage handling system performs successfully if all the bags are transported to their end point before a given time instant, from a central point of view, the primary objective is the minimization of the overdue time. A secondary objective is the minimization of the additional storage time at the end point. This objective is required due to the intense utilization of the end points in a busy airport. Hence, one way to construct the objective function J_i^{pen} corresponding to the bag with index i, $i \in \{1,2,\ldots,N^{\text{bags}}\}$, is to penalize the overdue time and the additional storage time. Accordingly, we define the following penalty for bag index i, see Figure 4.4:

$$J_i^{\text{pen}}(t_i^{\text{unload}}) = \sigma_i \max(0, t_i^{\text{unload}} - t_i^{\text{close}}) + \lambda_1 \max(0, t_i^{\text{close}} - \tau_i^{\text{open}} - t_i^{\text{unload}}) \tag{4.2}$$

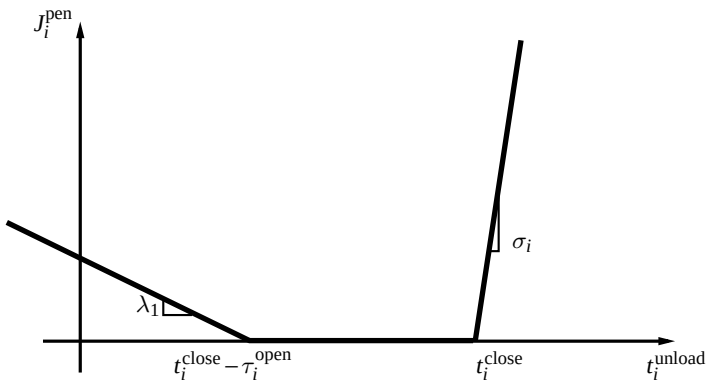


Figure 4.4: Objective function J_i^{pen}.

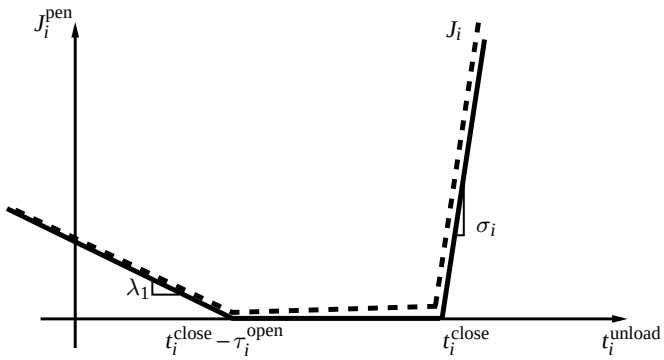


Figure 4.5: Objective functions J_i^{pen} and J_i.

where t_i^{close} is the time instant when the end point closes and the bags are loaded onto the plane, σ_i is the static priority of bag index i (the flight priority), and τ_i^{open} is the maximum possible length of the time window for which the end point corresponding to bag index i is open for that specific flight. The weighting parameter $\lambda_1 > 0$ expresses the penalty for the additionally stored bags.

However, the above performance function has some flat parts, which yield difficulties for many optimization algorithms. Therefore, in order to get some additional gradient and also minimize the energy consumption, we also include the time that a bag spends in the system. This results in see Figure 4.5:

$$J_i(t_i^{\text{unload}}) = J_i^{\text{pen}}(t_i^{\text{unload}}) + \lambda_2(t_i^{\text{unload}} - t_i^{\text{load}}) \tag{4.3}$$

where λ_2 is a small weight factor ($0 < \lambda_2 \ll 1$).

The final objective function to be used when comparing the proposed control approaches is given by:

$$J^{\text{tot}}(\mathbf{t}) = \sum_{i=1}^{N^{\text{bags,sim}}} J_i^{\text{pen}}(t_i^{\text{unload}}) \tag{4.4}$$

where $N^{\text{bags,sim}}$ is the number of bags that reached their end point during the simulation period $[t_0, t_0 + \tau^{\text{sim}})$, where τ^{sim} is either the time instant when all the bags have been handled (and then $N^{\text{bags,sim}} = N^{\text{bags}}$) or $\tau^{\text{sim}} = \tau^{\text{max\_sim}}$.

4.4 Control methods

In this section we develop and compare centralized, decentralized, and distributed predictive methods that could be used to optimize the performance of the system. The centralized control method results in a nonlinear, nonconvex, mixed integer optimization problem that is very expensive to solve in terms of computational effort. Therefore, we also propose an alternative approach for reducing the complexity of the computations by approximating the nonlinear optimization problem by a mixed integer linear programming (MILP) problem. Finally, in order to reduce the computational requirements, we also develop two heuristic methods and a hierarchical control framework.

4.4.1 Optimal control

Assume that there is a fixed number R of possible routes from a loading station to an unloading station and that the R routes are numbered $1,2,\ldots,R$. Let $r(i) \in \{1,2,\ldots,R\}$ denote the route of DCV_i. Then the route sequence is represented by $\mathbf{r} = [r(1)\, r(2) \cdots r(N^{\text{bags}})]^\top$.

The optimal control problem is defined as follows:

$$\begin{aligned}
&\min_{\mathbf{r}} J^{\text{tot}}(\mathbf{t}) \\
&\text{subject to} \\
&\qquad \mathbf{t} = \mathscr{M}^{\text{route_ctrl}}\big(\mathscr{T}, \mathbf{x}(t_0), \mathbf{r}\big) \\
&\qquad \mathscr{C}(\mathbf{t}) \leq 0
\end{aligned}$$

But computing the optimal route of each DCV transporting bags through the network so as to minimize the performance index J^{tot} requires extremely high computational effort as we have shown in Section 4.5.1. In practice, this problem becomes intractable when the number of possible routes and the number of bags to be transported are large.

4.4.2 Centralized MPC

We define now a variant of MPC, where k is not a time index, but a bag index. In this context bag step k corresponds to the time instant t_k^{load} when the kth bag has just entered the track network — if $k = 0$ bag step k corresponds to the time instant t_0. For this variant of MPC, the horizon N corresponds to the number of bags for which we look ahead, while computing the control inputs $r(k+1)$, $r(k+2)$, $\ldots$, $r(k+N)$ where $r(k+j)$ with $j \in \{1,2,\ldots,N\}$ represents the route of DCV_{k+j} (from a given loading station to the corresponding unloading station). Next, we implement all the computed control samples, and accordingly we shift the horizon with N steps. So, once we have assigned a route to a DCV, the route of that DCV cannot be later on changed.

The total objective function of centralized MPC is then defined as:

$$J_{k,N}^{\text{Centr\_MPC}}(\mathbf{t}(k)) = \sum_{i=1}^{k+N} J_i(\hat{t}_i^{\text{unload}})$$

where $\hat{t}_i^{\text{unload}}$ is the predicted unloading time of DCV_i depending on the routes of the first $k+N$ bags that entered the network, and $\mathbf{t}(k) = [t_1^{\text{load}} \ \ldots \ t_{k+N}^{\text{load}} \ t_1^{\text{unload}} \ \ldots \ t_{k+N}^{\text{unload}}]^\top$.

Now let $\mathbf{r}(k)$ denote the future route sequence for the next N bags entering the network at bag step k, $\mathbf{r}(k) = [r(k+1)\ r(k+2)\ \ldots\ r(k+N)]^\top$. Accordingly, the MPC optimization problem at bag step k is defined as follows:

$$\begin{aligned}
&\min_{\mathbf{r}(k)} J_{k,N}^{\text{Centr_MPC}}(\mathbf{t}(k)) \\
&\text{subject to} \\
&\qquad \mathbf{t}(k) = \mathscr{M}^{\text{route_ctrl}}\big(\mathscr{T}, \mathbf{x}(t_k^{\text{load}}), \mathbf{r}(k)\big) \\
&\qquad \mathscr{C}(\mathbf{t}(k)) \leq 0
\end{aligned}$$

When using centralized MPC, at each bag step k, the future route sequence $\mathbf{r}(k)$ is computed over an horizon of N bags so that the objective function is minimized subject to the dynamics of the system and the operational constraints.

Next we show how we compute for this control method the durations $\tau_s^{\text{sw\_in}}$ and $\tau_s^{\text{sw\_out}}$ with $s \in \{1,2,\ldots,S\}$. Assume that the switch-in at S_s is positioned on link $l \in \{0,1\}$. Then let z_s denote the bag closest to S_s and traveling at time instant t^{crt} on the incoming link $1-l$ of S_s. Also let τ_s^{arrival} be the time period that the DCV transporting bag z_s needs to travel (at maximum speed) the distance between the current position of bag z_s and S_s. Then the duration $\tau_s^{\text{sw\_in}}$ is given by:

$$\tau_s^{\text{sw\_in}} = \begin{cases} \max\left(\tau^{\text{switch}} - \tau_s^{\text{sw\_in\_prev}}, \tau_s^{\text{arrival}}\right) & \text{if at } t^{\text{crt}} \text{ there is a DCV on the incoming link } 1-l \text{ of } \mathrm{S}_s \\ \infty & \text{otherwise} \end{cases}$$

where $\tau_s^{\text{sw\_in\_prev}}$ is the time period for which the switch-in at junction S_s has been in its current position. Hence, the bag closest to junction S_s is allowed to pass first if the switch is positioned on the appropriate incoming link, or if a toggle is possible (due to constraint $\mathbf{C}_3$).

The duration $\tau_s^{\text{sw\_out}}$ is given by:

$$\tau_s^{\text{sw\_out}} = \begin{cases} \max\left(0, \tau^{\text{switch}} - \tau_s^{\text{sw\_out\_prev}}\right) & \text{if at } t^{\text{crt}} \text{ there is a DCV at } \mathrm{S}_s \text{ and its route asks for toggle} \\ \infty & \text{otherwise} \end{cases}$$

where $\tau_s^{\text{sw\_out\_prev}}$ is the time period for which the switch-out at junction S_s has been in its current position.

Centralized MPC can compute on-line the route of each DCV in the network, but it requires large computational efforts as will be illustrated in Section 4.5. Therefore, we also propose decentralized and distributed control approaches, which offer a trade-off between the optimality of the performance for the controlled system and the time required to compute the solution.

4.4.3 Decentralized MPC

In decentralized model predictive route choice control we consider each junction separately, as a local system. For all junctions we will then define similar local MPC problems. No communication and no coordination is involved between the local controllers.

Local system

Each local system consists of a junction, its incoming links, and its outgoing links. Let us now consider the most complex case, where junction S_s with $s \in \{1,2,\ldots,S\}$ has both a switch-in and a switch-out. Moreover, S_s is not directly connected to an unloading station. Then we first index[2] the bags that successively cross junction S_s during the entire simulation period $[t_0, t_0+\tau^{\text{max\_sim}})$ as $b_{s,1}, b_{s,2}, \ldots, b_{s,N_s^{\text{bags}}}$, where N_s^{bags} is the number of bags that cross S_s during the simulation period.

[2] This order depends on the evolution of the position of the switch-in at junction S_s.

Local control measures

In decentralized route choice control we compute for each junction S_s the positions of the switch-in and switch-out of junction S_s for each bag that crosses the junction S_s.

Now consider junction S_s. Recall from Section 4.4.2 that we use a variant of MPC with a bag index. So, in this approach, the local control is updated at every time instant when some bag has just entered an incoming link of junction S_s. Let t_s^{crt} be such a time instant. Then we determine bag index k such that $t_{s,k}^{\mathrm{cross}} \leq t_s^{\mathrm{crt}} < t_{s,k+1}^{\mathrm{cross}}$, where $t_{s,k}^{\mathrm{cross}}$ is defined as the time instant when bag $b_{s,k}$ has just crossed the junction. If no bag has crossed the junction yet, we set $k = 0$.

Let $N^{\max}$ be the maximum prediction horizon for a local MPC problem and $n_{s,l}^{\mathrm{horizon}}$ the number of DCVs traveling at time instant t^{crt} on link $l \in \{0,1\}$ going into S_s. Then, the local optimization is performed over the next $N_s = \min\left(N^{\max}, n_{s,0}^{\mathrm{horizon}} + n_{s,1}^{\mathrm{horizon}}\right)$ bags that will cross junction S_s after bag index k. By solving this local optimization problem we compute the control sequence

$$\mathbf{u}_s(k) = [u_s^{\mathrm{sw\_in}}(k+1) \ \ldots \ u_s^{\mathrm{sw\_in}}(k+N_s) \ u_s^{\mathrm{sw\_out}}(k+1) \ \ldots \ u_s^{\mathrm{sw\_out}}(k+N_s)]^\top$$

corresponding to the next N_s bags $b_{s,k+1}, b_{s,k+2}, \ldots, b_{s,k+N_s}$ that will cross the junction. The control variable $u_s^{\mathrm{sw\_in}}(k+j)$ with $j \in \{1,\ldots,N_s\}$ represents the position of the switch into S_s for the $k+j$th bag to cross S_s — $u_s^{\mathrm{sw\_in}}(k+j) = l$ with l the index of the incoming link on which the switch-in at S_s is positioned, $l \in \{0,1\}$. The control variable $u_s^{\mathrm{sw\_out}}(k+j)$ with $j \in \{1,\ldots,N_s\}$ represents the position of the switch out of S_s for the $k+j$th bag to cross S_s — $u_s^{\mathrm{sw\_out}}(k+j) = l$ with l the index of the outgoing link on which the switch-out at S_s is positioned, $l \in \{0,1\}$. So, the control decisions $u_s^{\mathrm{sw\_in}}(k+1)$, …, $u_s^{\mathrm{sw\_in}}(k+N_s)$ of the switch into S_s determine the order[3] in which the bags cross the junction and the time instants at which the bags $b_{s,k+1},\ldots,b_{s,k+N_s}$ enter S_s. The control decisions $u_s^{\mathrm{sw\_out}}(k+1)$, …, $u_s^{\mathrm{sw\_out}}(k+N_s)$ determine the next junction towards which the bags $b_{s,k+1},\ldots,b_{s,k+N_s}$ will travel.

Local objective function

When solving the local MPC optimization problem for junction S_s, we will use a local objective function $J_{s,k,N_s}^{\mathrm{Dec\_MPC}}$. The local objective function is computed via a simulation of the local system for the next N_s bags that will cross the junction, and is defined as follows:

$$J_{s,k,N_s}^{\mathrm{Dec\_MPC}}(\mathbf{t}_s(k)) = \sum_{j=1}^{\min(N_s,N_s^{\mathrm{cross}})} J_{k+j}(\hat{t}_{s,k+j}^{\mathrm{unload},*}) + \lambda^{\mathrm{pen}}(N_s - N_s^{\mathrm{cross}})$$

where

- N_s^{cross} is the number of DCVs that actually cross junction S_s during the prediction period,
- $\hat{t}_{s,k+j}^{\mathrm{unload},*}$ is the predicted unloading time instant of bag $b_{s,k+j}$,

[3]The order of DCVs is given by setting their speed in accordance with the control variables of the switch-in. For example if $u_s^{\mathrm{sw\_in}}(k+1) = 0$ and $u_s^{\mathrm{sw\_in}}(k+2) = 1$, but on the incoming link 0 there are more DCVs in a queue, then after allowing the first DCV traveling on incoming link 0 to enter S_s, the velocity of the other DCVs in the queue is set to 0 until the DCV coming from the incoming link 1 enters S_s and the switch-in is set back on link 0.

- λ^{pen} is a nonnegative weighting parameter,
- $\mathbf{t}_s(k) = [t^{\text{load}}_{s,k+1} \ \dots \ t^{\text{load}}_{s,k+N_s} \ \hat{t}^{\text{unload},*}_{s,k+1} \ \dots \ \hat{t}^{\text{unload},*}_{s,k+N_s}]^\top$ with $t^{\text{load}}_{s,k+j}$ the loading time for bags $b_{s,k+j}$.

The second term of the local objective function is included for the following reasoning. Assume that, at step k, there are no DCVs traveling on the incoming link $l \in \{0,1\}$ of junction S_s, while N DCVs travel on link $1-l$. If this term would not be considered, then $J^{\text{Dec\_MPC}}_{s,k,N_s}(\mathbf{t})$ would be minimum when the switch-in is positioned on link l during the prediction period. However, this is obviously not a good solution when the endpoints are open. Also note that N^{cross}_s and $\hat{t}^{\text{unload},*}_{s,k+j}$ are determined by simulating at time instant t^{crt} the prediction model presented next for a given control sequence $\mathbf{u}_s(k)$.

Local prediction model

The local prediction model at bag index k is an event-driven model for the local system over an horizon of N_s bags. So, according to **Algorithm 2**, for the next N_s bags to cross S_s, given the current state of the local system, we compute the period $\tau^{\min}_s$ until the next event will occur in the local system (loading if S_s is connected to loading stations, unloading if S_s is connected to unloading stations, switching at S_s, updating the speed of a DCV running through the local system), we shift the current time t^{crt}_s of the local prediction model at junction S_s with $\tau^{\min}_s$, take the appropriate action, and update the state of the local system.

First we show how we compute the durations $\tau^{\text{sw\_in}}_s$ and $\tau^{\text{sw\_out}}_s$ for the local prediction model at time instant t^{crt}_s. Note that we now compute $\tau^{\text{sw\_in}}_s$ and $\tau^{\text{sw\_out}}_s$ for each of the next N_s bags to cross S_s during the prediction period. Assume that the switch-in at junction S_s is positioned on link $l_{k+j} = u^{\text{sw\_in}}_s(k+j)$. Let $z_{s,k+j}$ denote the bag to cross S_s next, for $j = 1,2,\dots,N_s$. Also let $\tau^{\text{arrival}}_{s,k+j}$ be the time period that the DCV transporting bag $z_{s,k+j}$ needs to travel (at maximum speed) the distance between the current position of bag $z_{s,k+j}$ and S_s. Note that if at time instant t^{crt}_s with $t^{\text{cross}}_{s,k+j-1} \le t^{\text{crt}}_s < t^{\text{cross}}_{s,k+j}$, there is no bag on incoming link $1-u^{\text{sw\_in}}_s(k+j)$ of S_s, then $\tau^{\text{arrival}}_{s,k+j} = \infty$.

Then the duration $\tau^{\text{sw\_in}}_s$ is given by:

$$\tau^{\text{sw\_in}}_s = \begin{cases} \max\left(\tau^{\text{switch}} - \tau^{\text{sw\_in\_prev}}_s, \tau^{\text{arrival}}_{s,k+j}\right) & \text{if at time instant } t^{\text{crt}}_s \text{ the switch-in at } \mathrm{S}_s \text{ is not on link } l_{k+j} = u^{\text{sw\_in}}_s(k+j) \\ \infty & \text{otherwise} \end{cases}$$

Hence, now the order in which the DCVs cross junction S_s is given by the control inputs $u^{\text{sw\_in}}_s(k+1), \dots, u^{\text{sw\_in}}_s(k+N_s)$.

The duration $\tau^{\text{sw\_out}}_s$ is given by:

$$\tau^{\text{sw\_out}}_s = \begin{cases} \max\left(0, \tau^{\text{switch}} - \tau^{\text{sw\_out\_prev}}_s\right) & \text{if bag } b_{s,k+j} \text{ is at } \mathrm{S}_s \text{ and the switch-out at } \mathrm{S}_s \text{ is not positioned on link } u^{\text{sw\_out}}_s(k+j) \\ \infty & \text{otherwise} \end{cases}$$

The durations $\tau^{\text{sw\_in}}_s$ and $\tau^{\text{sw\_out}}_s$ for the model of **Algorithm 2** at time instant t^{crt} are computed similarly. But in this case, we do not compute any more these durations for N_s bags, but only for one bag, according to the control inputs $u^{\text{sw\_in}}_s(k+1)$ and $u^{\text{sw\_out}}_s(k+1)$.

Next, we present how we predict the unloading time instant for each of the next bags to cross S_s during the prediction period. To this aim, we consider a fixed release rate during

the prediction period for each outgoing link $l \in \{0,1\}$ of S_s. Let $\zeta_{s,l}$ be the fixed release rate at time instant t^{crt}. We now present how we calculate $\zeta_{s,l}$ given the state of the local system at t^{crt}. Let τ^{rate} be the length of the time window over which we compute the link release rate. The variable τ^{rate} can be derived using empirical data. If $t^{\mathrm{crt}} < t_0 + \tau^{\mathrm{rate}}$ we consider $\zeta_{s,l} = \zeta^{\max}$ with $\zeta^{\max}$ the maximum number of DCVs per time unit that can cross a junction using maximum speed. If $t^{\mathrm{crt}} \geq t_0 + \tau^{\mathrm{rate}}$, let $n_{s,l}^{\mathrm{rate}}$ denote the number of DCVs that left the outgoing link l within the time window $[t^{\mathrm{crt}} - \tau^{\mathrm{rate}}, t^{\mathrm{crt}})$. Then, if $n_{s,l}^{\mathrm{rate}} > 0$ the fixed release rate of link l out of S_s to be used during the entire prediction period is given by $\zeta_{s,l} = \dfrac{n_{s,l}^{\mathrm{rate}}}{\tau^{\mathrm{rate}}}$, while if $n_{s,l}^{\mathrm{rate}} = 0$ we set $\zeta_{s,l} = \varepsilon$ with $0 < \varepsilon \ll 1$. We do not set $\zeta_{s,l} = 0$ when $n_{s,l}^{\mathrm{rate}} = 0$ because later on this release rate will be used as denominator, and we want to avoid the division by zero, while obtaining consistent results.

Recall that we want to predict the arrival time of bag $b_{s,k+j}$ with $j \in \{1,\ldots,N_s\}$ at its end point. Let $\mathrm{S}_{s,l_{k+j}}^{\mathrm{next}}$ denote the junction that bag $b_{s,k+j}$ will cross next where $l_{k+j} = u_s^{\mathrm{sw\_out}}(k+j)$, and let $\mathrm{S}_{s,k+j}^{\mathrm{dest}}$ be the end point of bag $b_{s,k+j}$. Then, for each possible route $r \in \mathscr{R}_{s,k+j}^{\mathrm{next}}$, where $\mathscr{R}_{s,k+j}^{\mathrm{next}}$ is the set of routes from $\mathrm{S}_{s,l_{k+j}}^{\mathrm{next}}$ to $\mathrm{S}_{s,k+j}^{\mathrm{dest}}$, we predict the time when bag $b_{s,k+j}$ will arrive at $\mathrm{S}_{s,k+j}^{\mathrm{dest}}$ via route r as follows:

$$\hat{t}_{s,r,k+j}^{\mathrm{unload}} = t_{s,k+j}^{\mathrm{cross}} + \hat{\tau}_{s,k+j}^{\mathrm{link}} + \hat{\tau}_r^{\mathrm{route}} \tag{4.5}$$

where

- $t_{s,k+j}^{\mathrm{cross}}$ is the time instant (computed by the local prediction model) at which bag $b_{s,k+j}$ crosses S_s.
- $\hat{\tau}_{s,k+j}^{\mathrm{link}}$ is the time we predict[4] that bag $b_{s,k+j}$ spends on link l_{k+j} out of S_s. For this prediction we take:

$$\hat{\tau}_{s,k+j}^{\mathrm{link}} = \begin{cases} \max\left(\dfrac{d_{s,l_{k+j}}^{\mathrm{link}}}{v^{\max}}, \dfrac{1+n_{s,k+j}}{\zeta_{s,l_{k+j}}}\right) & \text{if link } l_{k+j} \text{ is not jammed} \\ \max\left(\dfrac{d_{s,l_{k+j}}^{\mathrm{link}}}{v^{\mathrm{jam}}}, \dfrac{1+n_{s,k+j}}{\zeta_{s,l_{k+j}}}\right) & \text{if link } l_{k+j} \text{ is jammed} \end{cases}$$

  where $d_{s,l_{k+j}}^{\mathrm{link}}$ is the length of link l_{k+j} out of S_s, $n_{s,k+j}$ is the number of DCVs on link l at time instant $t_{s,k+j}^{\mathrm{cross}}$, and v^{jam} is the speed to be used in case of jam, typically $v^{\mathrm{jam}} = 0.02\,\mathrm{m/s}$. We consider the outgoing link l_{k+j} of S_s to be jammed only if $Q_{s,l_{k+j}} \geq \alpha Q_{s,l_{k+j}}^{\max}$ where $Q_{s,l_{k+j}}$ is the capacity link l_{k+j} at time instant $t_{s,k+j}^{\mathrm{cross}}$, $Q_{s,l_{k+j}}^{\max}$ is its maximum capacity, and α is a weighting parameter determined based on empirical data (typically $\alpha = 0.8$).
- $\hat{\tau}_r^{\mathrm{route}}$ is the predicted travel time on route $r \in \mathscr{R}_{s,k+j}^{\mathrm{next}}$ for an average speed determined based on empirical data.

[4] If $\mathrm{S}_{s,l_{k+j}}^{\mathrm{next}}$ is an unloading station and $\mathrm{S}_{s,l_{k+j}}^{\mathrm{next}}$ is not $\mathrm{S}_{s,k+j}^{\mathrm{dest}}$ then $\hat{\tau}_{s,k+j}^{\mathrm{link}} = \tau^{\max}$ with $\tau^{\max}$ a large nonnegative scalar.

Then the optimal predicted unloading time instant is defined as follows:

$$\hat{t}_{s,k+j}^{\text{unload},*} = \underset{\{\hat{t}_{s,r,k+j}^{\text{unload}} | r \in \mathscr{R}_{s,k+j}^{\text{next}}\}}{\arg\min} \; J_{k+j}(\hat{t}_{s,r,k+j}^{\text{unload}})$$

Local optimization problem

So, the MPC optimization problem at junction S_s and bag index k is defined as follows:

$$\begin{aligned}
&\min_{\mathbf{u}_s(k)} J_{s,k,N_s}^{\text{Dec_MPC}}(\mathbf{t}(k)) \\
&\text{subject to} \\
&\qquad \mathbf{t}(k) = \mathscr{M}^{\text{local,switch_ctrl}}\big(\mathscr{T}, \mathbf{x}_s(t_{s,k}^{\text{cross}}), \mathbf{u}_s(k)\big) \\
&\qquad \mathscr{C}(\mathbf{t}(k)) \leq 0
\end{aligned}$$

where $\mathscr{M}^{\text{local,switch\_ctrl}}\big(\mathscr{T}, \mathbf{x}_s(t_{s,k}^{\text{cross}}), \mathbf{u}_s(k)\big)$ describes the local dynamics of junction S_s with its incoming and outgoing links, with $\mathbf{x}_s(t_{s,k}^{\text{cross}})$ the state of the local system at time instant $t_{s,k}^{\text{cross}}$.

After computing the optimal control, only $u_s^{\text{sw\_in}}(k+1)$ and $u_s^{\text{sw\_out}}(k+1)$ are applied. Next the state of the system is updated. At bag step $k+1$, a new optimization will be then solved over the next N_s bags.

The main advantage of decentralized MPC consists in a smaller computation time than the one needed when using centralized control due to the fact that we now compute for each junction, independently, the solution of a smaller and simplified optimization problem. However, using decentralized MPC to compute the DCVs' route choice also yields a decrease in the overall performance of the DCV-based baggage handling system.

4.4.4 Distributed MPC

One can increase the performance of the *decentralized* control approach proposed above by implementing a *distributed* approach that uses additional communication between neighboring junctions.

Levels of influence

In distributed model predictive route choice control we consider local subsystems, each consisting of a junction S_s with $s \in \{1, 2, \ldots, S\}$, its incoming and its outgoing links. But in contrast to decentralized MPC, data will be now communicated between neighboring junctions which are characterized by the concept of level of influence. The levels of influence are defined as follows.

Let us first assign one or more levels of *downstream* influence to each junction in the network. We assign downstream influence level 1 to each junction in the network connected via a link to a loading station. Next, we consider all junctions connected to some junction with influence level 1 via an outgoing link, and we assign influence level 2 to them. In this way we recursively assign an influence level to each junction with the constraint that at most $\kappa_{\text{d}}^{\max}$ downstream influence levels are assigned to a given junction[5]. For example see

[5]The constraint that at most $\kappa_{\text{d}}^{\max}$ downstream influence levels are assigned to a junction limits the computational complexity and keeps all levels of influence finite.

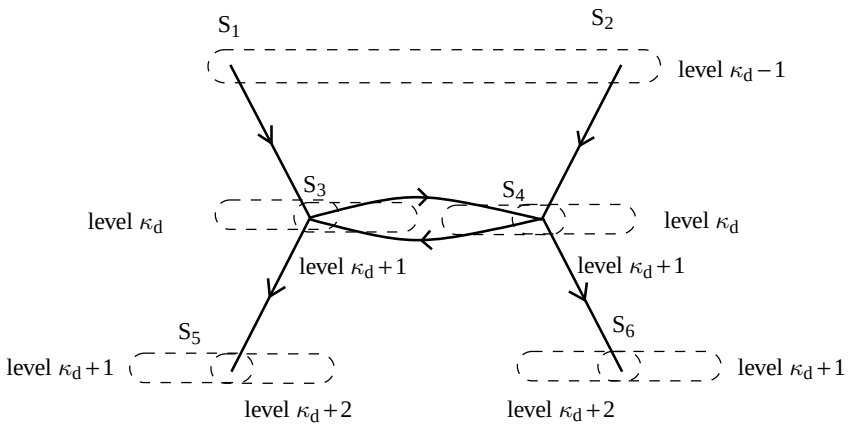


Figure 4.6: Levels of downstream influence for parallel computation.

Figure 4.6 where we define maximum 2 levels of downstream influence for each junction in the network. For this example we have considered the junctions S_1 and S_2 to have assigned downstream influence level $\kappa_d - 1$. Then S_3 and S_4 have been assigned level κ_d (since these junctions are connected to S_1 and S_2 via outgoing links). Next, we assign influence level $\kappa_d + 1$ to S_4, S_5, S_3, and S_6 (since they are connected to S_3 and S_4). Note that now S_3 and S_4 have 2 levels of downstream influence: κ_d and $\kappa_d + 1$. Therefore, S_5 and S_6 are also assigned influence level $\kappa_d + 2$ (since they are connected to S_3 and S_4 with influence level $\kappa_d + 1$).

Similarly we can also assign levels of *upstream* influence to each junction in the network. We assign upstream influence level 1 to each junction in the network connected via a link to an unloading station. Next, we assign upstream influence level 2 to all the junctions connected to some junction on upstream influence level 1 via its incoming links. Recursively, we then assign levels of upstream influence to each junction with the constraint that at most $\kappa_u^{\max}$ levels of upstream influence are assigned to a given junction.

Distributed MPC with a single round of downstream communication

Let us now consider distributed MPC with a single round of downstream communication. This means that first the local controller of each junction with influence level $\kappa_d = 1$ solves the local optimal control problem of Section 4.4.3.

After computing the optimal switch control sequence, each junction with influence level $\kappa_d = 1$ communicates to its neighboring junctions at level $\kappa_d + 1 = 2$ which bags (out of all the bags over which we make the prediction for the corresponding junction with influence level κ_d) will enter the incoming link of the junction at level $\kappa_d + 1$ and at which time instant. Next, we iteratively consider the junctions at levels $\kappa_d = 2, 3, \ldots, K^{\text{downstream}}$, were $K^{\text{downstream}}$ is the largest level of downstream influence assigned in the network. Then, for each junction with influence level $\kappa_d > 1$, we compute a local solution to the local MPC problem as presented next.

Assume S_s with $s \in \{1, \ldots, S\}$ has influence level $\kappa_d > 1$. Let $S_{s,l}^{\text{prev}}$ denote the neighboring junction of S_s connected via the incoming link $l \in \{0, 1\}$ of S_s (so, $S_{s,l}^{\text{prev}}$ has influence

level $\kappa_{\mathrm{d}}-1$). Then, we compute a local solution for S_s to the local MPC problem defined below over an horizon of

$$N_s = \min\Big(N^{\max}, \sum_{l=0}^{1}\big(n_{s,l}^{\text{horizon}} + n_{s,l,0}^{\text{pred\_cross}} + n_{s,l,1}^{\text{pred\_cross}}\big)\Big) \tag{4.6}$$

bags where $N^{\max}$ is the maximum prediction horizon for the local MPC problem, $n_{s,l}^{\text{horizon}}$ is the number of DCVs traveling at time instant t^{crt} on link $l \in \{0,1\}$ going into S_s, and $n_{s,l,m}^{\text{pred\_cross}}$ is the number of DCVs traveling towards $\mathrm{S}_{s,l}^{\text{prev}}$ on its incoming link m that we predict (while solving the local optimization problem at $\mathrm{S}_{s,l}^{\text{prev}}$) to cross $\mathrm{S}_{s,l}^{\text{prev}}$ and continue their journey towards S_s.

The MPC optimization problem at junction S_s and bag index k is defined as follows:

$$\begin{aligned}
&\min_{\mathbf{u}_s(k)} J_{s,k,N_s}^{\text{Distr_MPC}}(\mathbf{t}(k))\\
&\text{subject to}\\
&\qquad \mathbf{t}(k) = \mathscr{M}^{\text{local,switch_ctrl}}\big(\mathscr{T}, \mathbf{x}_s(t_{s,k}^{\text{cross}}), \mathbf{u}_s(k)\big)\\
&\qquad \mathscr{C}(\mathbf{t}(k)) \le 0
\end{aligned}$$

with N_s given by (4.6). Note that in this approach $\mathscr{M}^{\text{local,switch\_ctrl}}\big(\mathscr{T}, \mathbf{x}_s(t_{s,k}^{\text{cross}}), \mathbf{u}_s(k)\big)$ describes the local dynamics of junction S_s with its incoming and outgoing links and additional data from neighboring junctions (if any).

After computing the optimal control, only $u_s^{\text{sw\_in}}(k+1)$ and $u_s^{\text{sw\_out}}(k+1)$ are applied. Next the state of the system is updated. At bag step $k+1$, a new optimization will be then solved over the next N_s bags.

The computation of the local control is performed according to the following algorithm.

Algorithm 3. Distributed computation of local control with a single iteration of downstream communication

1: **for** $\kappa_{\mathrm{d}} = 1$ to $K^{\text{downstream}}$ **do**
2: compute independently local switching sequences for influence level κ_{d} taking into account the control on influence level $\kappa_{\mathrm{d}}-1$
3: **end for**

Every time some bag has crossed some junction we update the local control of junctions in the network as follows. Assume that some bag has just crossed junction S_s which has assigned level κ_{d}. Then, we update the control as follows. We consider a subtree rooted at S_s and consisting of nodes of subsequent levels of influence that are connected via a link. So, only the control of the switch-in and switch-out of the junctions in this subtree have to be updated.

Note that the controllers of the junctions on level κ_{d} have to wait for the completion of the computation of the switching sequences of the controllers on the previous level before they can start to compute their future control action. Therefore, when comparing with decentralized MPC, such distributed MPC may improve the performance of the system, but at the cost of higher computation time due to the required synchronization in computing the control actions.

Distributed MPC with a single round of downstream and upstream communication

In order to further improve the performance of the distributed control approach presented above, we now add an extra round of communication and consider distributed MPC with a round of downstream and upstream communication. This method involves the following steps:

- Every time a bag has crossed a junction we compute the local control sequences according to the downstream levels of influence as explained above.
- Next, for the junctions on level 1 of upstream influence we update the release rate of their incoming links as follows. We take as example junction S_s with $\kappa_{\mathrm{u}} = 1$. For all other junctions we will apply the same procedure. We virtually apply at S_s the optimal control sequence $\mathbf{u}_s^*$ that we have computed when optimizing downstream. Let $t_s^{\mathrm{last},*}$ be the time instant at which the last bag crossed S_s (out of all the bags over which we make the prediction for S_s). If $t_s^{\mathrm{last},*} < t_0 + \tau^{\mathrm{rate}}$ we set $\zeta_{s,l} = \zeta^{\max}$ for $l = 0, 1$. Otherwise, if $n_{s,l}^{\mathrm{rate}} > 0$ with $n_{s,l}^{\mathrm{rate}}$ the number of DCVs that left the outgoing link l of S_s within the time window $[t_s^{\mathrm{last},*} - \tau^{\mathrm{rate}}, t_s^{\mathrm{last},*})$, we set $\zeta_{s,l} = \dfrac{n_{s,l}^{\mathrm{rate}}}{\tau^{\mathrm{rate}}}$. Finally, if $n_{s,l}^{\mathrm{rate}} = 0$ we set $\zeta_{s,l} = \varepsilon$ with $0 < \varepsilon \ll 1$. Now we solve the local MPC problem presented on page 70 using the updated release rates and we compute the local control of all junctions at upstream level $\kappa_{\mathrm{u}} + 1$. Recursively, we compute the local control until level K^{upstream} where K^{upstream} is the largest level of upstream influence assigned in the network.

These steps are summarized in **Algorithm 4**.

Algorithm 4. Distributed computation of local control with a single round of downstream and upstream coordination

1: **for** $\kappa_{\mathrm{d}} = 1$ to $K^{\mathrm{downstream}}$ **do**
2: compute independently local switching sequences for influence level κ_{d} taking into account the local control on downstream influence level $\kappa_{\mathrm{d}} - 1$
3: **end for**
4: **for** $\kappa_{\mathrm{u}} = 1$ to K^{upstream} **do**
5: compute independently local switching sequences for influence level κ_{u} taking into account the local control on upstream influence level $\kappa_{\mathrm{u}} - 1$ and the updated release rate
6: **end for**

By also performing the upstream round of communication, more information about the future congestion is provided via the updated release rate. This information might change the initial intended control actions of each junction. Typically (if one allows sufficient time to compute the solution of each local optimization problem), this new variant of distributed MPC increases the performance of the system, but also the computational effort increases since we deal with one more round of optimizations.

In future work we will farther improve the performance of the system by considering multiple up and down rounds of optimizations and by extending the range of communication exchange to more than one level. Moreover, we will also extend the local control area to more than one node and assess the efficiency of such distributed approaches.

4.4.5 MPC with mixed-integer linear programming

We now present an alternative approach for reducing the complexity of the computations. In this approach we simplify and approximate the nonlinear route choice optimization problem by a mixed integer linear programming (MILP) problem. The advantage is that for MILP optimization problems solvers are available, see e.g. [27], that allow us to efficiently compute the global optimal solution. The solution of the MILP problem can then be used as a good initial starting point for the original nonlinear optimization problem of centralized MPC. In future work we will also consider this approach to compute a good initial starting point for the nonlinear optimization problem of decentralized and distributed MPC.

Mixed integer linear programming

Mixed integer linear programming (MILP) problems are optimization problems with a linear objective function, subject to linear equality and inequality constraints. The general formulation for a mixed-integer linear programming problem is the following:

$$\begin{aligned}
&\min_{\mathbf{x}^{\mathrm{MILP}}} \mathbf{c}^\top \mathbf{x}^{\mathrm{MILP}} \\
&\text{subject to} \\
&\quad \mathbf{A}^{\mathrm{eq}}\mathbf{x}^{\mathrm{MILP}} = \mathbf{b}^{\mathrm{eq}} \\
&\quad \mathbf{A}\mathbf{x}^{\mathrm{MILP}} \leq \mathbf{b} \\
&\quad \mathbf{x}^{\mathrm{low}} \leq \mathbf{x}^{\mathrm{MILP}} \leq \mathbf{x}^{\mathrm{up}}
\end{aligned}$$

where $\mathbf{c}$, $\mathbf{x}^{\mathrm{MILP}}$, $\mathbf{x}^{\mathrm{low}}$, $\mathbf{x}^{\mathrm{up}}$, $\mathbf{b}^{\mathrm{eq}}$, and $\mathbf{b}$ are vectors, with $\mathbf{x}^{\mathrm{low}}$ the lower bound of $\mathbf{x}^{\mathrm{MILP}}$ and $\mathbf{x}^{\mathrm{up}}$ its upper bound, and where $\mathbf{A}^{\mathrm{eq}}$ and $\mathbf{A}$ are matrices (all these vectors and matrices have appropriate size). Note that MILP solvers compute solutions $\mathbf{x}^{\mathrm{MILP}}$ for the problem above, where some of the elements of $\mathbf{x}^{\mathrm{MILP}}$ are restricted to integer values.

In order to transform the original nonlinear route choice model of a DCV-based baggage handling system into an MILP model we will use two equivalences, see, e.g., [7], where f is a function defined on a bounded set X with upper and lower bounds b^{up} and b^{low} for the function values, δ is a binary variable, y is a real-valued scalar variable, and ϵ is a small tolerance[6] (typically the machine precision):

P1: $[f(x) \leqslant 0] \iff [\delta = 1]$ is true if and only if

$$\begin{cases} f(x) \leqslant b^{\mathrm{up}}(1-\delta) \\ f(x) \geqslant \epsilon + (b^{\mathrm{low}} - \epsilon)\delta \end{cases},$$

P2: $y = \delta f(x)$ is equivalent to

$$\begin{cases} y \leqslant b^{\mathrm{up}}\delta \\ y \geqslant b^{\mathrm{low}}\delta \\ y \leqslant f(x) - b^{\mathrm{low}}(1-\delta) \\ y \geqslant f(x) - b^{\mathrm{up}}(1-\delta) \end{cases}.$$

[6] The tolerance ϵ is needed to transform a constraint of the form $y > 0$ into $y \geq 0$, since in MILP problems only nonstrict inequalities are allowed.

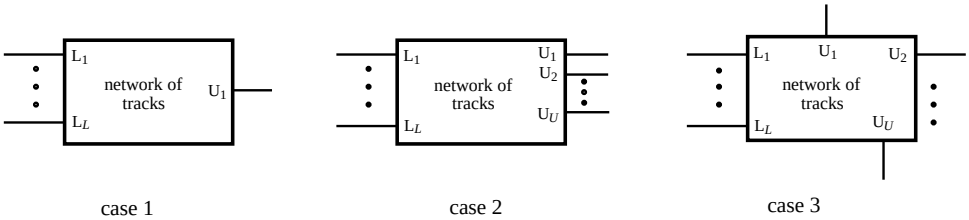


Figure 4.7: Three cases with a gradually increasing complexity: network with one unloading station, more unloading stations close together, more unloading station far apart.

Simplified route choice models

Now we present simplified route choice models that can be written as MILP models. We consider three cases with a gradually increasing complexity where the DCV-based baggage handling system has only one unloading station, more unloading stations close together, and more unloading stations far apart, as illustrated in Figure 4.7. We consider these cases since they grow in complexity and, for each of these cases, additional assumptions have to be made in order to obtain a simplified route choice model that can be recast as an MILP model. Note that these route choice models will not be event-based models, but a discrete-time models.

Common assumptions for all three cases

In order to transform the route choice problem into an MILP problem, we first simplify it by assuming the following:

$\mathbf{A}_8$: The DCVs run with maximum speed along the track segment and, if necessary, they wait at the end of the link in a vertical queue. In principle, the queue lengths should be integers as their unit is "number of DCVs", but we will approximate them using reals.

$\mathbf{A}_9$: The dynamic demand D_i of loading station L_i, $i \in \{1,\ldots,L\}$, where L is the number of loading stations, is approximated with a piecewise constant demand. The piecewise constant demand D_i has level changes occurring only at integer multiples of τ_s with τ_s the sampling time. This is necessary in order to easily combine the time when a bag reaches a queue at a junction with the time when the demand changes. Let the time instant t_k be defined as $t_k = t_0 + k\tau_\mathrm{s}$ with t_0 the initial simulation time which is assumed to be integer multiple of τ_s, and $k \in \mathbb{N}$ with $\mathbb{N}$ the set of natural numbers. Then, during the time interval $[t_k, t_{k+1})$, the demand at loading station L_i is $D_i(k)$.

$\mathbf{A}_{10}$: For each link a free-flow travel time is assigned. This free-flow travel time represents the time period that a DCV requires to travel on a link in case of no congestion, using, hence, maximum speed. The free-flow travel time of a link is always a multiple of τ_s.

Note that the assumptions $\mathbf{A}_1$-$\mathbf{A}_7$ that we have considered when writing the event-based model of the baggage handling systems (see Section 4.2.1) hold for all the cases to be considered next.

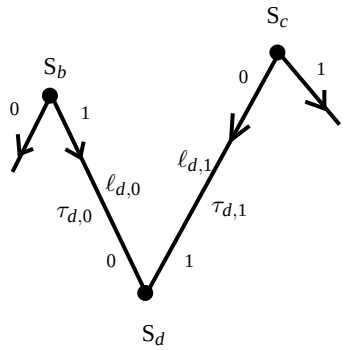


Figure 4.8: Network elements.

Case 1: one unloading station

We now consider the case of a DCV-based baggage handling system with only one unloading station.

Model The control time step for each junction in the network is τ_s. Then at time step k with $k \in \mathbb{N}$ and for each junction S_s with $s \in \{1,2,\ldots,S\}$, we compute the control actions $u_s^{\text{sw\_in}}(k)$ and $u_s^{\text{sw\_out}}(k)$, where $u_s^{\text{sw\_in}}(k)$ expresses the position of the switch-in at junction S_s during the time period $[t_k, t_{k+1})$ (if S_s has two incoming links) and $u_s^{\text{sw\_out}}(k)$ that expresses the position of the switch-out at junction S_s during the time period $[t_k, t_{k+1})$ (if S_s has two outgoing links).

In order to illustrate the derivation of the route choice model let us now consider the most complex cell a network can contain, as depicted in Figure 4.8 where junction S_d has 2 neighboring junctions S_b and S_c connected to it via its incoming links, and both junctions S_b and S_c have 2 outgoing links.

Next we present how the evolution of the queue length at the end of each incoming link of S_d is determined. At time step k, $u_b^{\text{sw\_out}}(k)$ and $u_c^{\text{sw\_out}}(k)$ are computed for junctions S_b and S_c, and $u_d^{\text{sw\_in}}(k)$ for junction S_d. Let $\ell_{s,l}$ denote the link between a junction S_s and its upstream neighbor connected to it via the incoming link l as illustrated in Figure 4.8. Also, let $q_{s,l}(k)$ denote the length of the queue at the end of link $\ell_{s,l}$ at time instant t_k. Recall that each link in the network has been assigned a given free-flow travel time (assumption $\mathbf{A}_{10}$). Then, let $\tau_{d,0}$ denote the free-flow travel time of link $\ell_{d,0}$ and let $\tau_{d,1}$ denote the free-flow travel time of link $\ell_{d,1}$. Hence, the control signals $u_b^{\text{sw\_out}}(k)$ and $u_c^{\text{sw\_out}}(k)$ influence $q_{d,0}$ and $q_{d,1}$ after $\frac{\tau_{d,0}}{\tau_s}$ and $\frac{\tau_{d,1}}{\tau_s}$ time steps[7] respectively.

The evolution of the length of the queue at the end of link $\ell_{d,l}$, is given by:

$$q_{d,l}(k+1) = \max\left(0, q_{d,l}(k) + \left(I_{d,l}\left(k - \frac{\tau_{d,l}}{\tau_s}\right) - O_{d,l}^{\max}(k)\right)\tau_s\right) \tag{4.7}$$

where

[7] Recall that, according to assumption $\mathbf{A}_{10}$, $\tau_{s,l}$ is an integer multiple of τ_s.

- $q_{d,l}(k+1)$ is the length of the queue at the end of link $\ell_{d,l}$ at time instant t_{k+1}.
- $I_{d,l}(k)$ represents the inflow[8] of link $\ell_{d,l}$ during the period $[t_k, t_{k+1})$. Note that if a junction S_s with $s \in \{1,2,\ldots,S\}$ is directly connected to a loading station L_i with $i \in \{1,2,\ldots,L\}$ via the incoming link $l \in \{0,1\}$ of S_s, then

$$I_{s,l}(k) = D_i(k).$$

- $O^{\max}_{d,l}(k)$ is the maximum number of DCVs per time unit that cross S_d during $[t_k, t_{k+1})$ after traveling on link $\ell_{d,l}$.

Note that we also have the constraint that the length of the queue at the end of link $\ell_{d,l}$ has an upper bound:

$$\sum_{d \in \mathscr{D}} q_{d,l}(k+1) \leq q^{\max}_{d,l} \tag{4.8}$$

where $q^{\max}_{d,l}$ expresses the maximum number of DCVs that link $\ell_{d,l}$ can accommodate. The variable $q^{\max}_{d,l}$ is defined as $q^{\max}_{d,l} = \lfloor \frac{d_{s,l}}{d^{\min}} \rfloor$ where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x, $d_{s,l}$ is the length of the link $\ell_{s,l}$ with $l \in \{0,1\}$, and $d^{\min}$ consists of the minimum safe distance between DCVs and the length of a DCV.

Furthermore, the maximum number of DCVs per time unit that wait in the queue or arrive at the end of link $\ell_{d,l}$, and that cross S_d during $[t_k, t_{k+1})$ is defined as follows:

$$O^{\max}_{d,0}(k) = (1 - u^{\text{sw\_in}}_d(k))O^{\max} \tag{4.9}$$

$$O^{\max}_{d,1}(k) = u^{\text{sw\_in}}_d(k)O^{\max} \tag{4.10}$$

where $O^{\max}$ is the maximum outflow[9] of a junction. Note that we have used the operator max in (4.7) since the length of the queue is always larger than or equal to 0.

The inflows $I_{d,0}(k)$ and $I_{d,1}(k)$ are given by:

$$I_{d,0}(k) = u^{\text{sw\_out}}_b(k)O_b(k) \tag{4.11}$$

$$I_{d,1}(k) = (1 - u^{\text{sw\_out}}_c(k))O_c(k) \tag{4.12}$$

with $O_b(k)$ and $O_c(k)$ respectively the outflow of junction S_b and S_c during the time interval $[t_k, t_{k+1})$.

The outflow $O_s(k)$ of a junction S_s with $s \in \{1,2,\ldots S\}$ can be defined as follows (we consider two cases):

- S_s has one incoming link. Then

$$O_s(k) = \min\left(O^{\max}, \left(\frac{q_{s,0}(k)}{\tau_{\mathrm{s}}} + I_{s,0}\left(k - \frac{\tau_{s,0}}{\tau_{\mathrm{s}}}\right)\right)\right) \tag{4.13}$$

- S_s has two incoming links. Then

$$O_s(k) = \min\Bigg(O^{\max}, \left(1 - u^{\text{sw\_in}}_s(k)\right)\left(\frac{q_{s,0}(k)}{\tau_{\mathrm{s}}} + I_{s,0}\left(k - \frac{\tau_{s,0}}{\tau_{\mathrm{s}}}\right)\right) + u^{\text{sw\_in}}_s(k)\left(\frac{q_{s,1}(k)}{\tau_{\mathrm{s}}} + I_{s,1}\left(k - \frac{\tau_{s,1}}{\tau_{\mathrm{s}}}\right)\right)\Bigg) \tag{4.14}$$

[8]The inflow of a link equals the number of DCVs that entered that link per time unit.

[9]The outflow of a junction is defined as the number of DCVs that cross that junction per time unit.

Next, assume junction S_e with $e \in \{1,2,\dots S\}$ to be directly connected to the unloading station U_1. Then the outflow $O_e(k)$ of junction S_e during the period $[t_k, t_{k+1})$ is derived as presented above. Furthermore, let $U_1(k)$ denote the outflow of unloading station U_1 during $[t_k, t_{k+1})$, and let τ_1 be the free-flow travel time between S_e and U_1. In order to derive the outflow $U_1(k)$ we distinguish two cases:

- S_e has only one outgoing link. Then

$$U_1(k) = O_e\big(k - \frac{\tau_1}{\tau_\mathrm{s}}\big).$$

- S_e has two outgoing links. Without loss of generality we assume that the unloading station is link 0 out of S_e. Then

$$U_1(k) = \Big(1 - u_e^{\text{sw\_out}}\big(k - \frac{\tau_1}{\tau_\mathrm{s}}\big)\Big) O_e\big(k - \frac{\tau_1}{\tau_\mathrm{s}}\big)$$

  where $u_e^{\text{sw\_out}}(k)$ expresses the position of the switch out of S_e during the time interval $[t_k, t_{k+1})$.

MILP model We now use the MILP properties **P1** and **P2** presented in this section in order to obtain an MILP model for the route choice model given by equations (4.7)–(4.14). Note that depending on the order in which properties **P1** and **P2** are applied and in which additional auxiliary variables are introduced, we may end up with more or less binary and real-valued variables in the final MILP problem. The number of binary variables — and to a lesser extent the number of real variables — should be kept as small as possible since this number has a direct impact on the computational complexity of the final MILP problem.

We start by transforming (4.14) using Property **P1**. Let the real-valued variable $f_s^{\text{out}}(k)$ be equal to

$$f_s^{\text{out}}(k) = \big(1 - u_s^{\text{sw\_in}}(k)\big)\Big(\frac{q_{s,0}(k)}{\tau_\mathrm{s}} + I_{s,0}\big(k - \frac{\tau_{s,0}}{\tau_\mathrm{s}}\big)\Big) + u_s^{\text{sw\_in}}(k)\Big(\frac{q_{s,1}(k)}{\tau_\mathrm{s}} + I_{s,1}\big(k - \frac{\tau_{s,1}}{\tau_\mathrm{s}}\big)\Big) \quad (4.15)$$

Now, we introduce the binary variable $\delta_s^{\text{out}}(k)$ that equals 1 if and only if $O^{\max} \leq f_s^{\text{out}}(k)$. Then we rewrite (4.14) as follows:

$$O_s(k) = \delta_s^{\text{out}}(k) O^{\max} + \big(1 - \delta_s^{\text{out}}(k)\big) f_s^{\text{out}}(k) \quad (4.16)$$

where the condition $\delta_s^{\text{out}}(k) = 1$ if and only if $O^{\max} - f_s^{\text{out}}(k) \leq 0$ is equivalent to (cf. Property **P1**):

$$\begin{cases} O^{\max} - f_s^{\text{out}}(k) \leq b^{\text{up}}\big(1 - \delta_s^{\text{out}}(k)\big) \\ O^{\max} - f_s^{\text{out}}(k) \geq \epsilon + (b^{\text{low}} - \epsilon)\delta_s^{\text{out}}(k) \end{cases}$$

with $b^{\text{up}} = O^{\max}$ and $b^{\text{low}} = -\frac{1}{\tau_\mathrm{s}} q^{\max}$ where $q^{\max} = q_{s,0}^{\max} + q_{s,1}^{\max}$.

However, (4.16) is not yet linear. So, we use Property **P2** and introduce the real-valued scalar variables $y_s^{\text{out}}(k)$ such that:

$$y_s^{\text{out}}(k) = \delta_s^{\text{out}}(k) f_s^{\text{out}}(k)$$

or equivalently:

$$\begin{cases} y_s^{\text{out}}(k) \leqslant b^{\text{up}}\delta_s^{\text{out}}(k) \\ y_s^{\text{out}}(k) \geqslant 0 \\ y_s^{\text{out}}(k) \leqslant f_s^{\text{out}}(k) \\ y_s^{\text{out}}(k) \geqslant f_s^{\text{out}}(k) - b^{\text{up}}(1-\delta_s^{\text{out}}(k)) \end{cases} .$$

Hence, one obtains:

$$O_s(k) = O^{\max}\delta_s^{\text{out}}(k) + f_s^{\text{out}}(k) - y_s^{\text{out}}(k)$$

which is linear. Note that (4.15) can be written as a linear expression by introducing the additional variables $y_{\text{q},s,l}^{\text{in}}(k) = u_s^{\text{sw\_in}}(k)q_{s,l}(k)$ and $y_{\text{I},s,l}^{\text{in}}(k) = u_s^{\text{sw\_in}}(k)I_{s,l}\left(k - \frac{\tau_{s,l}}{\tau_{\text{s}}}\right)$ and the corresponding system of linear inequalities corresponding to Property **P2** for $f(x) = q_{s,l}(k)$ with $b^{\text{up}} = q^{\max}$, and $b^{\text{low}} = 0$, and $f(x) = I_{s,l}\left(k - \frac{\tau_{s,l}}{\tau_{\text{s}}}\right)$ with $b^{\text{up}} = O^{\max}$, and $b^{\text{low}} = 0$ respectively.

Similarly we can write the MILP equivalent for (4.13). Finally, we transform (4.7) into its MILP equivalent. Let the real-valued variable $f_{d,l}(k)$ be equal to $q_{d,l}(k) + \left(I_{d,l}\left(k - \frac{\tau_{d,l}}{\tau_{\text{s}}}\right) - O_{d,l}^{\max}(k)\right)\tau_{\text{s}}$. Additionally we also introduce the binary variable $\delta_{d,l}(k)$ that equals 1 if and only if $f_{d,l}(k) \leq 0$ and we rewrite (4.7) as:

$$q_{d,l}(k+1) = \left(1 - \delta_{d,l}(k)\right) f_{d,l}(k)) \tag{4.17}$$

together with the system of linear inequalities corresponding to Property **P1** with $b^{\text{up}} = q^{\max} + O^{\max}\tau_{\text{s}}$ and $b^{\text{low}} = -O^{\max}\tau_{\text{s}}$.

But (4.17) is not yet linear. Therefore, we introduce the variable $y_{d,l}(k) = \delta_{d,l}(k) f_{d,l}(k)$ and the system of linear inequalities corresponding to Property **P2** for $f(x) = f_{d,l}(k)$, with b^{up} and b^{low} as defined above, and we obtain:

$$q_{d,l}(k+1) = f_{d,l}(k) - y_{d,l}(k)$$

which is linear. Next we collect all the variables for the route choice model (i.e., inputs, control variables, and extra variables introduced by the MILP transformations) in a vector denoted by $\mathbf{x}^{\text{MILP}}(k)$ and all the partial queue lengths $q_{s,l}(k)$ in a vector denoted by $\mathbf{q}(k+1)$. Then the expressions derived above allow us to express $\mathbf{q}(k+1)$ as an affine function of $\mathbf{x}^{\text{MILP}}(k)$:

$$\mathbf{q}(k+1) = \mathbf{\Lambda}\mathbf{x}^{\text{MILP}}(k) + \boldsymbol{\gamma}$$

with a properly defined matrix $\mathbf{\Lambda}$ and vector $\boldsymbol{\gamma}$, where $\mathbf{x}^{\text{MILP}}(k)$ satisfies a system of linear equations and inequalities

$$\mathbf{A}^{\text{eq}}\mathbf{x}^{\text{MILP}}(k) = \mathbf{b}^{\text{eq}}$$
$$\mathbf{A}\mathbf{x}^{\text{MILP}}(k) \leq \mathbf{b},$$

which corresponds to the linear equations and constraints introduced above by the MILP transformations.

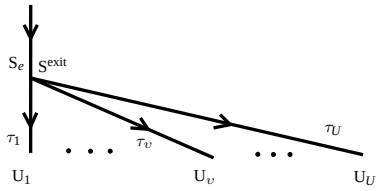


Figure 4.9: Unloading stations close together.

Case 2: more unloading stations close together

We now determine the route choice model for a network of tracks with more unloading stations close together as illustrated in Figure 4.9 where, without loss of generality, we consider that a junction can directly serve all unloading stations (this can be done by lumping together a sequence of junctions that are located closely together and connected to unloading stations). Let S_e with $e \in \{1,2,\ldots,S\}$ denote this junction. Also, let U denote the number of unloading stations in the system. Then the free-flow travel time from S_e to unloading station U_υ with $\upsilon \in \{1,\ldots,U\}$, is expressed by τ_υ which is an integer multiple of τ_s.

Assumptions In this case we make an additional assumption:

A_{11}: Out of the total demand of bags, a certain fraction ρ_υ of bags have to be transported to unloading station U_υ for $\upsilon = 1,\ldots,U$ such that $\sum_{\upsilon=1}^{U} \rho_\upsilon = 1$. So, at junction S_e the stream of bags is split into υ substreams according to the fractions ρ_υ.

Model Note that one can virtually expand junction S_e to two junctions S_e and S_e connected via a link of length 0 (S^{exit} has only one incoming link). Then the flow model for all junctions in the network except S^{exit} can be derived as in Case 1 above. Next we will determine the flow model corresponding to the junction S^{exit}.

The stream of DCVs waiting at the end of the link going into S^{exit} can be now divided into substreams (each substream corresponding to an unloading station). Let $q_\upsilon^{exit}(k)$ denote the queue length (located at the end of the link going into S^{exit}) of the substream corresponding to unloading station U_υ at time instant t_k. The evolution of $q_\upsilon^{exit}(k)$ is then defined as follows:

$$q_\upsilon^{exit}(k+1) = q_\upsilon^{exit}(k) + \left(\rho_\upsilon O_e(k) - U_\upsilon(k + \frac{\tau_\upsilon}{\tau_s})\right)\tau_s$$

with $O_e(k)$ the outflow of S_e and $U_\upsilon(k)$ the outflow of unloading station U_υ during $[t_k, t_{k+1})$. Note that the "max" operator is not needed here due to the definition of $U_\upsilon(k)$ ($U_\upsilon(k) \geq 0$ always by definition).

We consider two patterns that the low-level switch-out controller could follow:

***Pattern* 1:** During the time interval $[t_k, t_{k+1})$ the low-level switch-out controller at S^{exit} serves only one unloading station. To determine which unloading station to serve, we introduce the integer control variable $u^{exit}(k)$ that indicates the index

of the unloading station to be served during the time interval $[t_k, t_{k+1})$. Then the outflow of unloading station U_1 during $[t_k, t_{k+1})$ is given by:

$$U_\upsilon(k) = \begin{cases} \min\Big(O^{\max}, \dfrac{q_\upsilon^{\text{exit}}(k-\frac{\tau_\upsilon}{\tau_\mathrm{s}})}{\tau_\mathrm{s}} + \rho_\upsilon O_e(k-\dfrac{\tau_\upsilon}{\tau_\mathrm{s}})\Big) & \text{if } \upsilon = u^{\text{exit}}(k-\dfrac{\tau_\upsilon}{\tau_\mathrm{s}}) \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

***Pattern* 2:** During the time interval $[t_k, t_{k+1})$ all unloading stations are served (we consider fast switching). Then each partial queue is emptied according to the fractions ρ_υ for $\upsilon = 1, \ldots, U$. Note that we now assume a different type of switch for S^{exit} (since S^{exit} has more than two outgoing links). This switch can be more expensive so as to allow fast switching. However, we only need one, so, we can spend more money on it. The outflow of unloading station U_1 during $[t_k, t_{k+1})$ is then given by:

$$U_\upsilon(k) = \min\Big(O^{\max}, \frac{q_\upsilon^{\text{exit}}(k-\frac{\tau_\upsilon}{\tau_\mathrm{s}})}{\tau_\mathrm{s}} + \rho_\upsilon O_e(k-\frac{\tau_\upsilon}{\tau_\mathrm{s}})\Big). \tag{4.19}$$

MILP model The MILP equivalents for the additional equations describing the outflow of an unloading station except (4.18) can be derived using a reasoning similar to that above.

We now briefly explain how we write the MILP equivalents for (4.18). One can introduce U binary variables $\delta_1^{\text{exit}}(k), \ldots, \delta_U^{\text{exit}}(k)$ where $\delta_\upsilon^{\text{exit}}(k) = 1$ means that unloading station U_υ is served during the time interval $[t_k, t_{k+1})$. Additionally, we introduce the constraint that:

$$\sum_{\upsilon=1}^{U} \delta_\upsilon^{\text{exit}}(k) = 1$$

which means that there can only be one unloading station served at the time. Then, for $\upsilon = 1, \ldots, U$, we have:

$$U_\upsilon(k) = \delta_\upsilon^{\text{exit}}(k-\frac{\tau_\upsilon}{\tau_\mathrm{s}}) \min\Big(O^{\max}, \frac{q_\upsilon^{\text{exit}}(k-\frac{\tau_\upsilon}{\tau_\mathrm{s}})}{\tau_\mathrm{s}} + \rho_\upsilon O_e(k-\frac{\tau_\upsilon}{\tau_\mathrm{s}})\Big)$$

So, one can now write the complete MILP model for the case of a network with more unloading stations close together.

Case 3: more unloading stations far apart

Finally, we analyze the case where the track network has more unloading stations far apart.

Assumptions If for the previous case (of a network with more unloading stations close together), we made the additional assumption $\mathbf{A}_{11}$, for this case we make different additional assumptions:

$\mathbf{A}_{12}$**:** We now define partial demand patterns at loading stations. So, each loading station has a demand pattern corresponding to each end point. Then, for each loading station L_i, with $i \in \{1, 2, \ldots, L\}$ and for each unloading station U_υ, with $\upsilon \in \{1, 2, \ldots, U\}$,

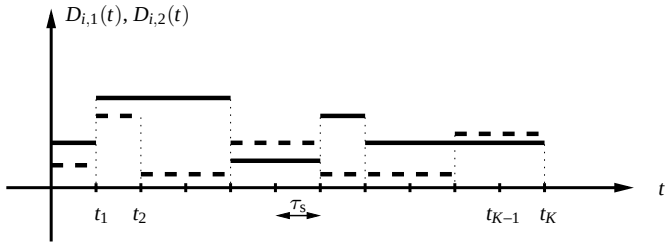


Figure 4.10: Demand profile at loading station L_i *for a network with two unloading stations. The solid line corresponds to unloading station* U_1 *and the dashed line demand corresponds to* U_2.

there is a dynamic, piecewise constant demand pattern $D_{i,v}(\cdot)$ as shown in Figure 4.10 where $D_{i,v}(k)$ is the demand of bags at loading station L_i with destination U_v in the time interval $[t_k, t_{k+1})$ for $k = 0, \ldots, K-1$ with K the demand horizon (we assume that beyond t_K the demand is 0).

As example we illustrate in Figure 4.10 the dynamic demand pattern at loading station L_i, with $i \in \{1, 2, \ldots, L\}$ for a network with two unloading stations. In this figure the piecewise constant demand represented as a solid line corresponds to unloading station U_1, and is denoted by $D_{i,1}(t)$, while the dashed piecewise constant demand corresponds to U_2, and is denoted by $D_{i,2}(t)$. Then for a network with U unloading stations, the total demand of L_i during the time interval $[t_k, t_{k+1})$ is given by $D_i(k) = \sum_{v=1}^{U} D_{i,v}(k)$.

$\mathbf{A}_{13}$: Since we deal with partial demands at each loading station, we assume that the DCVs wait before the junctions in partial vertical queues according to the unloading station towards which the DCVs travel.

$\mathbf{A}_{14}$: Recall from $\mathbf{A}_1$ that we assume enough DCVs present at loading stations so that when a bag is at a loading station, there is a DCV ready to transport it. Additionally, we now assume that no buffer overflow can occur on the link connected to the loading station, and than the demand at a loading station is smaller than the loading capacity. As a consequence, no queues can appear at the loading stations.

Model The control time step for each junction in the network is τ_s. So, at each time step k, for each junction S_s with 2 incoming links, we compute the position of the switch-in during the time interval $[t_k, t_{k+1})$ that has been denoted by $u_s^{\text{sw\_in}}(k)$. The position of the switch-out is controlled as presented next.

We consider two patterns that the switch-out controller of a junction S_s with 2 outgoing links could follow. The first pattern is a realistic one, which has been already used for the first two cases presented above, while the second pattern has been considered in order to decrease the computational complexity.

***Pattern* 1:** During the time interval $[t_k, t_{k+1})$ the switch-out controller serves only one outgoing link of S_s. To determine which outgoing link to serve, we compute $u_s^{\text{sw\_out}}(k)$

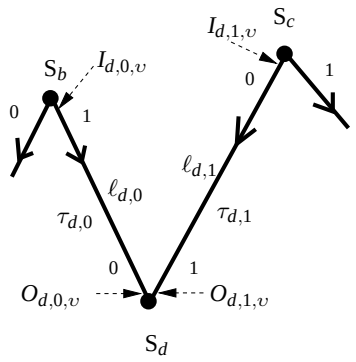


Figure 4.11: Network elements when the switch-out is controlled according to ***Pattern*** 1.

which will then be a variable of the optimization problem presented at the end of this subsection.

Pattern 2**:** During the time interval $[t_k, t_{k+1})$ a low-level switch-out controller serves both outgoing links. This pattern gives a fair distribution over all the outflows while considering fixed turning rates for each junction S_s as presented below. Note that when using this pattern, no extra variables are introduced. Therefore, we will solve simpler optimization problems, and this will give faster results. In future work we will define a similar pattern for the control of the switch-in and then compare the performance and the computation time obtained when using *Pattern* 2 for the control of both the switch-in and switch-out.

According to these patterns, we derive the route choice model by referring to the network cell illustrated in Figure 4.11.

Pattern 1**:** In this case we consider partial queues at *the end of each link* that correspond to each unloading station. Let $q_{s,l,\upsilon}(k)$ denote the length of the partial queue during the time interval $[t_k, t_{k+1})$ at the end of the incoming link l of junction S_s that consists of DCVs going towards unloading station U_υ with $\upsilon \in \{1, \ldots, U\}$. Then the evolution of length of the partial queue is given by:

$$q_{d,l,\upsilon}(k+1) = q_{d,l,\upsilon}(k) + \Big(I_{d,l,\upsilon}\big(k - \frac{\tau_{d,l}}{\tau_s}\big) - O_{d,l,\upsilon}(k) \Big) \tau_s$$

where $I_{d,l,\upsilon}(k)$ is the partial inflow corresponding to U_υ of link $\ell_{d,l}$ and $O_{d,l,\upsilon}(k)$ is the partial outflow corresponding to U_υ of link $\ell_{d,l}$ during the time interval $[t_k, t_{k+1})$. The partial inflows $I_{d,l,\upsilon}(k)$ are defined at the beginning of a link, while the partial outflows $O_{d,l,\upsilon}(k)$ are defined at the end of a link as pointed in Figure 4.11.

If junction S_b has 2 incoming links, the inflow $I_{d,0,\upsilon}(k)$ is defined as:

$$I_{d,0,\upsilon}(k) = u_b^{\text{sw\_out}}(k) \Big(\big(1 - u_b^{\text{sw\_in}}(k)\big) O_{b,0,\upsilon}(k) + u_b^{\text{sw\_in}}(k) O_{b,1,\upsilon}(k) \Big),$$

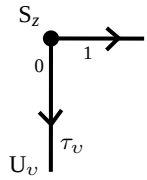


Figure 4.12: Junction S_z *is directly connected to* U_υ*. The unloading station is connected via link* 0 *out of* S_z*.*

while if S_b has only one incoming link the inflow $I_{d,0,\upsilon}(k)$ is defined as:

$$I_{d,0,\upsilon}(k) = u_b^{\text{sw\_out}}(k) O_{b,0,\upsilon}(k).$$

Similarly, one can define $I_{d,1,\upsilon}(k)$. Note that if a junction S_s with $s \in \{1,2,\ldots,S\}$ is directly connected to a loading station L_i $i \in \{1,2,\ldots,L\}$ via the incoming link $l \in \{0,1\}$ of S_s, then

$$I_{s,l,\upsilon}(k) = D_{i,\upsilon}(k).$$

The partial outflows $O_{s,l,\upsilon}(k)$ at the end of link $\ell_{s,l}$ with $l = u_s^{\text{sw\_in}}(k)$ are determined such that we have maximal exhaustion of the available capacity as described in **Algorithm 5** for $O_{s,l,\upsilon}(k) = O_\upsilon^{\text{alg}}(k)$ and $q_{s,l,\upsilon}(k) = q_\upsilon^{\text{alg}}(k)$ where $O_\upsilon^{\text{alg}}(k)$ and $q_\upsilon^{\text{alg}}(k)$ are variables that characterize **Algorithm 5**. Note that if junction S_s has 2 incoming links, then $O_{s,1-l,\upsilon}(k) = 0$ since only the partial queues at the end of the incoming link indexed by $l = u_s^{\text{sw\_in}}(k)$ are emptied during $[t_k, t_{k+1})$.

Without loss of generality we assume that for any junction S_z directly connected to U_υ, the unloading station is connected via link 0 out of S_z, see Figure 4.12.

Then, the outflow of unloading station U_υ during the period $[t_k, t_{k+1})$ is given by:

$$U_\upsilon(k) = \min\left(\left(1 - u_z^{\text{sw\_out}}(k - \frac{\tau_\upsilon}{\tau_s})\right) O_{z,0,\upsilon}(k - \frac{\tau_\upsilon}{\tau_s}), O^{\max}\right)$$

with τ_υ the free-flow travel time of the link directly connected to unloading station U_υ with $\upsilon \in \{1,\ldots,U\}$, τ_υ is considered to be an integer multiple of τ_s.

***Pattern* 2:** In this case we consider partial queues at *each junction* S_s with $s \in \{1,\ldots,S\}$ corresponding to each unloading station U_υ. The length of the partial queue at junction S_s that consists of DCVs going towards unloading station U_υ is denoted by $q_{s,\upsilon}$. Then we determine the partial outflows $O_{s,\upsilon}(k)$ for a junction S_s such that $\sum_{\upsilon=1}^{U} O_{s,\upsilon}(k) \leq O^{\max}$. To this aim we consider again a fair distribution over all flows as described in **Algorithm 5** for $O_{s,\upsilon}(k) = O_\upsilon^{\text{alg}}(k)$ and $q_{s,\upsilon}(k) = q_\upsilon^{\text{alg}}(k)$, and $\xi_{s,\upsilon}(k) = I_\upsilon^{\text{alg}}(k)$ with $\xi_{s,\upsilon}(k)$ the number of DCVs going towards unloading station U_υ that enter the partial queue at junction S_s during the time interval $[t_k, t_{k+1})$. The partial inflows $\xi_{s,\upsilon}(k)$ and the partial outflows $O_{s,l,\upsilon}(k)$ are defined at for each junction S_s as pointed in Figure 4.13.

Based on off-line optimization, for each junction S_s we can determine U fixed turning rates $\eta_{s,\upsilon}$ with $\upsilon = 1,\ldots,U$. These fixed turning rates represent the fraction of the partial queue $q_{s,\upsilon}(k)$ that will be sent to link 0 out of S_s during $[t_k, t_{k+1})$.

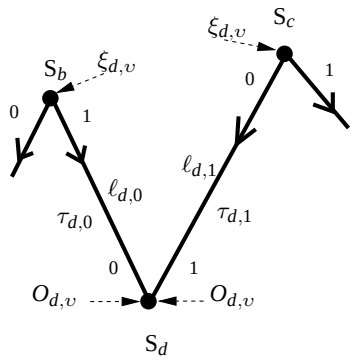


Figure 4.13: Network elements when the switch-out is controlled by a low level controller according to ***Pattern*** 2.

Then $\tau_s \sum_{\upsilon=1}^{U} \eta_{s,\upsilon} O_{s,\upsilon}(k)$ DCVs will be sent towards the outgoing link 0 of S_s, and $\tau_s \sum_{\upsilon=1}^{U} (1-\eta_{s,\upsilon}) O_{s,\upsilon}(k)$ DCVs will be sent towards its outgoing link 1.

Then the evolution of the length of the partial queue is given by:

$$q_{d,\upsilon}(k+1) = q_{d,\upsilon}(k) + \big(\xi_{d,\upsilon}(k) - O_{d,\upsilon}(k)\big)\tau_s$$

where $\xi_{d,\upsilon}(k)$ expresses the number of DCVs going towards unloading station U_υ that enter the partial queue at junction S_d during the time interval $[t_k, t_{k+1})$:

$$\xi_{d,\upsilon}(k) = (1-u_d^{\text{sw\_in}}(k))(1-\eta_{b,\upsilon})O_{b,\upsilon}(k-\frac{\tau_{d,0}}{\tau_s}) + u_d^{\text{sw\_in}}(k)\eta_{c,\upsilon}O_{c,\upsilon}(k-\frac{\tau_{d,1}}{\tau_s}).$$

Note that if a junction S_s with $s \in \{1,2,\ldots,S\}$ is directly connected to a loading station L_i $i \in \{1,2,\ldots,L\}$ then

$$\xi_{s,\upsilon}(k) = D_{i,\upsilon}(k).$$

Accordingly, $U_\upsilon(k) = \min\left(O^{\max}, \left(1-u_z^{\text{sw\_out}}(k-\frac{\tau_\upsilon}{\tau_s})\right)\eta_{z,\upsilon}O_{z,\upsilon}(k-\frac{\tau_\upsilon}{\tau_s})\right)$.

Algorithm 5 describes the procedure that we consider in order to determine the distribution of the partial outflows such that we have maximal exhaustion of the available capacity. We use this algorithm since it results in a fair distribution over all the outflows.

Algorithm[10] **5. Outflow distribution at the end of link** $\ell_{s,l}$

1: $\Omega = \{1,2,\ldots,U\}$
2: **while** $\Omega \neq \emptyset$ **do**
3: $\quad \Lambda = \arg\min_{\upsilon\in\Omega} \left(q_\upsilon^{\text{alg}}(k) + I_\upsilon^{\text{alg}}(k)\tau_s\right)$
4: $\quad$ **for** all $\upsilon \in \Lambda$ **do**
5: $\quad\quad O_\upsilon^{\text{alg}}(k) = \min\left(\frac{O^{\max}}{|\Omega|}, \frac{q_\upsilon^{\text{alg}}(k)}{\tau_s} + I_\upsilon^{\text{alg}}(k)\right)$

[10] In **Algorithm 5**, $|\Omega|$ represents the cardinality of the set Ω.

```
    O^max ← O^max − O_υ^alg(k)
  end for
  Ω ← Ω \ Λ
end while
```

Let us now consider *Pattern* 1 and derive (as example) the output of **Algorithm 5** for link $\ell_{d,l}$ of the cell illustrated in Figure 4.8 and for $U = 2$. According to **Algorithm 5**, if $q_{d,l,1}(k) + I_{d,l,1}(k)\tau_s \geq q_{d,l,2}(k) + I_{d,l,2}(k)\tau_s$ then the outflow $O_{d,l,\upsilon}(k)$, for $\upsilon = 1,2$ is given by:

$$O_{d,l,1}(k) = \min\left(\frac{O^{\max}}{2}, \frac{q_{d,l,1}(k)}{\tau_s} + I_{d,l,1}(k)\right) \tag{4.20}$$

$$O_{d,l,2}(k) = \min\left(O^{\max} - \frac{q_{d,l,1}(k)}{\tau_s} - I_{d,l,1}(k), \frac{q_{d,l,2}(k)}{\tau_s} + I_{d,l,2}(k)\right) \tag{4.21}$$

and otherwise (i.e., if $q_{d,l,1}(k) + I_{d,l,1}(k)\tau_s < q_{d,l,2}(k) + I_{d,l,2}(k)\tau_s$) the outflow $O_{d,l,\upsilon}(k)$ is given by:

$$O_{d,l,1}(k) = \min\left(O^{\max} - \frac{q_{d,l,2}(k)}{\tau_s} - I_{d,l,2}(k), \frac{q_{d,l,1}(k)}{\tau_s} + I_{d,l,1}(k)\right) \tag{4.22}$$

$$O_{d,l,2}(k) = \min\left(\frac{O^{\max}}{2}, \frac{q_{d,l,2}(k)}{\tau_s} + I_{d,l,2}(k)\right). \tag{4.23}$$

Similarly, one can derive the outflow $O_{s,l,\upsilon}(k)$ with $s \in \{1,2,\ldots,S\}$, $l \in \{0,1\}$, and $\upsilon \in \{1,2,\ldots,U\}$, for networks of tracks with $U > 2$. Then (if $U > 2$), one gets more complex formulas, but these new formulas can still be written using "if-then-else" statements and "min" operators.

MILP model We now transform (4.20)–(4.23) into their MILP equivalents. The rest of the MILP route choice model for the case with more unloading stations far apart, can be derived using a reasoning similar to that above (see pages 76–77 describing the MILP route choice model for a network with one unloading station).

To transform (4.20)–(4.23), we introduce the binary variables $\delta_{d,l,1}(k)$, $\delta_{d,l,2}(k)$, $\delta_{d,l,3}(k)$, and $\delta_{d,l,4}(k)$ such that:

- $\delta_{d,l,1}(k) = 1$ if and only if $q_{d,l,1}(k) + I_{d,l,1}(k)\tau_s \geq q_{d,l,2}(k) + I_{d,l,2}(k)\tau_s$,
- $\delta_{d,l,2}(k) = 1$ if and only if $q_{d,l,1}(k) + I_{d,l,1}(k)\tau_s \leq \dfrac{O^{\max}}{2}\tau_s$,
- $\delta_{d,l,3}(k) = 1$ if and only if $q_{d,l,2}(k) + I_{d,l,2}(k)\tau_s \leq \dfrac{O^{\max}}{2}\tau_s$,
- $\delta_{d,l,4}(k) = 1$ if and only if $q_{d,l,2}(k) + I_{d,l,2}(k)\tau_s \leq O^{\max}\tau_s - q_{d,l,1}(k) - I_{d,l,1}(k)\tau_s$,

together with the system of linear inequalities corresponding to Property **P1**. Then the

outflows $O_{d,l,1}(k)$ and $O_{d,l,2}(k)$ can be written as follows:

$$\begin{aligned} O_{d,l,1}(k) =& \delta_{d,l,1}(k)\left(\delta_{d,l,2}(k)\left(\frac{q_{d,l,1}(k)}{\tau_s}+I_{d,l,1}(k)\right)+\left(1-\delta_{d,l,2}(k)\right)\frac{O^{\max}}{2}\right)+\left(1-\delta_{d,l,1}(k)\right) \\ & \left(\delta_{d,l,4}(k)\left(\frac{q_{d,l,1}(k)}{\tau_s}+I_{d,l,1}(k)\right)+\left(1-\delta_{d,l,4}(k)\right)\left(O^{\max}-\frac{q_{d,l,2}(k)}{\tau_s}-I_{d,l,2}(k)\right)\right) \end{aligned} \tag{4.24}$$

$$\begin{aligned} O_{d,l,2}(k) =& \delta_{d,l,1}(k)\left(\delta_{d,l,4}(k)\left(\frac{q_{d,l,2}(k)}{\tau_s}+I_{d,l,2}(k)\right)+\right. \\ & \left.\left(1-\delta_{d,l,4}(k)\right)\left(O^{\max}-\frac{q_{d,l,1}(k)}{\tau_s}-I_{d,l,1}(k)\right)\right)+ \\ & \left(1-\delta_{d,l,1}(k)\right)\left(\delta_{d,l,3}(k)\left(\frac{q_{d,l,2}(k)}{\tau_s}+I_{d,l,2}(k)\right)+\left(1-\delta_{d,l,3}(k)\right)\frac{O^{\max}}{2}\right). \end{aligned} \tag{4.25}$$

To transform (4.24)–(4.25) into MILP equations one has to further introduce real-valued scalar variables and the corresponding systems of linear inequalities corresponding to Property **P2** using a reasoning similar to that above (see pages 76–77).

Model predictive route choice control

Next we define the general MILP model that will be used in MPC framework, the MPC objective function, and the MPC optimization problem for both the nonlinear and the MILP case.

MPC MILP model For each MPC step k corresponding to time instant t_k, we now derive the overall MILP model. Let $\mathbf{q}_{k+1,N}$ be a vector that consists of all the partial queue lengths at MPC step k, over an horizon of N steps. Then the general MPC MILP model can be written as follows:

$$\mathbf{q}_{k+1,N} = \mathbf{\Lambda}_{k,N}\mathbf{x}_{k,N}^{\mathrm{MILP}} + \boldsymbol{\gamma}_{k,N}$$

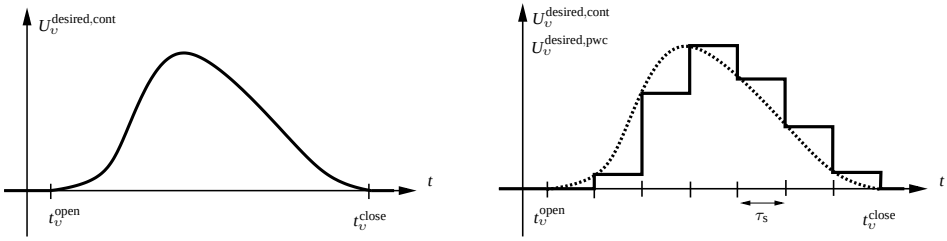
with a properly defined matrix $\mathbf{\Lambda}_{k,N}$ and vector $\boldsymbol{\gamma}_{k,N}$, where $\mathbf{x}_{k,N}^{\mathrm{MILP}}$ consists of all the variables for the MILP route choice model (i.e., inputs, control variables, and extra variables introduced by the MILP transformations) and satisfies a system of linear equations and inequalities

$$\begin{aligned} \mathbf{A}_{k,N}^{\mathrm{eq}}\mathbf{x}_{k,N}^{\mathrm{MILP}} &= \mathbf{b}_{k,N}^{\mathrm{eq}} \\ \mathbf{A}_{k,N}\mathbf{x}_{k,N}^{\mathrm{MILP}} &\leq \mathbf{b}_{k,N}, \end{aligned}$$

which corresponds to the linear equations and constraints introduced above by the MILP transformations.

MPC objective function Recall that the baggage handling system performs optimally if each of the bags to be handled arrives at its given end point within a specific time window[11]

[11] In order to simplify the explanation, we now consider that each unloading station is assigned to one flight only. However, this case can be easily extended to the general case, where more flights are assigned to an unloading station.



(a) continuous outflow profile (b) piecewise constant outflow profile

Figure 4.14: Desired outflow profile at unloading station U_υ.

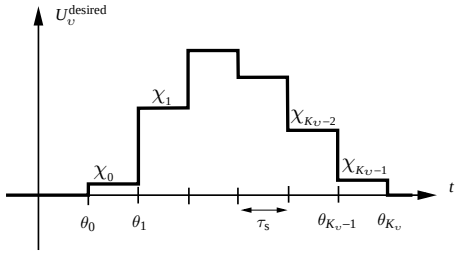


Figure 4.15: Desired piecewise constant outflow profile at unloading station U_υ.

$[t_\upsilon^{\text{close}} - \tau_\upsilon^{\text{open}}, t_\upsilon^{\text{close}})$ where $t_\upsilon^{\text{close}}$ is the time instant when the end point U_υ closes and the last bags are loaded onto the plane, and $\tau_\upsilon^{\text{open}}$ is the time period for which the end point U_υ stays open for a specific flight. We have assumed $t_\upsilon^{\text{close}}$ and $\tau_\upsilon^{\text{open}}$ to be integer multiple of τ_{s}. We consider the objective of reaching a desired outflow for each unloading station.

Note that the desired outflow at each unloading station is in general a dynamic signal. Let $U_\upsilon^{\text{desired,cont}}$ denote the desired flow profile at unloading station U_υ as sketched in Figure 4.14(a) where the area under the curve $U_\upsilon^{\text{desired,cont}}$ equals the total number of bags out of the total demand to be sent to U_υ. Note that outside the time window $[t_\upsilon^{\text{open}}, t_\upsilon^{\text{close}})$ with $t_\upsilon^{\text{open}} = t_\upsilon^{\text{close}} - \tau_\upsilon^{\text{open}}$ no bags should enter the incoming link of unloading station U_υ, and consequently, $U_\upsilon^{\text{desired,cont}}(t) = 0$ for t outside the given time window. Since we want to use this profile for our control, we first have to sample it and approximate it with a piecewise constant one.

The most straightforward way to perform this approximation is to define the piecewise constant flow $U_\upsilon^{\text{desired,pwc}}(t) = U_\upsilon^{\text{desired,cont}}(t_k)$ for $t_k \leq t < t_{k+1}$ and $k \in \mathbb{N}$, as illustrated in Figure 4.14(b). However, one can perform an even better approximation by computing the piecewise constant outflow profile that minimizes the area between the desired continuous outflow profile $U_\upsilon^{\text{desired,cont}}(t)$ and the piecewise constant outflow profile $U_\upsilon^{\text{desired,pwc}}(t)$. This can be obtained by solving off-line, for each unloading station, the optimization problem defined below. For the sake of simplicity of notations let us consider unloading station U_υ and omit the subscript υ for the variables that clearly refer to U_υ. Furthermore, we define χ_k as follows, see, e.g., Figure 4.15:

$$\text{for } t \in [t_\upsilon^{\text{open}}, t_\upsilon^{\text{close}}),\ U_\upsilon^{\text{desired,pwc}}(t) = \chi_j \text{ if } \theta_j \leq t < \theta_{j+1} \text{ with } \theta_j = t_\upsilon^{\text{open}} + j\tau_{\text{s}}.$$

Then one can approximate $U_\upsilon^{\text{desired,pwc}}(t)$ for $t \in [t_\upsilon^{\text{open}}, t_\upsilon^{\text{close}})$ with the solution of the following optimization problem:

$$\min_{\chi_0,\ldots,\chi_{K_\upsilon-1}} \sum_{j=0}^{K_\upsilon-1} \int_{\theta_j}^{\theta_{j+1}} \left(U_\upsilon^{\text{desired,cont}}(t) - \chi_j\right)^2 \mathrm{d}t \tag{4.26}$$

where $K_\upsilon = \frac{\tau_\upsilon^{\text{open}}}{\tau_{\text{s}}}$. Let χ_0^*, χ_1^*, $\ldots$, $\chi_{K_\upsilon-1}^*$ denote the solution of (4.26).

Then, the optimal piecewise constant flow at unloading station U_υ is given by:

$$\begin{cases} U_\upsilon^{\text{desired}}(j) = \chi_{j-k_\upsilon^{\text{open}}}^* & \text{for } k_\upsilon^{\text{open}} \leq j \leq k_\upsilon^{\text{close}} - 1 \\ U_\upsilon^{\text{desired}}(j) = 0 & \text{otherwise} \end{cases}$$

with $j \in \mathbb{N}$, k_υ^{open} defined as $k_\upsilon^{\text{open}} = \frac{t_\upsilon^{\text{open}} - t_0}{\tau_{\text{s}}}$, and $k_\upsilon^{\text{close}}$ defined as $k_\upsilon^{\text{close}} = \frac{t_\upsilon^{\text{close}} - t_0}{\tau_{\text{s}}}$.

The first objective of the MILP controller is to reach a desired outflow for each unloading station during the simulation period. Let $U_\upsilon(i)$ denote the actual outflow of unloading station U_υ during the period $[t_i, t_{i+1})$ with $i \in \mathbb{N}$. Then, one could define the following objective function during the period $[t_i, t_{i+1})$:

$$J_i^{\text{outflow}}\big(U_1(i), \ldots, U_U(i)\big) = \sum_{\upsilon=1}^{U} w_\upsilon \left| U_\upsilon(i) - U_\upsilon^{\text{desired}}(i) \right|$$

where w_υ is a nonnegative weighting parameter that expresses the penalty on the unloading stations (in this way we can penalize differently the unloading stations depending on, e.g., the priority of the assigned flight).

However, to add some additional gradient to this objective function and make sure that all the bags will be handled, we also consider the weighted length of queues at each junction in the network, but only for time steps bigger than or equal to $k_\upsilon^{\text{close}}$ with $\upsilon \in \{1, \ldots, U\}$. Then we define the additional penalty:

$$J_i^{\text{add}}\big(q_1(i), \ldots, q_S(i)\big) = \begin{cases} 0 & \text{if } i < k_\upsilon^{\text{close}} \\ \sum_{s=1}^{S} \sum_{\upsilon=1}^{U} \lambda_{s,\upsilon} q_s(i) & \text{otherwise} \end{cases}$$

where $q_s(i)$ is the summation of the lengths of the partial queues at time instant t_i consisting of DCVs that wait before junction S_s, while $\lambda_{s,\upsilon}$ is a nonnegative weighting parameter that expresses the penalty[12] on junction S_s.

Finally, at MPC step k which corresponds to time instant t_k, the MPC performance index is defined as follows:

$$J_{k,N}^{\text{MILP}}\big(\mathbf{x}_{k,N}^{\text{MILP}}, \mathbf{q}_{k+1,N}\big) = \sum_{i=k}^{k+N-1} J_i^{\text{outflow}}\big(U_1(i), \ldots, U_U(i)\big) + \sum_{i=k+1}^{k+N} J_i^{\text{add}}\big(q_1(i), \ldots, q_S(i)\big) \tag{4.27}$$

[12] Since a baggage handling system has to transport all the checked in or transfer bags to the corresponding end points before the planes have to be loaded, the weighting parameter $\lambda_{s,\upsilon}$ is set to be proportional to the shortest distance from junction S_s to unloading station U_υ.

Next, we want to write the MILP optimization problem at MPC time step k. Let us first consider the simplest case where $k+N < k_\upsilon^{\text{close}}$, case for which

$$J_{k,N}^{\text{MILP}}\left(\mathbf{x}_{k,N}^{\text{MILP}}, \mathbf{q}_{k+1,N}\right) = \sum_{i=k}^{k+N-1} \sum_{\upsilon=1}^{U} w_\upsilon \left| U_\upsilon(i) - U_\upsilon^{\text{desired}}(i) \right|,$$

since $\sum_{i=k}^{k+N-1} J_i^{\text{add}}\left(q_1(i), \ldots, q_S(i)\right) = 0$.

Then one can write the MPC optimization problem for an MILP model as follows:

$$\begin{aligned}
&\min_{U_\upsilon(k),\ldots,U_\upsilon(k+N-1)} \sum_{i=k}^{k+N-1} \sum_{\upsilon=1}^{U} w_\upsilon U_\upsilon^{\text{diff}}(i) \\
&\text{subject to} \\
&\qquad \text{MILP model} \\
&\qquad \text{MILP constraints} \\
&\qquad U_\upsilon^{\text{diff}}(i) \geqslant U_\upsilon(i) - U_\upsilon^{\text{desired}}(i) \quad \text{for } i = k, \ldots, k+N-1 \\
&\qquad U_\upsilon^{\text{diff}}(i) \geqslant -U_\upsilon(i) + U_\upsilon^{\text{desired}}(i) \text{ for } i = k, \ldots, k+N-1.
\end{aligned}$$

Then the MPC optimization problem above is a linear programming problem that has as optimal solution

$$U_\upsilon^{\text{diff},*}(i) = \max(U_\upsilon^*(i) - U_\upsilon^{\text{desired}}(i), -U_\upsilon^*(i) + U_\upsilon^{\text{desired}}(i)) = \left| U_\upsilon^*(i) - U_\upsilon^{\text{desired}}(i) \right|.$$

For the case where $k+N \geq k_\upsilon^{\text{close}}$ we will still obtain an MILP optimization problem by applying a similar procedure because the penalty J_k^{add} is linear.

Optimization problems Next, we formulate the optimization problem for both the nonlinear and the MILP model formulations at time step k.

The nonlinear MPC optimization problem is defined as:

$$\begin{aligned}
&\min_{\mathscr{U}_{k,N}} \quad J_{k,N}^{\text{nonlinear}}\left(\mathbf{t}(k)\right) \\
&\text{subject to} \\
&\qquad \mathbf{t}(k) = \mathscr{M}^{\text{switch_ctrl}}\left(\mathscr{T}, \mathbf{x}(t_k), \mathscr{U}_{k,N}\right) \\
&\qquad \mathscr{C}(\mathbf{t}(k)) \leq 0
\end{aligned} \tag{4.28}$$

where

- $J_{k,N}^{\text{nonlinear}}\left(\mathbf{t}(k)\right)$ penalizes the absolute difference between the actual outflow and the desired outflow at each unloading station, and the queues in the network, as (4.27) does,
- $\mathscr{T}$ is the L-tuple that comprises the vectors of bag arrival times $\mathscr{T} = (\mathbf{t}_1^{\text{arrival}}, \ldots, \mathbf{t}_L^{\text{arrival}})$ defined in Section 4.1.1,
- $\mathbf{x}(t_k)$ is the state of the system at time instant t_k,
- the N-tuple $\mathscr{U}_{k,N} = \left(\mathbf{u}(k), \mathbf{u}(k+1), \ldots, \mathbf{u}(k+N-1)\right)$ represents the route choice control, with $\mathbf{u}(k+j)$ for $j = 0, 1, \ldots, N-1$ consisting of the positions of the switch-in and switch-out of all junctions in the network at time instant t_{k+j}.

The outflows of the unloading stations are determined via simulation when using the event-driven model presented in Section 4.2.2.

Similarly, the MILP MPC optimization problem is defined as:

$$\min_{\mathbf{x}_{k,N}^{\text{MILP}}} \quad J_{k,N}^{\text{MILP}}\left(\mathbf{x}_{k,N}^{\text{MILP}}, \mathbf{q}_{k+1,N}\right)$$

subject to

$$\begin{aligned} \mathbf{q}_{k+1,N} &= \boldsymbol{\Lambda}_{k,N}\mathbf{x}_{k,N}^{\text{MILP}} + \boldsymbol{\gamma}_{k,N} \\ \mathbf{A}_{k,N}^{\text{eq}}\mathbf{x}_{k,N}^{\text{MILP}} &= \mathbf{b}_{k,N}^{\text{eq}} \\ \mathbf{A}_{k,N}\mathbf{x}_{k,N}^{\text{MILP}} &\leq \mathbf{b}_{k,N} \end{aligned} \tag{4.29}$$

To solve the MILP optimization problem one could use solvers such as CPLEX, Xpress-MP, GLPK, see, e.g., [3].

In general, computing the route for each DCV in the network when solving nonlinear MPC optimization problems will give a better performance than when solving the MILP optimization problems (due to the simplifying assumptions used to write the MILP model), but at the cost of higher computational efforts. So, one could use MILP to compute a good initial point for the nonlinear optimization problem and this will reduce the computation time. One could also use directly the MILP solution, but at the cost of suboptimality. The results obtained when using MPC with nonlinear and MILP formulation respectively, for its optimization problems will be presented in Section 4.5.3.

4.4.6 Decentralized heuristic approach

In this subsection and the next one we propose heuristic approaches that could be used to efficiently control the route of each DCV, for the model determined in Section 4.2. Each switch is now locally controlled based on heuristic rules as presented next. Note that the local switch control of the decentralized heuristic approach is determined based only on local information regarding the flow of DCVs on the incoming and outgoing links of a junction. Consider junction S_s with $s \in \{1, 2, ..., S\}$.

4.4.6.1 Control of the switch-in

If S_s has a switch-in, as the junctions illustrated in Figure 4.16, every time when a bag enters one of the incoming links of S_s we update the local control of the switch-in at S_s. Let $t_s^{\text{compute\_sw\_in}}$ be such a time instant. Then we compute (as presented below) the control variable $\tau_s^{\text{sw–in}}$, which represents the time period until the position of the switch-in has to be changed next.

For a junction S_s, we define the following variables:

- $\Gamma_{s,l}$ is the set of bags transported by DCVs that travel on the incoming link $l \in \{0, 1\}$ of junction S_s at time instant $t_s^{\text{compute\_sw\_in}}$,
- $\rho_{s,l}^{\text{static}}$ is the total static priority of the bags transported on the incoming link l of junction S_s at time instant $t_s^{\text{compute\_sw\_in}}$, $\rho_{s,l}^{\text{static}} = \sum_{i \in \Gamma_{s,l}} \sigma_i$, where σ_i is the static priority of bag index i,

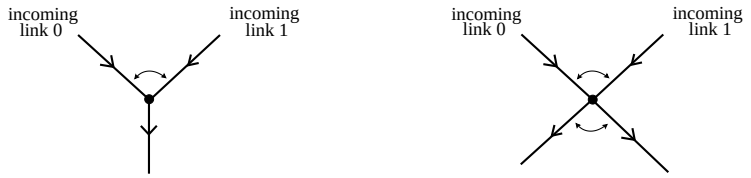


Figure 4.16: Junctions with a switch-in. The time when the next toggle will take place is determined by a local rule-based controller.

- $\rho_{s,l}^{\text{dyn}}$ is the total dynamic priority of the bags transported on the incoming link l of junction S_s at time instant $t_s^{\text{compute\_sw\_in}}$, $\rho_{s,l}^{\text{dyn}} = \sum_{i \in \Gamma_{s,l}} \frac{\hat{\delta}_i}{\delta_i^{\max}}$ with $\hat{\delta}_i$ the estimate of the actual time bag index i requires to get from its current position to its final destination in case of no congestion and maximum speed, and $\delta_i^{\max}$ the maximum time left to bag index i to spend in the system while still arriving at the plane on time. If bag index i misses the flight, then the bag has to wait for a new plane with the same destination. Hence, a new departure time is assigned to bag index i, and consequently $t_i^{\text{new\_end}}$ for bag index i is considered. Then the variable $\delta_i^{\max}$ is defined as follows:

$$\delta_i^{\max} = \begin{cases} t_i^{\text{close}} - t_s^{\text{compute\_sw\_in}} & \text{if } t_i^{\text{close}} - t_s^{\text{compute\_sw\_in}} > 0 \\ t_i^{\text{new\_end}} - t_s^{\text{compute\_sw\_in}} & \text{if } t_i^{\text{close}} - t_s^{\text{compute\_sw\_in}} \leq 0 \end{cases}$$

In order to determine the next position of the switch-in at junction S_s we compute a performance measure $p_{s,l}^{\text{sw\_in}}$ for $l = 0,1$ at time instant $t_s^{\text{compute\_sw\_in}}$. This performance measure takes into account the static and dynamic priorities of the bags transported by DCVs on the incoming link l, and the current position of the switch-in at junction S_s (due to the operational constraint $\mathbf{C}_3$ according to which the position of a switch at a junction can only change after minimum τ^{switch} time units):

$$p_{s,0}^{\text{sw\_in}} = w^{\text{st\_pr}} \rho_{s,0}^{\text{static}} + w^{\text{dyn\_pr}} \rho_{s,0}^{\text{dyn}} - w^{\text{sw\_in}} \tau^{\text{switch}} I_s^{\text{crt}} \tag{4.30}$$

$$p_{s,1}^{\text{sw\_in}} = w^{\text{st\_pr}} \rho_{s,1}^{\text{static}} + w^{\text{dyn\_pr}} \rho_{s,1}^{\text{dyn}} - w^{\text{sw\_in}} \tau^{\text{switch}} (1 - I_s^{\text{crt}}) \tag{4.31}$$

where I_s^{crt} denotes the current position of the switch-in at junction S_s (i.e. $I_s^{\text{crt}} = 0$ if the switch-in is positioned on the incoming link 0, and $I_s^{\text{crt}} = 1$ if the switch-in is positioned on the incoming link 1). The weighting parameters $w^{\text{st\_pr}}$, $w^{\text{dyn\_pr}}$, and $w^{\text{sw\_in}}$ can be tuned as explained in Section 4.4.6.3.

Let $z_{s,l} \in \Gamma_{s,l}$ denote the bag traveling on the incoming link l of S_s and which is closest to S_s and let $\tau_l^{\text{arrival\_at\_S}_s}$ be the time period that the DCV transporting bag $z_{s,l}$ needs to travel (at maximum speed) to reach S_s.

The position of the switch-in at S_s is toggled only if $p_{s,0}^{\text{sw\_in}} > p_{s,1}^{\text{sw\_in}}$ and $I_s^{\text{crt}} = 1$, or if $p_{s,1}^{\text{sw\_in}} > p_{s,0}^{\text{sw\_in}}$ and $I_s^{\text{crt}} = 0$. If this is the case, then the current position of the switch-in is toggled after

$$\tau_s^{\text{sw\_in}} = \max(\tau^{\text{switch}} - \tau_s^{\text{sw\_in\_prev}}, \tau_{1-I_s^{\text{crt}}}^{\text{arrival\_at\_S}_s})$$

time units where $\tau_s^{\text{sw\_in\_prev}}$ is the time for which the switch-in at junction S_s has been in its current position. Otherwise, we set $\tau_s^{\text{sw\_in}} = \infty$.

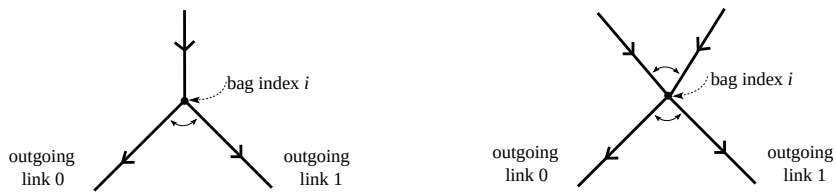


Figure 4.17: Junctions with a switch-out. The time when the next toggle will take place is determined by a local rule-based controller.

4.4.6.2 Control of the switch-out

If S_s has a switch-out, as the junctions illustrated in Figure 4.17, every time when a bag is positioned just before junction S_s and the switch-in at S_s allows its crossing, we update the local control of the switch-out at S_s. Let bag index i be the bag positioned just before junction S_s when the switch-in at S_s allows bag index i to cross S_s, and let $t_{s,i}^{\text{compute\_sw\_out}}$ be the time instant when this happens. Then we compute (as presented below) the control variable $\tau_s^{\text{sw\_out}}$, which represents the time period until the position of the switch-out has to be changed next. This goes as follows.

Assume that bag index i is at junction S_s. Let $S_{s,l}^{\text{next}}$ denote the junction that is connected to S_s via the outgoing link $l \in \{0,1\}$ of S_s, and let $S_{s,i}^{\text{dest}}$ be the end point of bag index i. Then, we can predict the unloading time $\hat{t}_{s,l,r,i}^{\text{unload}}$ of bag index i at $S_{s,i}^{\text{dest}}$, when traveling on link $l \in \{0,1\}$ out of S_s and next along route $r \in \mathscr{R}_{s,l,i}^{\text{next}}$ where $\mathscr{R}_{s,l,i}^{\text{next}}$ is the set of routes from $S_{s,l}^{\text{next}}$ to $S_{s,i}^{\text{dest}}$.

We estimate the time that bag index i needs to reach its end point similar to how we proceeded in Section 4.4.3. Hence, we can define $\hat{t}_{s,l,r,i}^{\text{unload}}$ as follows:

$$\hat{t}_{s,l,r,i}^{\text{unload}} = t_{s,i}^{\text{compute\_sw\_out}} + \hat{\tau}_{s,l,i}^{\text{link}} + \hat{\tau}_r^{\text{route}}$$

where

- $\hat{\tau}_{s,l,i}^{\text{link}}$ is the time we predict that bag index i spends on link l with $l \in \{0,1\}$ out of S_s. For this prediction we take:

$$\hat{\tau}_{s,l,i}^{\text{link}} = \begin{cases} \max\left(\dfrac{d_{s,l}^{\text{link}}}{v^{\max}}, \dfrac{1+n_{s,l,i}}{\zeta_{s,l}}\right) & \text{if link } l \text{ out of } S_s \text{ is not jammed} \\ \max\left(\dfrac{d_{s,l}^{\text{link}}}{v^{\text{jam}}}, \dfrac{1+n_{s,l,i}}{\zeta_{s,l}}\right) & \text{if link } l \text{ out of } S_s \text{ is jammed} \end{cases} \tag{4.32}$$

  where

  - $d_{s,l}^{\text{link}}$ is the length of link l out of S_s,
  - $n_{s,l,i}$ is the number of DCVs on link l at time instant $t_{s,i}^{\text{compute\_sw\_out}}$,
  - $\zeta_{s,l}$ is the release rate at time instant $t_{s,i}^{\text{compute\_sw\_out}}$ computed over the time window $[t_{s,i}^{\text{compute\_sw\_out}} - \tau^{\text{rate}}, t_{s,i}^{\text{compute\_sw\_out}})$ (recall from Section 4.4.3 that when

$t_{s,i}^{\text{compute\_sw\_out}} < t_0 + \tau^{\text{rate}}$ we consider $\zeta_{s,l} = \zeta^{\max}$, while if no DCV left link l within the time window $[t_{s,i}^{\text{compute\_sw\_out}} - \tau^{\text{rate}}, t_{s,i}^{\text{compute\_sw\_out}})$ we set $\zeta_{s,l} = \varepsilon$ with $0 < \varepsilon \ll 1$),

– v^{jam} is the speed to be used in case of jam, typically $v^{\text{jam}} = 0.02\,\text{m/s}$. We consider the outgoing link l of S_s to be jammed only if $Q_{s,l} \geq \alpha Q_{s,l}^{\max}$ where $Q_{s,l}$ is the capacity link l at time instant $t_{s,i}^{\text{cross}}$, $Q_{s,l}^{\max}$ is its maximum capacity, and α is a weighting parameter determined based on empirical data (typically $\alpha = 0.8$).

- $\hat{\tau}_r^{\text{route}}$ is the predicted travel time on route $r \in \mathscr{R}_{s,i}^{\text{next}}$ for an average speed determined based on empirical data.

Next we define the cost criterion $c_{s,l,i}^{\text{sw\_out}}$ for $l = 0,1$ that takes into account $J_i(\hat{t}_{s,l,i}^{\text{unload},*})$, where

$$\hat{t}_{s,l,i}^{\text{unload},*} = \underset{\{\hat{t}_{s,l,r,i}^{\text{unload}} | r \in \mathscr{R}_{s,l,i}^{\text{next}}\}}{\arg\min} J_i(\hat{t}_{s,l,r,i}^{\text{unload}}),$$

and the position O_s^{crt} of the outgoing switch at time instant $t_{s,i}^{\text{compute\_sw\_out}}$:

$$c_{s,0,i}^{\text{sw\_out}} = w^{\text{pen}} J_i(\hat{t}_{s,0,i}^{\text{unload},*}) + w^{\text{sw\_out}} \tau^{\text{switch}} O_s^{\text{crt}} \tag{4.33}$$

$$c_{s,1,i}^{\text{sw\_out}} = w^{\text{pen}} J_i(\hat{t}_{s,1,i}^{\text{unload},*}) + w^{\text{sw\_out}} \tau^{\text{switch}} (1 - O_s^{\text{crt}}) \tag{4.34}$$

The last term of $c_{s,l,i}^{\text{sw\_out}}$ for $l = 0,1$ is necessary due to the operational constraint $\mathbf{C}_3$. The weighting parameters w^{pen} and $w^{\text{sw\_out}}$ can be tuned as explained in Section 4.4.6.3.

The position of the switch-out at junction S_s is toggled only if $c_{s,0,i}^{\text{sw\_out}} < c_{s,1,i}^{\text{sw\_out}}$ and $O_s^{\text{crt}} = 1$, or if $c_{s,1,i}^{\text{sw\_out}} < c_{s,0,i}^{\text{sw\_out}}$ and $O_s^{\text{crt}} = 0$. If this is the case, then the switch-out is toggled after

$$\tau_s^{\text{sw\_out}} = \max(0, \tau^{\text{switch}} - \tau_s^{\text{sw\_out\_prev}})$$

where $\tau_s^{\text{sw\_out\_prev}}$ is the time for which the switch-out at junction S_s has been in its current position. Otherwise, we set $\tau_s^{\text{sw\_out}} = \infty$.

The results obtained for this control approach will be illustrated in Section 4.5.2.

4.4.6.3 Tuning the weighting parameters for the heuristic approach

The switch control sequence of each junction depends now also on the weighting parameters $w^{\text{st\_pr}}$, $w^{\text{dyn\_pr}}$, w^{pen}, $w^{\text{sw\_in}}$, $w^{\text{sw\_out}}$ introduced above. Consequently, the total performance index J^{tot} given by (4.4) depends on the weighting parameters.

In Figure 4.18 we have plotted — for the case study[13] presented in [78] and for a typical loading profile[14] — the total performance index J^{tot} as follows. The curve corresponding to e.g. $J_{\text{st\_pr}}^{\text{tot}}$ is obtained when $w^{\text{st\_pr}}$ varies between 10^{-5} and 1 for a discretization step of 0.01 and between 1.1 and 4 for a discretization step of 0.1, while all the other weighting parameters are kept constant, equal to 0.5. Similarly, we have plotted $J_{\text{dyn\_pr}}^{\text{tot}}$, $J_{\text{sw\_in}}^{\text{tot}}$, $J_{\text{pen}}^{\text{tot}}$, and $J_{\text{sw\_out}}^{\text{tot}}$.

[13] In this case study we consider a network of tracks with one loading station, one unloading station, and four junctions.

[14] We consider 200 bags to be handled, their arrival at the loading station being dynamically assigned according to a uniform distribution.

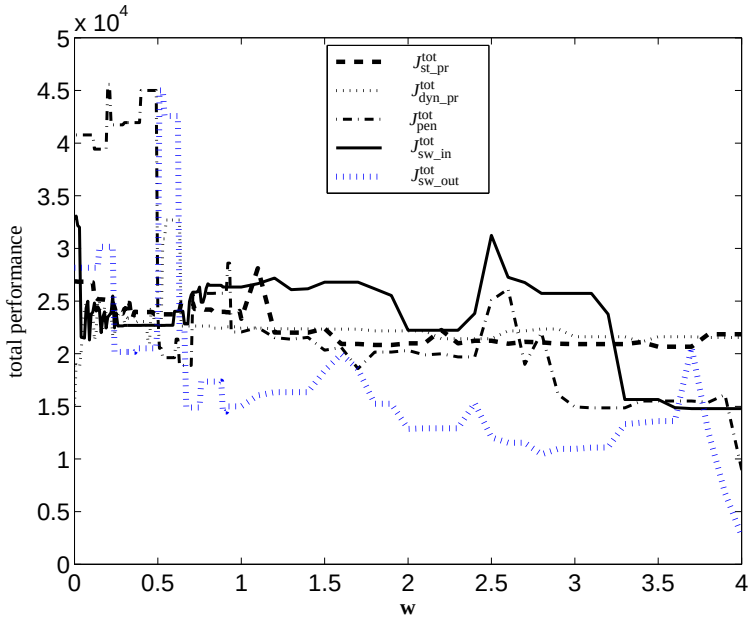


Figure 4.18: Influence of weighting parameters over J^{tot}. The curve corresponding to e.g. $J^{tot}_{st\_pr}$ is obtained when w^{st_pr} varies between 10^{-5} and 1 for a discretization step of 0.01 and between 1.1 and 4 for a discretization step of 0.1, while all the other weighting parameters are kept constant.

As illustrated in Figure 4.18 there are many variations in the amplitude of the total performance index. Therefore, the weighting parameters have to be first calibrated.

In order to calibrate the weighting parameters, we use the event based model of a DCV-based baggage handling system described by **Algorithm 2** of Section 4.2.2 that can be recast as $\mathbf{t} = \mathscr{M}^{\text{switch\_ctrl}}\left(\mathscr{T}, \mathbf{x}(t_0), \mathbf{u}^{\text{heuristic}}\right)$ where $\mathbf{u}^{\text{heuristic}}$ is the heuristic control sequence for the entire network consisting of all the time intervals after which the position of a switch-in and the position of a switch-out at each junction is toggled.

The tuning of the control weighting parameters will be then done by solving off-line the following optimization problem:

$$\begin{aligned}
&\min_{\mathbf{w}} \sum_{j=1}^{N^{\text{scenario}}} J^{tot}_{j,\mathbf{w}}(\mathbf{t}) \\
&\text{subject to} \\
&\qquad \mathbf{t} = \mathscr{M}^{\text{switch_ctrl}}\left(\mathscr{T}, \mathbf{x}(t_0), \mathbf{u}^{\text{heuristic}}\right) \text{ depending on } \mathbf{w} \\
&\qquad \mathscr{C}(\mathbf{t}) \leq 0
\end{aligned}$$

where N^{scenario} is the number of scenarios over which the tuning is performed, $J^{tot}_{j,\mathbf{w}}$ is the total objective function corresponding to scenario j, with $j \in \{1, 2, \ldots, N^{\text{scenario}}\}$, and $\mathbf{w} = [w^{st\_pr}\ w^{dyn\_pr}\ w^{sw\_in}\ w^{pen}\ w^{sw\_out}]^\top$.

The above optimization problem is nonlinear, nonconvex and has continuous variables. So, in order to solve this problem, one can use multi-start local optimization algorithms such

as *sequential quadratic programming* or multi-start global optimization algorithms such as *pattern search*, *simulated annealing* algorithms, or *genetic* algorithms, see e.g. Section 2.1.2.

4.4.7 Distributed heuristic approach

In this subsection we develop an approach where the switch control is performed based on both local information and additional data regarding the flow of DCV on the incoming and outgoing links of the neighboring junctions. This is an extension of the previous decentralized heuristic approach.

4.4.7.1 Control of the switch-in

As in Section 4.4.6.1, we compute based on heuristic rules the control of a switch-in for each junction in the network that has two incoming links (see Figure 4.19). Let S_s with $s \in \{1,2,\ldots,S\}$ be such a junction. The control of the switch-in is updated every time ($t_s^{\text{compute\_sw\_in}}$) some bag enters the incoming links of S_s.

When applying the distributed heuristic approach we compute again the objective functions $p_{s,0}^{\text{sw\_in}}$ of (4.30) and $p_{s,1}^{\text{sw\_in}}$ of (4.31) defined in the previous subsection. However, we now also take into account the bags that will come towards S_s from its neighboring junctions in the next τ^{pred} time units. The period τ^{pred} is determined based on empirical data.

Let $\mathrm{S}_{s,l}^{\text{prev}}$ be the junction connected to S_s via the incoming link $l \in \{0,1\}$ of S_s. We predict which bags will cross $\mathrm{S}_{s,l}^{\text{prev}}$ and continue traveling towards S_s as follows. At time instant $t_s^{\text{compute\_sw\_in}}$ we compute the control sequence for the switch-in and switch-out at $\mathrm{S}_{s,0}^{\text{prev}}$ and $\mathrm{S}_{s,0}^{\text{prev}}$ using the decentralized heuristic rules for switch-in presented above, and the heuristic rules for switch-out presented in the paragraph *Control of the switch-out* of this subsection respectively. As prediction model we use the simulation model of **Algorithm 2** for the time period $[t_s^{\text{compute\_sw\_in}}, t_s^{\text{compute\_sw\_in}} + \tau^{\text{pred}})$. As result of this simulation we determine which bags will cross $\mathrm{S}_{s,l}^{\text{prev}}$ with $l \in \{0,1\}$, and continue traveling towards S_s and at which time they will enter the incoming link l of S_s.

Let $\Omega_{s,l}$ be the set of bags that will cross $\mathrm{S}_{s,l}^{\text{prev}}$ when traveling towards S_s in the next τ^{pred} time units. Then, the time when junction S_s toggles its position is computed as in *Control of the switch-in* of Section 4.4.6. The difference is that here we use the following performance measures:

$$p_{s,0}^{\text{sw\_in}} = w^{\text{st\_pr}}(\rho_{s,0}^{\text{static}} + \varphi_{s,0}^{\text{static}}) + w^{\text{dyn\_pr}}(\rho_{s,0}^{\text{dyn}} + \varphi_{s,0}^{\text{dyn}}) - w^{\text{sw\_in}}\tau^{\text{switch}} I_s^{\text{crt}} \tag{4.35}$$

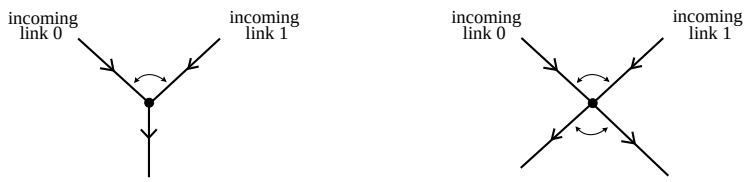


Figure 4.19: Junctions with a switch-in. The time when the next toggle will take place is determined by a local rule-based controller.

$$p_{s,1}^{\text{sw\_in}} = w^{\text{st\_pr}}(\rho_{s,1}^{\text{static}} + \varphi_{s,1}^{\text{static}}) + w^{\text{dyn\_pr}}(\rho_{s,1}^{\text{dyn}} + \varphi_{s,1}^{\text{dyn}}) - w^{\text{sw\_in}}\tau^{\text{switch}}(1 - I_s^{\text{crt}}) \tag{4.36}$$

where

- $\varphi_{s,l}^{\text{static}}$ is total static priority of the bags in $\Omega_{s,l}$, $\varphi_{s,l}^{\text{static}} = \sum_{i \in \Omega_{s,l}} \sigma_i$,
- $\varphi_{s,l}^{\text{dyn}}$ is the total dynamic priority of the bags in $\Omega_{s,l}$, $\varphi_{s,l}^{\text{dyn}} = \sum_{i \in \Omega_{s,l}} \frac{\hat{\delta}_i}{\delta_i^{\max}}$.

4.4.7.2 Control of the switch-out

For each junction which has a switch-out, as the junctions illustrated in Figure 4.20, we update the control of the switch-out as presented next. Let S_s with $s \in \{1, 2, \ldots, S\}$ be such a junction. Then, every time when a bag is positioned just before junction S_s and the switch-in at S_s allows its crossing, we compute (as presented below) the control variable $\tau_s^{\text{sw\_out}}$, which represents the time period until the position of the switch-out has to be changed next. Assume that bag index i is just before junction S_s. Then we update the control of the switch-out at time instant $t_{s,i}^{\text{compute\_sw\_out}}$. The control of the switch-out at junction S_s is computed using a reasoning similar to that in Section 4.4.6.2. However, in this case, when computing the predicted objective function for the outgoing link $l = 0, 1$ and bag index i, we do not look only at the congestion on the outgoing links of junction S_s, but also at the congestion farther (downstream) in the network.

So, we will predict the time that bag index i needs to travel on the next[15] $\nu^{\text{next,max}}$ links when trying to reach its destination where $\nu^{\text{next,max}}$ denotes the maximum number of links we look ahead.

Let us consider next the case where $\nu^{\max} = 2$. As sketched in Figure 4.21, $\mathrm{S}_{s,l,m}^{\text{next}}$ for $m = 0, 1$ denotes the neighboring junction of $\mathrm{S}_{s,l}^{\text{next}}$ connected via link m out of $\mathrm{S}_{s,l}^{\text{next}}$. Then the time period that bag index i needs to travel link m out of $\mathrm{S}_{s,l}^{\text{next}}$ considering the release rate $\zeta_{s,l,m}$ of link m out of $\mathrm{S}_{s,l}^{\text{next}}$ is defined as:

$$\hat{\tau}_{s,l,m,i}^{\text{link}} = \begin{cases} \max\left(\dfrac{d_{s,l,m}^{\text{link}}}{v^{\max}}, \dfrac{1 + \tilde{N}_{s,l,m,i}^{\text{DCV}}}{\zeta_{s,l,m}}\right) & \text{if link } m \text{ out of } \mathrm{S}_{s,l}^{\text{next}} \text{ is not jammed} \\ \max\left(\dfrac{d_{s,l,m}^{\text{link}}}{v^{\text{jam}}}, \dfrac{1 + \tilde{N}_{s,l,m,i}^{\text{DCV}}}{\zeta_{s,l,m}}\right) & \text{if link } m \text{ out of } \mathrm{S}_{s,l}^{\text{next}} \text{ is jammed} \end{cases}$$

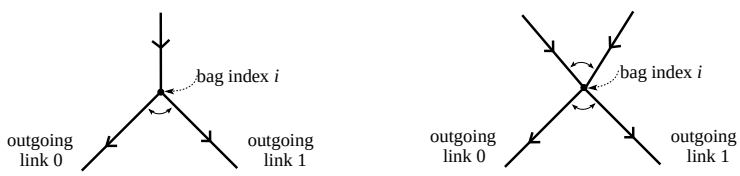


Figure 4.20: Junctions with a switch-out. The time when the next toggle will take place is determined by a local rule-based controller.

[15] We look only at the next $\nu^{\text{next,max}}$ links in order to get some extra information on the network congestion state, while keeping the communication requirements low.

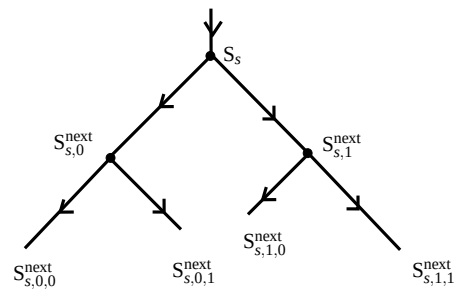


Figure 4.21: Subtree of neighboring junctions for junction S_s.

where

- $d^{\text{link}}_{s,l,m}$ is the length of the link m out of $S^{\text{next}}_{s,l}$,
- $\tilde{N}^{\text{DCV}}_{s,l,m,i}$ represents the number of DCV that we estimate to have on link m out of $S^{\text{next}}_{s,l}$ at the time instant when bag index i will cross f $S^{\text{next}}_{s,l}$

$$\tilde{N}^{\text{DCV}}_{s,l,m,i} = n_{s,l,m,i} + \tilde{n}_{s,l,m,i} - \tilde{n}^{\text{cross}}_{s,l,m,i}$$

  with

  - $n_{s,l,m,i}$ the number of DCVs on link m out of $S^{\text{next}}_{s,l}$ at time instant $t^{\text{compute\_sw\_out}}_{s,i}$.
  - $\tilde{n}_{s,l,m,i}$ the estimated number of DCVs on link l out of S_s that choose link m out of $S^{\text{next}}_{s,l}$. In order to estimate $\tilde{n}_{s,l,m,i}$, we assume that for a junction $S^{\text{next}}_{s,l}$, a fraction $\eta_{s,l}$ of the DCVs crossing $S^{\text{next}}_{s,l}$ take link $m = 0$ out of $S^{\text{next}}_{s,l}$. The fraction $\eta_{s,l}$ is determined based on historical data.
  - $\tilde{n}^{\text{cross}}_{s,l,m,i}$ the estimated number of DCVs that cross $S^{\text{next}}_{s,l,m}$ after traveling on link m out of $S^{\text{next}}_{s,l}$. We define $\tilde{n}^{\text{cross}}_{s,l,m,i}$ as follows:

$$\tilde{n}^{\text{cross}}_{s,l,m,i} = \min\left(n_{s,l,m,i} + \tilde{n}_{s,l,m,i}, \zeta_{s,l,m}\hat{\tau}^{\text{link}}_{s,l,i}\right)$$

    where $\hat{\tau}^{\text{link}}_{s,l,i}$ is the time we predict that bag index i spends on link l out of S_s (see (4.32).

Let $\mathscr{R}^{\text{next}}_{s,l,m,i}$ with $l \in \{0,1\}$ and $m \in \{0,1\}$ denote the set of routes from junction $S^{\text{next}}_{s,l,m}$ to $S^{\text{dest}}_{s,i}$, the end point of bag index i. In this case, for each route $r \in \mathscr{R}^{\text{next}}_{s,l,m,i}$ we predict the time $\hat{t}^{\text{unload}}_{s,l,m,r,i}$ when bag index i will reach $S^{\text{dest}}_{s,i}$ if the bag takes link l out of S_s, link m out of $S^{\text{next}}_{s,l}$, and route r. This time is given by:

$$\hat{t}^{\text{unload}}_{s,l,m,r,i} = t^{\text{compute\_sw\_out}}_{s,i} + \hat{\tau}^{\text{link}}_{s,l,i} + \hat{\tau}^{\text{link}}_{s,l,m,i} + \hat{\tau}^{\text{route}}_{r}$$

where

- $\hat{\tau}^{\text{link}}_{s,l,m,i}$ is the time we predict that bag index i will spend on link m out of $S^{\text{next}}_{s,l}$,

- $\hat{\tau}_r^{\text{route}}$ is the average travel time on route $r \in \mathscr{R}_{s,l,m,i}^{\text{next}}$, determined based on historical data.

Finally, in computing the cost criterion $c_{s,l,i}^{\text{sw\_out}}$ for $l = 0,1$ defined in Section 4.4.6 we use $J_i(\hat{t}_{s,l,i}^{\text{unload},*})$ where $\hat{t}_{s,l,i}^{\text{unload},*}$ is the predicted unloading time that optimizes the objective function of bag index i when choosing link $m \in \{0,1\}$ out of $\mathrm{S}_{s,l}^{\text{next}}$, and route $r \in \mathscr{R}_{s,l,m,i}^{\text{next}}$:

$$\hat{t}_{s,l,i}^{\text{unload},*} = \underset{\{\hat{t}_{s,l,m,r,i}^{\text{unload}} | r \in \mathscr{R}_{s,l,m,i}^{\text{next}}, m \in \{0,1\}\}}{\arg\min} J_i(\hat{t}_{s,l,m,r,i}^{\text{unload}})$$

The analysis of the results obtained for this control approach will be illustrated in Section 4.5.2.

4.4.8 Hierarchical control

In this subsection we propose a hierarchical control framework for DCV-based baggage handling systems. In this control framework switch controllers provide position instructions for each switch in the network. A collection of switch controllers is then supervised by a so-called network controller that mainly takes care of the route choice instructions for DCVs. We will first focus on the route choice control problem for the network controller. Next we will also present the independent, but supervised, switch control.

Control Framework

In order to efficiently compute the route choice of each DCV we propose a hierarchical control framework that consists of a multi-level control structure as shown in Figure 4.22. The layers of the framework can be characterized as follows:

- The *network controller* provides the route choice for DCVs by determining reference flow trajectories over time for each link in the network. These flow trajectories are computed so that the performance of the DCV-based baggage handling system is optimized. Then the optimal reference flow trajectories are communicated to switch controllers.

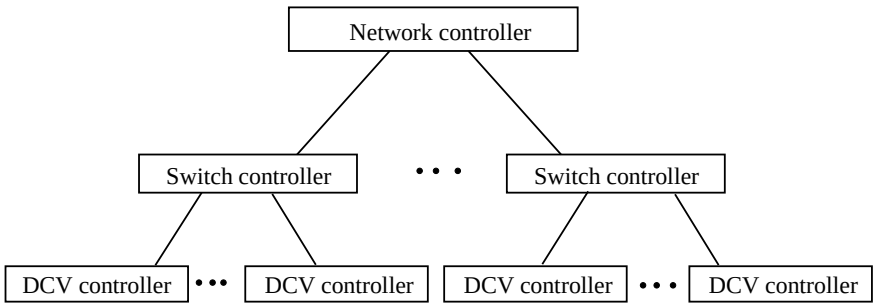


Figure 4.22: Hierarchical control for DCV-based baggage handling systems.

- The *switch controller* present in each junction receives the information sent by the network controller and determines the sequence of optimal positions for its ingoing and outgoing switches at each time step so that the tracking error between the reference flow trajectory and the actual flow trajectory is minimal.

- The *DCV controller* present in each vehicle detects the speed and position of the vehicle in front of it, if any, and the position of the switch into the junction the DCV travels towards to. This information is then used to determine the speed to be used next such that no collision will occur and such that the DCV stops in front of a junction when its ingoing switch is not positioned on the link that the DCV travels on.

The lower levels in this hierarchy deal with faster time scales (typically in the milliseconds range for the DCV controllers up to the seconds range for the switch controllers), whereas for the higher-level layer (network controller) the frequency of updating is up to the minutes range.

In the remainder of this subsection we will focus on the higher-level controllers of the proposed hierarchy and in particular on how the optimal routes can be determined for the DCVs transporting bags through the network.

Approach

In general, the predictive switch control problem in DCV-based baggage handling systems results in a huge nonlinear integer optimization problem with high computational complexity and requirements, making the problem in fact intractable in practice as will be illustrated in Section 4.5. So, since considering each individual switch is too computationally intensive we will consider streams of DCVs instead (characterized by real-valued demands and flows expressed in vehicles per second). The routing problem will then be recast as the problem of determining the flows on each link. Once these flows are determined, they can be implemented by switch controllers at the junctions. So, the network controller provides flow targets to the switch controllers, which then have to control the position of the switch into and out of each junction in such a way that these targets are met as well as possible.

Set-up

We consider the following set-up. We have a transportation network with a set of origin nodes $\mathscr{O}$ consisting of the loading stations, a set of destination nodes $\mathscr{D}$ consisting of the unloading stations, and a set of internal nodes $\mathscr{I}$ consisting of all the junctions in the network, see Figure 4.23. We define the set of all nodes as $\mathscr{V} = \mathscr{O} \cup \mathscr{I} \cup \mathscr{D}$. The nodes are connected by unidirectional links. Let $\mathscr{L}$ denote the set of all links.

Let the time instant t_k be defined as $t_k = k\tau^{\text{nc}}$ with τ^{nc} the sampling time for the network controller. Then, for each pair $(o,d) \in \mathscr{O} \times \mathscr{D}$, there is a dynamic, piecewise constant demand pattern $D_{o,d}(\cdot)$ as shown in Figure 4.24 with $D_{o,d}(k)$ the demand of bags at origin o with destination d in the time interval $[t_k, t_{k+1})$ for $k = 0, \ldots, K-1$ with K the demand horizon (we assume that beyond t_K the demand is 0).

Next, let $\mathscr{L}_d$ be the set of links that belong to some route going to destination d, $\mathscr{L}_d \subseteq \mathscr{L}$. We denote the set of incoming links for node $v \in \mathscr{V}$ by $\mathscr{L}_v^{\text{in}}$, and the set of outgoing links of v by $\mathscr{L}_v^{\text{out}}$. Note that for origins $o \in \mathscr{O}$ we have $\mathscr{L}_o^{\text{in}} = \emptyset$ and for destinations $d \in \mathscr{D}$

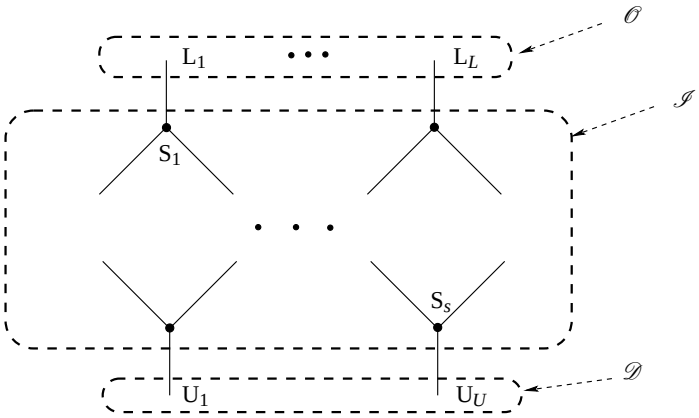


Figure 4.23: Set-up for the DCV-based baggage handling system. The transportation network has a set of origin nodes $\mathscr{O} = \{L_1, L_2, \ldots, L_L\}$, a set of destination nodes $\mathscr{D} = \{U_1, U_2, \ldots, U_U\}$, and a set of internal nodes $\mathscr{I} = \{S_1, S_2, \ldots, S_S\}$.

we have $\mathscr{L}_d^{\text{out}} = \emptyset$. Also, recall from Section 4.2.1 that we have assumed each origin node to have only one outgoing link and each destination node to have only one incoming link (assumption $\mathbf{A}_3$). Then $|\mathscr{L}_o^{\text{out}}| = 1$ and $|\mathscr{L}_d^{\text{in}}| = 1$.

Next, for each destination $d \in \mathscr{D}$ and for each link $\ell \in \mathscr{L}_d$ in the network we will define a real-valued flow $u_{\ell,d}(k)$. The flow $u_{\ell,d}(k)$ denotes the number of DCVs per time unit traveling towards destination d that enter link ℓ during the time interval $[t_k, t_{k+1})$.

The aim is now to compute using MPC, for each time step k, flows $u_{\ell,d}(k)$ for every destination $d \in \mathscr{D}$ and for every link $\ell \in \mathscr{L}_d$ in such a way that the capacity of the links is not exceeded and such that the performance criterion is minimized over a given prediction period $[t_k, t_{k+N})$. Later on we will write a model of the baggage handling system to be used by the network controller, and show that this model can be rewritten as an MILP model. Therefore, in order to obtain an MILP optimization problem one has to define a linear or piecewise affine performance criterion. Possible goals for the network controller that allow linear or piecewise affine performance criteria are reaching a desired outflow at destination d or minimizing the lengths of the queue in the network.

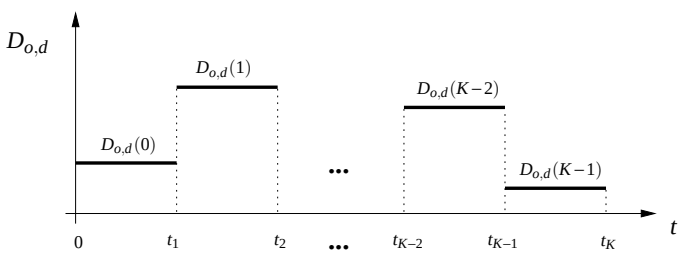


Figure 4.24: Piecewise constant time-varying demand profile $D_{o,d}$.

Model

We now determine the model for the DCV flows through the network. Let τ_ℓ denote the free-flow travel time on link ℓ. Recall that the free-flow travel time of link ℓ represents the time period that a DCV requires to travel on link ℓ when using maximum speed. In this subsection we assume the travel time τ_ℓ to be an integer multiple of τ^{nc}, say

$$\tau_\ell = \kappa_\ell \tau^{\text{nc}} \quad \text{with } \kappa_\ell \text{ an integer.} \tag{4.37}$$

In case the capacity of a loading station is less than the demand, queues might appear at the origin of the network. Let $q_{o,d}(k)$ denote the length at time instant t_k of the partial queue of DCVs at origin o going to destination d. In principle, the queue lengths should be integers as their unit is "number of vehicles", but we will approximate them using reals.

For every origin node $o \in \mathscr{O}$ and for every destination $d \in \mathscr{D}$ we now have:

$$u_{\ell,d}(k) \leqslant D_{o,d}(k) + \frac{q_{o,d}(k)}{\tau^{\text{nc}}} \qquad \text{for } \ell \in \mathscr{L}_o^{\text{out}} \cap \mathscr{L}_d \tag{4.38}$$

with $D_{o,d}(k) = 0$ for $k \geqslant K$. Moreover,

$$q_{o,d}(k+1) = \max\left(0, q_{o,d}(k) + \Big(D_{o,d}(k) - \sum_{\ell \in \mathscr{L}_o^{\text{out}} \cap \mathscr{L}_d} u_{\ell,d}(k)\Big)\tau^{\text{nc}}\right) \tag{4.39}$$

But queues can form also inside the network. We assume that the DCVs run with maximum speed along the track segments and, if necessary, they wait before crossing the junction in vertical queues. Let $q_{v,d}(k)$ denote the length at time instant t_k of the vertical queue at junction $v \in \mathscr{I}$, for DCVs going to destination $d \in \mathscr{D}$. In this subsection we do not consider outflow restrictions on queues to destination d for a junction v connected via a link to destination d, and hence $q_{v,d}(k) = 0$ for all k.

Taking into account that a flow on link ℓ has a delay of κ_ℓ time steps before it reaches the end of the link, for every internal node $v \in \mathscr{I}$ and for every $d \in \mathscr{D}$ we have:

$$F_{v,d}^{\text{out}}(k) \leqslant F_{v,d}^{\text{in}}(k) + \frac{q_{v,d}(k)}{\tau^{\text{nc}}} \tag{4.40}$$

where $F_{v,d}^{\text{in}}(k)$ is the flow into the queue at junction v, being defined as:

$$F_{v,d}^{\text{in}}(k) = \sum_{\ell \in \mathscr{L}_v^{\text{in}} \cap \mathscr{L}_d} u_{\ell,d}(k - \kappa_\ell) \tag{4.41}$$

and where $F_{v,d}^{\text{out}}(k)$ is the flow out of the queue at junction v, defined as:

$$F_{v,d}^{\text{out}}(k) = \sum_{\ell \in \mathscr{L}_v^{\text{out}} \cap \mathscr{L}_d} u_{\ell,d}(k) \ . \tag{4.42}$$

The evolution of the length of the queue for every internal node $v \in \mathscr{I}$ and for every $d \in \mathscr{D}$ is given by:

$$q_{v,d}(k+1) = \max\left(0, q_{v,d}(k) + \big(F_{v,d}^{\text{in}}(k) - F_{v,d}^{\text{out}}(k)\big)\tau^{\text{nc}}\right) \tag{4.43}$$

Moreover, for each origin $o \in \mathscr{O}$ and for each junction $v \in \mathscr{I}$ we have the following constraints:

$$\sum_{d \in \mathscr{D}} q_{o,d}(k+1) \leq q_o^{\max} \tag{4.44}$$

$$\sum_{d \in \mathscr{D}} q_{v,d}(k+1) \leq q_v^{\max} \tag{4.45}$$

where $q_o^{\max}$ and $q_v^{\max}$ express (respectively) the maximum number of DCVs the conveyor belt transporting bags towards loading stations can accommodate and the maximum number of DCVs the track segments of the incoming links of that junction can accommodate.

We also have the following condition for every link ℓ:

$$\sum_{d \in \mathscr{D}} u_{\ell,d}(k) \leqslant U^{\max} \tag{4.46}$$

where $U^{\max}$ is the maximum flow of DCVs that can enter a link.

Then, at time step k, the model of the DCV flows through the network of tracks describing (4.38)–(4.46) can be written as a system of equalities and a system of inequalities as follows:

$$\begin{aligned} \mathbf{q}_{k+1} &= \mathscr{M}^{\text{eq}}(\mathbf{q}_k, \mathbf{u}_k) \\ \mathscr{M}^{\text{ineq}}(\mathbf{q}_{k+1}, \mathbf{u}_k) &\leq 0 \end{aligned}$$

where

- $\mathbf{q}_k$ is the vector consisting of all the queue lengths $q_{o,d}(k)$, for all $o \in \mathscr{O}$ and for all $d \in \mathscr{D}$, and of all the queue lengths $q_{v,d}(k)$, for all $v \in \mathscr{I}$ and for all $d \in \mathscr{D}$
- $\mathbf{u}_k$ is the vector consisting of all the flows $u_{\ell,d}(k)$, for all $d \in \mathscr{D}$ and for all $\ell \in \mathscr{L}_d$.

Performance index

Next we define the performance index to be used for computing the optimal routing at step k for a prediction period of N time steps.

The objective is to have each bag arriving at its end point within a given time interval $[t_d^{\text{close}} - \tau_d^{\text{open}}, t_d^{\text{close}})$ where t_d^{close} is the time instant when the end point d closes and τ_d^{open} is the time period for which the end point d stays open for a specific flight. We assume t_d^{close} and τ_d^{open} to be integer multiples of τ_{s}.

Hence, one MPC objective that allows a piecewise affine performance criterion is to achieve a desired flow at destination d during the prediction period. Let u_d^{desired} denote the desired piecewise constant flow profile at destination d as sketched in Figure 4.25, where the area under u_d^{desired} equals the total number of bags out of the total demand that have to be sent to destination d. Note that $u_d^{\text{desired}}(k) = 0$ for all $k < k_d^{\text{open}}$ and all $k \geq k_d^{\text{close}}$ with $k_d^{\text{open}} = \frac{t_d^{\text{close}} - \tau^{\text{open}}}{\tau^{\text{nc}}}$ and $k_d^{\text{close}} = \frac{t_d^{\text{close}}}{\tau^{\text{nc}}}$.

Let $\kappa_{\ell_d} = \frac{\tau_{\ell_d}}{\tau^{\text{nc}}}$. Hence, one can define the following penalty for flow profiles corresponding to destination $d \in \mathscr{D}$:

$$J_d^{\text{pen}}(k) = u_d^{\text{desired}}(k) - u_{\ell_d,d}(k + \kappa_{\ell_d})$$

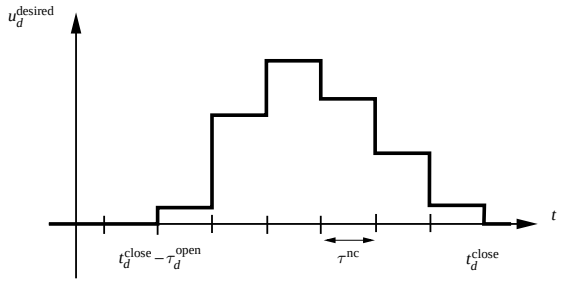


Figure 4.25: Desired arrival profile at destination d.

where ℓ_d is the incoming link of destination d.

Later on we will include the penalty term $\sum_{i=k}^{k+N-1-\kappa_{\ell_d}} J_d^{\text{pen}}(i)$ into the MPC performance criterion for each destination d and for each time step k. Note that we make the summation of these penalization indices only up to $k+N-1-\kappa_{\ell_d}$ since for $i > k+N-1-\kappa_{\ell_d}$ the variable $u_{\ell_d,d}(k+\kappa_{\ell_d})$ is not defined at MPC step k.

Moreover, note that using as MPC performance criterion $\sum_{i=k}^{k+N-1-\kappa_{\ell_d}} J_d^{\text{pen}}(i)$ for each destination d and for each time step k, could have adverse effects for small prediction horizons. Therefore, to counteract these effects, we also consider as additional controller goal maximizing the flows of all links that are not directly connected to unloading stations. To this aim, let $\tau_{\ell,d,k}^{\text{link}}$ be the typical[16] time required for a DCV that entered link ℓ in $[t_k, t_{k+1})$ to reach destination d, with $\tau_{\ell,d,k}^{\text{link}}$ an integer multiple of τ_{s}. Also, let $\kappa_{l,d} = \frac{\tau_{\ell,d,k}^{\text{link}}}{\tau^{\text{nc}}}$. Then one can define the following penalty:

$$J_{\ell,d}^{\text{flow}}(k) = \begin{cases} u_{\ell,d}(k) & \text{if } k_d^{\text{open}} - \kappa_{l,d} \leq k < k_d^{\text{close}} - \kappa_{l,d} \\ 0 & \text{otherwise} \end{cases}$$

This penalty will be later on used in the MPC performance criterion.

Next, in order to make sure that *all* the bags will be handled in finite time, we also include in the MPC performance criterion the weighted length of queues at each junction in the network as presented next. Let $\tau_{v,d}^{\text{junc}}$ be the typical[16] time required for a DCV in the queue at junction v to reach destination d, with $\tau_{v,d}^{\text{junc}}(k)$ an integer multiple of τ^{nc}. Also, let $\kappa_{v,d} = \frac{\tau_{v,d}^{\text{junc}}(k)}{\tau^{\text{nc}}}$. Then we define the new penalty:

$$J_{v,d}^{\text{overdue}}(k) = \begin{cases} d_{v,d}^{\min} q_{v,d}(k) & \text{if } k \geq k_d^{\text{close}} - \kappa_{v,d} \\ 0 & \text{otherwise} \end{cases}$$

where $d_{v,d}^{\min}$ represents the length of the shortest route from junction v to destination d. Note that $J_{v,d}^{\text{overdue}}(k)$ is nonzero only for steps that are larger than or equal to $k_d^{\text{close}} - \kappa_{v,d}$. Moreover, for these steps $J_{v,d}^{\text{overdue}}(k)$ is proportional to $d_{v,d}^{\min}$. The reason for this is that

[16] These durations are determined based on historical data.

we want to penalize more the queues at junctions that are further away from destination d because the DCVs in those queues will need longer time to travel to destination d.

Finally, let $\mathscr{L}^{\text{dest}}$ denote the set of links directly connected to unloading stations. Then the MPC performance index is defined as follows:

$$J_{k,N}(\mathbf{q}_k,\mathbf{u}_k) = \sum_{d\in\mathscr{D}} \Bigg(\lambda_d \sum_{i=k}^{k+N-1-\kappa_{\ell_d}} J_d^{\text{pen}}(i) + \beta \sum_{i=k}^{k+N-1} \sum_{v\in\mathscr{I}} J_{v,d}^{\text{overdue}}(i) - \alpha \sum_{i=k}^{k+N-1} \sum_{\ell\in(\mathscr{L}\setminus\mathscr{L}^{\text{dest}})\cap\mathscr{L}_d} J_{\ell,d}^{\text{flow}}(i) \Bigg) \tag{4.47}$$

with $\lambda_d > 0$ a weight that expresses the importance of the flight assigned to destination d, $\alpha \ll 1$ and $\beta \ll 1$ nonnegative weighting parameters.

Then the nonlinear MPC optimization problem is defined as follows:

$$\begin{aligned}
&\min_{\mathbf{u}_k,\ldots,\mathbf{u}_{k+N-1},\mathbf{q}_{k+1},\ldots,\mathbf{q}_{k+N}} J_{k,N}(\mathbf{q}_k,\mathbf{u}_k)\\
&\quad\text{subject to}\\
&\qquad \mathbf{q}_{k+1} = \mathscr{M}^{\text{eq}}(\mathbf{q}_k,\mathbf{u}_k)\\
&\qquad \vdots\\
&\qquad \mathbf{q}_{k+N} = \mathscr{M}^{\text{eq}}(\mathbf{q}_{k+N-1},\mathbf{u}_{k+N-1})\\
&\qquad \mathscr{M}^{\text{ineq}}(\mathbf{q}_{k+1},\mathbf{u}_k) \le 0\\
&\qquad \vdots\\
&\qquad \mathscr{M}^{\text{ineq}}(\mathbf{q}_{k+N},\mathbf{u}_{k+N-1}) \le 0
\end{aligned}$$

MILP optimization problem for the network controller

Hence, we deal with a nonlinear, nonconvex, and nonsmooth optimization problem. However, using the properties **P1** and **P2** presented in Section 4.4.5, this problem can also be written as an MILP problem.

In principle, — i.e., when an MILP optimization algorithm is not terminated prematurely due to time or memory limitations, — the MILP optimization algorithm guarantees to find the global optimum. This global optimization feature is not present in the other optimization methods that can be used to solve the original nonlinear, nonconvex, nonsmooth route choice optimization problem. Moreover, if the computation time is limited (as is often the case in on-line real-time control), then it might occur that the MILP solution can be found within the allotted time whereas the global and multi-start local optimization algorithm still did not converge to a good solution. As a result, the MILP solution may even give a better system performance than the solution returned by the prematurely terminated global and multi-start local optimization method.

Switch control

We now focus on the switch controller for the proposed hierarchy, and on how optimal switch positions can be determined.

Recall that at each control step k, the network controller provides optimal flows for each link in the network and for each destination. Let these flows be denoted by $u^{\text{opt}}_{\ell,d}(k),\dots,$ $u^{\text{opt}}_{\ell,d}(k+N-1)$ with $d \in \mathscr{D}$, $\ell \in \mathscr{L} \cap \mathscr{L}_d$ and N the prediction horizon of the network controller. Then the switch controller of each junction has to compute optimal switch-in and switch-out positions such that the tracking error between the reference optimal flow trajectory and the flow trajectory obtained by the switch controller is minimal for each network controller time step $k = 0,\dots,K^{\text{sim}}$.

Recall that the optimal flows $u^{\text{opt}}_{\ell,d}(k),\dots,u^{\text{opt}}_{\ell,d}(k+N-1)$ are determined for the time window $[t_k, t_{k+N})$ with $t_k = t_0 + k\tau^{\text{nc}}$. Moreover, note that in order to determine the switch control action during the time window $[t_k, t_{k+N})$ we will use again MPC. Next we will refer to one junction $v \in \mathscr{I}$ only. For all other junctions, the switch control actions are determined similarly.

Let τ^{sc} be the switch controller sampling[17] time. Also, let k^{sc} be an integer that expresses the number of switch control actions determined until now. At t_k, k^{sc} is defined as $k^{\text{sc}} = \frac{\tau^{\text{nc}}}{\tau^{\text{sc}}}k$. Then let $t^{\text{sw}}_{k^{\text{sc}}}$ denote the time instant corresponding to the time step k^{sc} of the switch controller, $t^{\text{sw}}_{k^{\text{sc}}} = t_0 + k^{\text{sc}}\tau^{\text{sc}}$ with t_0 the time instant when we start the simulation.

Furthermore, let $s^{\text{in}}_v(k^{\text{sc}})$ denote the position of the switch-in at junction v during the time interval $\left[t^{\text{sw}}_{k^{\text{sc}}}, t^{\text{sw}}_{k^{\text{sc}}+1}\right)$ and let $s^{\text{out}}_v(k^{\text{sc}})$ denote the position of the switch-out at junction v during $\left[t^{\text{sw}}_{k^{\text{sc}}}, t^{\text{sw}}_{k^{\text{sc}}+1}\right)$.

We want to determine the switch control sequence during the time window $[t_k, t_{k+N})$ while using MPC with prediction period of N^{sc} steps. Hence, at each MPC step k^{sc}, the switch controller solves the following optimization problem:

$$\min_{\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}} \quad J^{\text{sw}}_{v,k^{\text{sc}},N^{\text{sc}}}\left(\mathbf{x}_{v,k^{\text{sc}}}, \mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}\right) \tag{4.48}$$

with $\mathbf{x}_{v,k^{\text{sc}}}$ the current local state at junction v, $\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}} = [s^{\text{in}}_v(k^{\text{sc}}) \dots s^{\text{in}}_v(k^{\text{sc}}+N^{\text{sc}}-1) \dots$ $s^{\text{out}}_v(k^{\text{sc}}) \dots s^{\text{out}}_v(k^{\text{sc}}+N^{\text{sc}}-1)]^\top$ if junction v has 2 incoming and 2 outgoing links ($\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}$ contains only switch-in or only switch-out positions if junction v has only 1 outgoing or only 1 incoming link respectively) and with $J^{\text{sw}}_{v,k^{\text{sc}},N^{\text{sc}}}$ the local MPC performance index defined as:

$$\begin{aligned} J^{\text{sw}}_{v,k^{\text{sc}},N^{\text{sc}}}\left(\mathbf{x}_{v,k^{\text{sc}}}, \mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}\right) = \sum_{\ell \in \mathscr{L}^{\text{out}}_v} & \left| X^{\text{opt}}_{\ell,k,k^{\text{sc}},N^{\text{sc}}}(\mathbf{u}^{\text{opt}}_\ell) - X_{\ell,k^{\text{sc}},N^{\text{sc}}}\left(\mathbf{x}_{v,k^{\text{sc}}}, \mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}\right)\right| \\ & + \gamma\left(n^{\text{sw\_in}}_{k^{\text{sc}},N^{\text{sc}}}\left(\mathbf{x}_{v,k^{\text{sc}}}, \mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}\right) + n^{\text{sw\_out}}_{k^{\text{sc}},N^{\text{sc}}}\left(\mathbf{x}_{v,k^{\text{sc}}}, \mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}\right)\right) \end{aligned}$$

where

- $X^{\text{opt}}_{\ell,k,k^{\text{sc}},N^{\text{sc}}}(\mathbf{u}^{\text{opt}}_\ell)$ denotes the optimal number of DCVs to enter the outgoing link ℓ of junction v during the period $\left[t^{\text{sw}}_{k^{\text{sc}}}, t^{\text{sw}}_{k^{\text{sc}}+N^{\text{sc}}-1}\right)$, where $\mathbf{u}^{\text{opt}}_\ell$ is the vector consisting of all the flows $u^{\text{opt}}_{\ell,d}(k),\dots,u^{\text{opt}}_{\ell,d}(k+N)$ with $d \in \mathscr{D}$ and $\ell \in \mathscr{L} \cap \mathscr{L}_d$. The variable $X^{\text{opt}}_{\ell,k,k^{\text{sc}},N^{\text{sc}}}(\mathbf{u}^{\text{opt}}_\ell)$ is derived later on (see (4.49)).
- $X_{\ell,k^{\text{sc}},N^{\text{sc}}}\left(\mathbf{x}_{v,k^{\text{sc}}}, \mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}\right)$ is the actual number of DCVs entering link ℓ during the prediction period. Given the current state of local system and the sequence of switch

[17] We select the sampling time τ^{nc} of the network controller and the sampling time τ^{sc} of the switch controller such that τ^{nc} is an integer multiple of τ^{sc}.

control, the variable $X_{\ell,k^{\mathrm{sc}},N^{\mathrm{sc}}}$ is determined via simulation for a nonlinear (event-based) model similar to the one of Tarău et al. [80] (the difference is that now the switch positions $\mathbf{s}_{v,k^{\mathrm{sc}},N^{\mathrm{sc}}}$ are given for each period $[t^{\mathrm{sw}}_{k^{\mathrm{sc}}}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+1})$, ..., $[t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}})$ instead of for each of the next N^{sc} DCVs to cross a junction).

- $n^{\mathrm{sw\_in}}_{k^{\mathrm{sc}},N^{\mathrm{sc}}}(\mathbf{x}_{v,k^{\mathrm{sc}}}, \mathbf{s}_{v,k^{\mathrm{sc}},N^{\mathrm{sc}}})$ and $n^{\mathrm{sw\_out}}_{k^{\mathrm{sc}},N^{\mathrm{sc}}}(\mathbf{x}_{v,k^{\mathrm{sc}}}, \mathbf{s}_{v,k^{\mathrm{sc}},N^{\mathrm{sc}}})$ represent the number of toggles of the switch-in and of the switch-out respectively during the prediction window $\left[t^{\mathrm{sw}}_{k^{\mathrm{sc}}}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}}\right)$. These variables are obtained from simulation (for the given the current state of local system and the sequence of switch control).

- γ is a nonnegative weighting parameter.

Next we derive the variable $X^{\mathrm{opt}}_{\ell,k,k^{\mathrm{sc}},N^{\mathrm{sc}}}(\mathbf{u}^{\mathrm{opt}}_{\ell})$. To this aim, we first determine how many steps $p_{k^{\mathrm{sc}}}$ of the network controller will be involved in solving (4.48) as follows: $p_{k^{\mathrm{sc}}} = \left\lceil \frac{N^{\mathrm{sc}}\tau^{\mathrm{sc}}}{\tau^{\mathrm{nc}}} \right\rceil$ where $\lceil x \rceil$ denotes the smallest integer larger than or equal to x (so, $p_{k^{\mathrm{sc}}} \geq 1$). Furthermore, note that the index k of the time instant t_k for which $t_k \leq t^{\mathrm{sw}}_{k^{\mathrm{sc}}} < t_{k+1}$ can be computed as follows: $k = \left\lfloor \frac{k^{\mathrm{sc}}\tau^{\mathrm{sc}}}{\tau^{\mathrm{nc}}} \right\rfloor$ where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Figure 4.26 illustrates the prediction window $\left[t^{\mathrm{sw}}_{k^{\mathrm{sc}}}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1}\right)$ with respect to the window $\left[t_k, t_{k+p_{k^{\mathrm{sc}}}}\right)$.

The variable $X^{\mathrm{opt}}_{\ell,k,k^{\mathrm{sc}}}(\mathbf{u}^{\mathrm{opt}}_{\ell})$ is given by:

$$\begin{aligned} X^{\mathrm{opt}}_{\ell,k,k^{\mathrm{sc}},N^{\mathrm{sc}}}(\mathbf{u}^{\mathrm{opt}}_{\ell}) = & \tau^{\mathrm{left}}_{1,k^{\mathrm{sc}}} \sum_{d\in\mathscr{D}} u^{\mathrm{opt}}_{\ell,d}(k) + \tau^{\mathrm{nc}} \sum_{i=k+1}^{k+p_{k^{\mathrm{sc}}}-2} \sum_{d\in\mathscr{D}} u^{\mathrm{opt}}_{\ell,d}(i) + \\ & \tau^{\mathrm{left}}_{2,k^{\mathrm{sc}}} \sum_{d\in\mathscr{D}} u^{\mathrm{opt}}_{\ell,d}(k+p_{k^{\mathrm{sc}}}-1) \end{aligned} \tag{4.49}$$

where $\sum_{i=k+1}^{k+j} x(i) = 0$ by definition for $j < 1$ and where

$$\begin{aligned} \tau^{\mathrm{left}}_{1,k^{\mathrm{sc}}} &= \min(t_{k+1}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1}) - t^{\mathrm{sw}}_{k^{\mathrm{sc}}}, \\ \tau^{\mathrm{left}}_{2,k^{\mathrm{sc}}} &= \begin{cases} t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1} - t_{k+p_{k^{\mathrm{sc}}}-1} & \text{if } p_{k^{\mathrm{sc}}} > 1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The results obtained when using the hierarchical control framework are presented in Section 4.5.4.

4.5 Experimental results

In this section we present the experimental results obtained when determining the optimal DCV route choice of a baggage handling system by using the control methods proposed

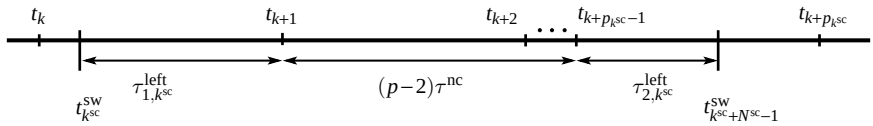


Figure 4.26: Prediction window $\left[t^{\mathrm{sw}}_{k^{\mathrm{sc}}}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1}\right)$ over which we solve the MPC optimization problem (4.48) illustrated with respect to the window $\left[t_k, t_{k+p_{k^{\mathrm{sc}}}}\right)$ for $p_{k^{\mathrm{sc}}} > 2$.

in Section 4.4. Hence, we now assess the efficiency of the following approaches: optimal control, centralized, decentralized, and distributed MPC, decentralized and distributed heuristics — these approaches are developed for a 1-level route choice control framework —, and MPC in a 2-level route choice control framework. In order to reduce the computational complexity of MPC, we have also compared the results obtained when solving nonlinear optimization problems, and when solving MILP optimizations.

4.5.1 Optimal control versus model predictive control

In this subsection we compare the results (the system performance and the total computation time required to determine the route choice control) obtained when using the optimal control method presented in Section 4.4.1 and when using the model predictive control (MPC) approach presented in Section 4.4.2. The comparison will be made based on simulation results.

Set-up

We consider the network of tracks depicted in Figure 4.27 with two loading stations L_1 and L_2, two unloading stations U_1 and U_2, and two junctions S_1 and S_2. We have considered this simple network since the computational complexity of these methods increases with the number of junctions in the network. Note that the control approaches considered in this section allow the choice of routes containing loops.

We assume that the velocity of each DCV varies between $0\,\mathrm{m/s}$ and $20\,\mathrm{m/s}$. The lengths of the track segments are indicated in Figure 4.27.

Scenarios

We have defined fifteen scenarios where the stream of bags that will enter the network of tracks after t_0 has the length $N^{\mathrm{bags}} = 10, 20, 30, \ldots, 150$, the destination of each bag being randomly assigned using a uniform distribution. Recall from Section 4.1.1 that the 2-tuple $\mathscr{T} = (\mathbf{t}_1^{\mathrm{arrival}}, \mathbf{t}_2^{\mathrm{arrival}})$ comprises the vectors of bag arrival times defined as $\mathbf{t}_\iota^{\mathrm{arrival}} = [t_{\iota,1}^{\mathrm{arrival}} \ldots t_{\iota,N_\iota^{\mathrm{load}}}^{\mathrm{arrival}}]^\top$ with N_ι^{load} the number of bags to be loaded onto DCVs at loading station L_ι. According to these scenarios $N_\iota^{\mathrm{load}} = 5, 10, 15, \ldots, 75$ for $\iota = 1, 2$. The arrival times $t_{\iota,1}^{\mathrm{arrival}}, \ldots, t_{\iota,N_\iota^{\mathrm{load}}}^{\mathrm{arrival}}$ at loading station L_ι with $\iota \in \{1, 2\}$, are allocated randomly during the time interval $[t_0, t_0 + 100\,\mathrm{s})$, using a uniform distribution.

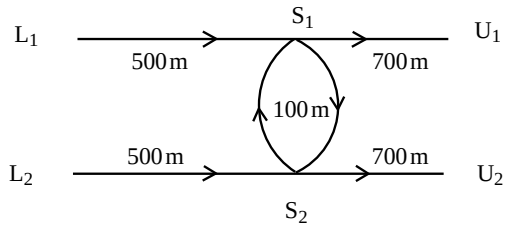


Figure 4.27: Case study for a DCV-based baggage handling system of Section 4.5.1.

For this case study we assume that both end points U_1 and U_2 are open during the time window $[t_0+100\,\text{s}, t_0+200\,\text{s})$. Also, we assume that at time instant t_0, no DCV is transporting bags through the network, while all the considered scenarios start from the same initial state of the system.

Results

To solve the route choice optimization problems of optimal control and MPC, we have chosen the *genetic* algorithm with multiple runs and "bitstring" population, implemented via the function `ga` of the Matlab optimization toolbox *Genetic Algorithm and Direct Search*. We made this choice since simulations show that this optimization technique gives good performance, with a short computation time.

Regarding the optimal control optimization, we set $N^{\text{run}} = 6 + \frac{N^{\text{bags}}}{5}$ and $N^{\text{gen}} = N^{\text{bags}} \cdot 40$ where N^{run} is the number of times that we run the genetic algorithm while solving the optimization problem, and N^{gen} is the maximum number of generations of population that we allow when computing a solution. All the other options are set by using the default options of the function `ga`. Note that we have chosen these large values for N^{run} and N^{gen} since this should increase the chance of finding a solution that is close to the global one, see, e.g., Section 2.1.2.

For the MPC approach we set the horizon to $N = 10$ bags. Recall from Section 4.4.2 that, at each MPC step, we implement all the computed control samples, and accordingly we shift the horizon with N steps. When solving the MPC optimization problem with `ga`, we set $N^{\text{run}} = 3$ and $N^{\text{gen}} = 70$. Note that for MPC N^{run} and N^{gen} have smaller values than the ones we have considered for optimal control. This choice has been made since MPC solves smaller optimization problems than optimal control.

Based on simulations we now compare, for the given scenarios, the proposed control methods. Let $J^{\text{tot,approach}}$ denote the total performance index expressed by (4.4) that corresponds to the control methods that we compare in this subsection (optimal control and MPC). In Figure 4.28 we have plotted the performance index $J^{\text{tot,approach}}$ versus the number of bags that we handle in each scenario, and the total computation time[18] needed for the total closed-loop simulation. Note that the lower the performance index $J^{\text{tot,approach}}$ is, the better the performance of the baggage handling system is, e.g if the baggage handling system has transported all the bags to their end points within the given time window, then no penalization is needed. Hence, the best performance of the system would correspond to $J^{\text{tot,approach}} = 0$.

The results indicate that optimal control gives a better system performance than MPC or as good as MPC only for $N^{\text{bags}} \leq 60$, while for $N^{\text{bags}} > 60$ MPC performs better than optimal control. This happens because the values set for the number of runs N^{run} and for the number of population generations N^{gen} are too low. This means that in order to obtain better results when using optimal control, one has to increase even more N^{run} and N^{gen}. The results also indicate, that even for the computational restrictions mentioned above, the total computation time obtained when computing the route choice solution using optimal control, is always larger than the one obtained when using MPC. Moreover, the difference in computational effort required by the considered approaches increases with the number of bags to be handled by the DCV-based baggage handling system. In particular, the total

[18] The simulations were performed on a 3.0 GHz P4 with 1 GB RAM.

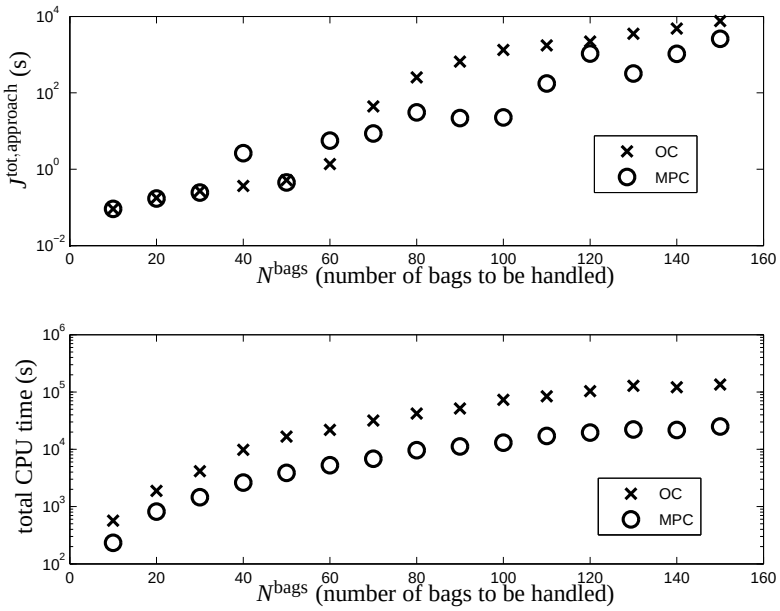


Figure 4.28: Comparison of the results obtained using optimal control (OC) and model predictive control (MPC) for the total closed-loop simulation. In order to assess the efficiency of these approaches, we progressively increase the number of bags to be handled.

computation time of MPC is about 2.7 hours, while the total computation time of optimal control is about 40 hours for $N^{\text{bags}} = 150$.

In practice, optimal control is not suitable for computing the route choice control of a DCV-based baggage handling system. The first reason to support this remark is that, in practice, the arrival time at loading stations of all the bags to be handled is not known at time instant t_0. The second reason, is the huge amount of computational effort required to compute the optimal route choice control. However, even if MPC could be used to compute on-line the DCV route choice control because this method only looks ahead at a buffer of bags to be handled, the total computation time is still quite high for real-time optimizations.

4.5.2 Centralized, decentralized, and distributed control approaches

In this subsection we compare the performance of the centralized, decentralized, and distributed MPC, and the decentralized and distributed heuristics based on simulation examples. These control approaches have been presented in Section 4.4.2, Section 4.4.3, Section 4.4.4, Section 4.4.6, and Section 4.4.7 respectively.

Set-up

We consider the network of tracks depicted in Figure 4.29 with four loading stations, two unloading station, nine junctions, and twenty unidirectional links. Note that this network al-

lows more than four possible routes to each destination from any origin point (e.g., U_1 can be reached from L_1 via junctions S_1, S_4, S_8; S_1, S_4, S_8, S_9, S_8; S_1, S_2, S_5, S_4, S_8; $S_1, S_2, S_5, S_6, S_5, S_4, S_8$; $S_1, S_2, S_6, S_7, S_9, S_8$, and so on). We consider this network because on the one hand it is simple, allowing an intuitive understanding of and insight in the operation of the system and the results of the control[19], and because on the other hand, it also contains all the relevant elements of a real set-up.

We assume that the velocity of each DCV varies between $0\,\mathrm{m/s}$ and $v^{\max} = 20\,\mathrm{m/s}$, and that the minimum time period after we allow a switch toggle is $\tau^{\mathrm{switch}} = 2\,\mathrm{s}$. The lengths of the track segments are indicated in Figure 4.29.

In order to faster assess the efficiency of our control method we assume that we do not start with an empty network but with a network already populated by DCVs transporting bags.

Scenarios

For the tuning of the weighting parameters $\mathbf{w}$ we define eighteen scenarios where 120 bags will be loaded into the baggage handling system (30 bags at each loading station). We consider three classes of demand profiles called "$\mathbf{dp}_1$", "$\mathbf{dp}_2$", and "$\mathbf{dp}_3$" hereafter. According to these classes, the bags arrive at each loading station in the time interval $[t_0, t_0 + 100\,\mathrm{s})$ with t_0 the time instant when we start the simulation. The arrival times at a loading station are allocated randomly, using a uniform distribution according to the following cases:

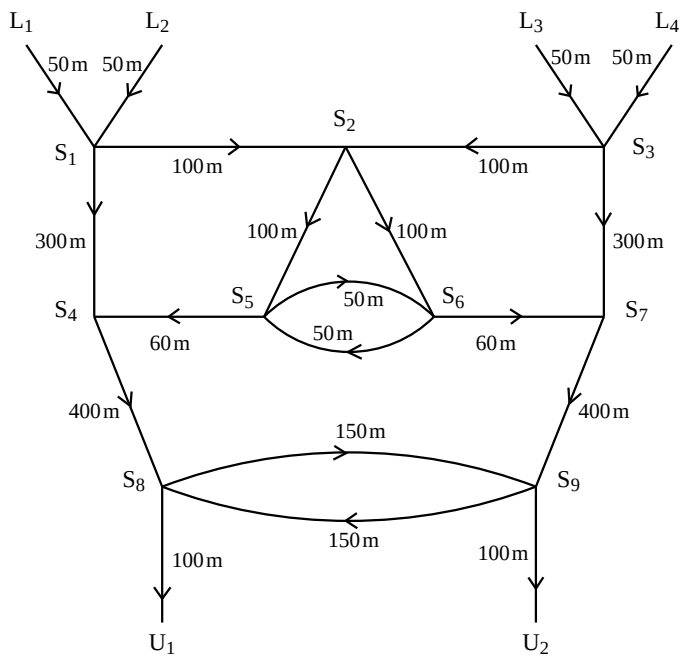


Figure 4.29: Case study for a DCV-based baggage handling system of Section 4.5.2.

[19]The proposed control approaches allow the choice of routes containing loops.

Table 4.1: Considered Scenarios

Scenario type	End time	x_0	Demand	Scenario ID
Relaxed	same end time	**Init**$_1$	**dp**$_1$	1
			dp$_2$	2
			dp$_3$	3
	$t_1^{\text{close}} = 200\,\text{s}$ $t_2^{\text{close}} = 200\,\text{s}$	**Init**$_2$	**dp**$_1$	4
			dp$_2$	5
			dp$_3$	6
Relaxed	different end time	**Init**$_1$	**dp**$_1$	7
			dp$_2$	8
			dp$_3$	9
	$t_1^{\text{close}} = 100\,\text{s}$ $t_2^{\text{close}} = 200\,\text{s}$	**Init**$_2$	**dp**$_1$	10
			dp$_2$	11
			dp$_3$	12
Tight	different end time	**Init**$_1$	**dp**$_1$	13
			dp$_2$	14
			dp$_3$	15
	$t_1^{\text{close}} = 100\,\text{s}$ $t_2^{\text{close}} = 144\,\text{s}$	**Init**$_2$	**dp**$_1$	16
			dp$_2$	17
			dp$_3$	18

dp$_1$**:** the bags arrive at the loading station with a constant rate of 1.2 bags/s;

dp$_2$**:** 5 bags arrive at a loading station during each of the time intervals $[t_0, t_0 + 40\,\text{s})$ and $[t_0 + 60\,\text{s}, t_0 + 100\,\text{s})$, and the rest of 110 bags arrives during $[t_0 + 40\,\text{s}, t_0 + 60\,\text{s})$;

dp$_3$**:** 10 bags arrive during the time interval $[t_0, t_0 + 80\,\text{s})$ and the rest of the bags, i.e., 110 bags, arrives after $t = t_0 + 80\,\text{s}$, i.e., during $[t_0 + 80\,\text{s}, t_0 + 100\,\text{s})$.

Note that for the demand profiles "**dp**$_2$" and "**dp**$_3$" the bags arrival time is uniformly distributed over the mentioned time intervals.

More specifically, we consider two different initial states of the system called "**Init**$_1$" and "**Init**$_2$", where 60, and respectively 120 DCVs are already transporting bags in the network, running from loading stations $\text{L}_1, \ldots, \text{L}_4$ to junctions S_1 and S_2, from S_1 to S_2, and from S_3 to S_2. Their positions at t_0 and their static priorities are assigned randomly.

The bags to be handled can be organized in two groups of bags. Let "group 1" consist of the bags that populate the DCV network before t_0 and "group 2" consist of the bags that enter the network after t_0. For a maximum storage period of 100 s at unloading stations, we examine both situations where the transportation of the bags is very tight (the last bag that enters the system can only arrive in time at its end point if the DCV travels the shortest route with maximum speed), and respectively more relaxed. For $t_0 = 0\,\text{s}$ we denote the scenarios according to Table 4.1 where t_1^{close} and t_2^{close} indicate the time when the end point closes for "group 1" and "group 2" respectively.

We first calibrate the weighting parameters **w** over all the scenarios we have considered. Next, for the same scenarios we compare the control methods, but now we consider different samples of the demand profiles than those used for calibration.

Results

To solve the MPC optimization problems we have chosen again the *genetic* algorithm with multiple runs and "bitstring" population of the Matlab optimization toolbox *Genetic Algorithm and Direct Search*.

Based on simulations we now compare, for the given scenarios, the proposed control methods. For all the proposed predictive control methods we set the horizon to $N = 5$ bags. We make this choice since for a larger horizon, the computation time required to obtain a good solution of the local optimization problem increases substantially. Hence, using larger horizons for the considered MPC optimization problems, yields a considerable larger total computation time.

Let $J_j^{\text{tot,approach}}$ denote the performance index of the baggage handling system corresponding to scenario index j and the considered control approach. In Figure 4.30 we plot the total performance index $J^{\text{tot,approach}}$ and the total computation time[20] obtained when using the proposed control approaches — centralized, decentralized, and distributed MPC, decentralized and distributed heuristics — versus the considered scenarios. In this figure we plot the performance index $J^{\text{tot,approach}}$ corresponding to centralized MPC, but only for the scenarios where the initial population of the network of tracks is small (60 DCVs). We do this since the computation time for the case where the network is populated with 120 DCVs is larger than 10^6 s. Recall that the lower the performance index $J^{\text{tot,approach}}$ is, the better the performance of the baggage handling system.

Let $J^{\text{approach,avg}}$ denote the average performance index obtained when using the predictive control methods and the heuristic approaches. This performance index is defined over the scenarios for which we have plotted the results of Figure 4.30:

$$J^{\text{approach,avg}} = \frac{1}{|\Delta^{\text{approach}}|} \sum_{j \in \Delta^{\text{approach}}} J_j^{\text{tot,approach}}$$

with Δ^{approach} the set of scenarios for which we have illustrated the performance index $J^{\text{tot,approach}}$ in Figure 4.30, e.g., $\Delta^{\text{approach}} = \{1,2,3,7,8,9,13,14,15\}$ for centralized MPC and $\Delta^{\text{approach}} = \{1,2,\ldots,18\}$ for all the other approaches. Then in Table 4.2 we list the average results $J^{\text{approach,avg}}$ of Figure 4.30.

One expects that the best performance of the system is obtained when using *centralized* predictive switch control. This would have happened if we had allowed more runs and if we had allowed a larger computation time to calculate the solution of an MPC optimization problem (in these simulations, in order to reduce the computational effort of the route choice control using centralized MPC, we ran the *genetic* algorithm 4 times for each optimization problem, while limiting the time allowed to compute a solution to 400 s). Moreover, note that centralized control becomes intractable in practice when the number of junctions is large due to the high computation time required.

The simulation results indicate that using *decentralized MPC* lowers the computation time. Furthermore, the results indicate that *distributed MPC* gives better performance than decentralized MPC (in many cases $J^{\text{tot,approach}}$ is much lower than the rest), but at the cost of higher computational effort. Note that when using decentralized and distributed MPC we ran the *genetic* algorithm 4 times for each local optimization problem, while allowing a

[20] The simulations were performed on a 3.0 GHz P4 with 1 GB RAM.

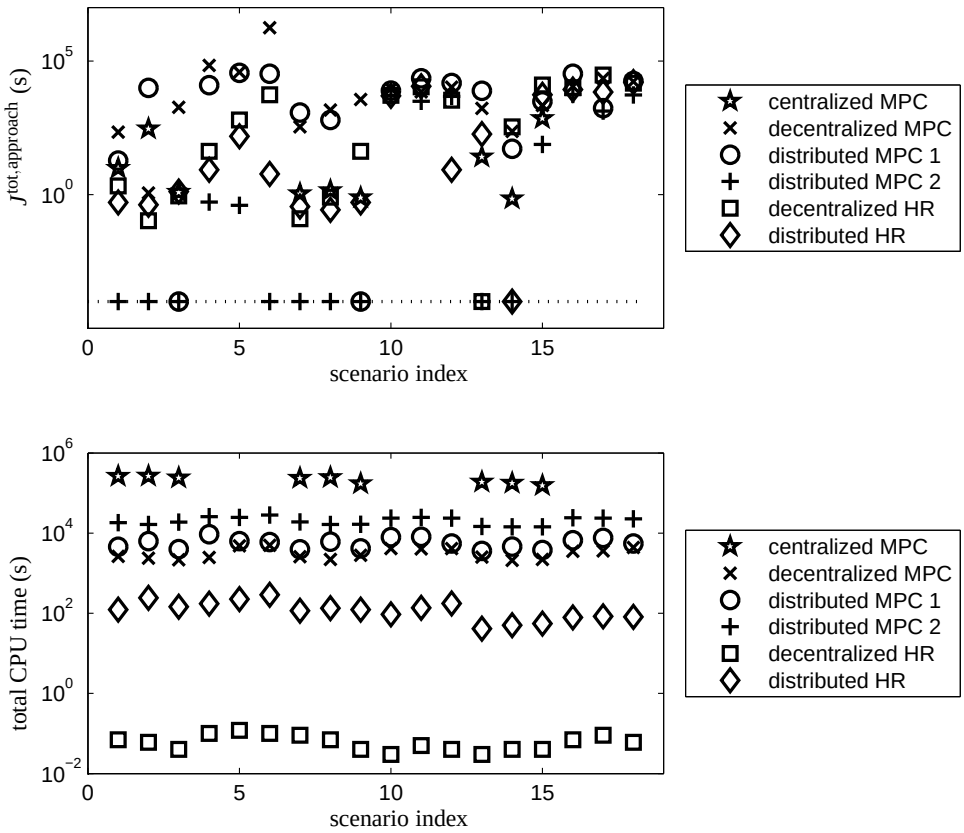


Figure 4.30: Comparison of the results obtained using the proposed centralized, decentralized, and distributed control approaches (MPC and heuristics) for the total closed-loop simulation. In order to visualize on the logarithmic scale results such as $J^{\text{tot,approach}} = 0$ for some scenario, we set $J^{\text{tot,approach}} = 10^{-4}$ for that scenario. The scenarios over which we make this comparison have been defined in Section 4.5.2.

maximum of 20 generations of population. We have chosen these options in order to have a balance between the overall performance and the total computation time.

Finally, the *decentralized* and *distributed heuristic approaches* give typically worse results than distributed MPC with a single round of downstream and upstream communication, but with very low computation time.

Let τ^{handle} denote the length of the time period in which we handle all the bags. According to simulations $110\,\text{s} \leq \tau^{\text{handle}} \leq 240\,\text{s}$ for all scenarios and all proposed methods. Hence, according to the present implementation, only the decentralized and distributed heuristic approaches would give real-time results. However, note that one can easily gain several orders of magnitude in the total computation time of the proposed control approaches by using parallel computation when solving an optimization problem, better implementation, object

Table 4.2: Comparison of average performance of the system and total computation time for the proposed control methods.

Control approach	$J^{\text{approach,avg}}$ (s)	total CPU time (s)
Centralized MPC	$1.16 \cdot 10^2$	$2.2 \cdot 10^5$
Decentralized MPC	$1.10 \cdot 10^5$	$3.2 \cdot 10^3$
Distributed MPC downstream communication	$1.13 \cdot 10^4$	$5.8 \cdot 10^3$
Distributed MPC communication back & forth	$1.4 \cdot 10^3$	$2.0 \cdot 10^4$
Decentralized heuristics	$5.24 \cdot 10^3$	0.06
Distributed heuristics ($\tau^{\text{pred}} = 5\,\text{s}$)	$3.01 \cdot 10^3$	131.48

coded programming languages instead of Matlab, or dedicated optimization algorithms.

4.5.3 Switch control using mixed integer linear programming

In this subsection we compare the results obtained when using the nonlinear and MILP formulation for the optimization problem of centralized MPC. These formulations have been presented in Section 4.4.5.

Set-up

We are interested in analyzing the trade-off between performance and computation time when using the two formulations of the MPC optimization problem. To this aim we consider as benchmark case study the network depicted in Figure 4.31. This network consists of four loading stations, five junctions, and three unloading stations close together connected via single-direction track segments, where the free-flow travel time is indicated for each link.

We consider the second case where the network has *more unloading stations close together*. Hence, we will compute the control for the switch-in and the switch-out of each junction in the network except the control of the switch-out of the junction directly connected to the unloading stations. The low-level controller for this special switch-out is computed according to *Pattern* 2. So, during the time interval $[t_k, t_{k+1})$ with $t_k = t_0 + k\tau_\text{s}$ with $k \in \mathbb{N}$, all unloading stations are served.

Then the evolution of each queue at the end of a link in the network, $q_{s,l}$ for $s = 1,2,3,4$ and $l = 0,1$ is given by:

$$q_{s,l}(k+1) = \max\left(0, f_{s,l}(k)\right)$$

with $f_{s,l}(k)$ defined as follows:

$$\begin{aligned}
f_{1,0}(k) =& q_{1,0}(k) + \left(D_1(k-2) - \left(1 - u_1^{\text{sw_in}}(k)\right)O^{\max}\right)\tau_\text{s} \\
f_{1,1}(k) =& q_{1,1}(k) + \left(\left(1 - u_2^{\text{sw_out}}(k-4)\right)O_2(k-4) - u_1^{\text{sw_in}}(k)O^{\max}\right)\tau_\text{s} \\
f_{2,0}(k) =& q_{2,0}(k) + \left(D_2(k-2) - \left(1 - u_2^{\text{sw_in}}(k)\right)O^{\max}\right)\tau_\text{s}
\end{aligned}$$

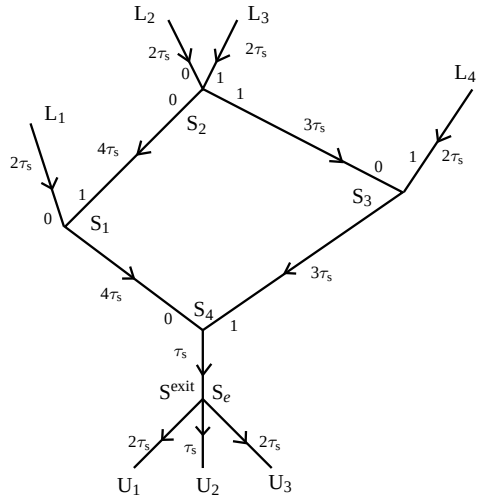


Figure 4.31: Case study for a DCV-based baggage handling system of Section 4.5.3.

$$
\begin{aligned}
f_{2,1}(k) =& q_{2,1}(k) + \big(D_3(k-2) - u_2^{\text{sw\_in}}(k) O^{\max}\big)\tau_{\text{s}} \\
f_{3,0}(k) =& q_{3,0}(k) + \Big(u_2^{\text{sw\_out}}(k-3) O_2(k-3) - \big(1 - u_3^{\text{sw\_in}}(k)\big) O^{\max}\Big)\tau_{\text{s}} \\
f_{3,1}(k) =& q_{3,1}(k) + \big(D_4(k-2) - u_3^{\text{sw\_in}}(k) O^{\max}\big)\tau_{\text{s}} \\
f_{4,0}(k) =& q_{4,0}(k) + \Big(O_1(k-4) - \big(1 - u_4^{\text{sw\_in}}(k)\big) O^{\max}\Big)\tau_{\text{s}} \\
f_{4,1}(k) =& q_{4,1}(k) + \big(O_3(k-3) - u_4^{\text{sw\_in}}(k) O^{\max}\big)\tau_{\text{s}}
\end{aligned}
$$

where $O_s(k)$ with $s \in \{1,2,3\}$ is given by (4.14).

The evolution of the partial queues corresponding to unloading station U_υ for $m = 1,2,3$ at the end of the link leading to S^{exit} is given by:

$$
q_\upsilon^{\text{exit}}(k+1) = q_\upsilon^{\text{exit}}(k) + \big(\rho_\upsilon O_e(k) - U_\upsilon(k + \frac{\tau_\upsilon}{\tau_{\text{s}}})\big)\tau_{\text{s}}
$$

with $U_\upsilon(k)$ given by (4.19), and

$$
\begin{aligned}
O_e(k) = \min\Bigg(O^{\max}, \Big(\frac{q_{4,0}(k-1)}{\tau_{\text{s}}} + O_1(k-5)\Big)\big(1 - u_4^{\text{sw\_in}}(k-1)\big) + \\
\Big(\frac{q_{4,1}(k-1)}{\tau_{\text{s}}} + O_3(k-4)\Big) u_4^{\text{sw\_in}}(k-1)\Bigg).
\end{aligned}
$$

Scenarios

We assume that the velocity of each DCV varies between $0\,\text{m/s}$ and $20\,\text{m/s}$. In order to faster assess the efficiency of our control method we assume that we do not start with an empty network but with a network already populated by DCVs transporting bags.

To compare the results we consider six scenarios where 800 bags have to be handled for different initial states of the system, where DCVs are waiting to cross different junctions

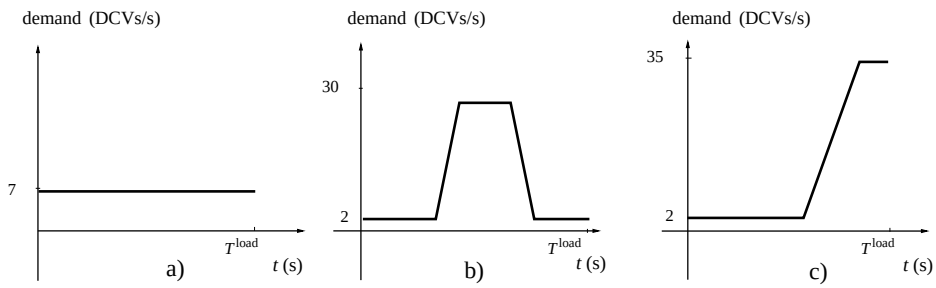


Figure 4.32: Demand profile.

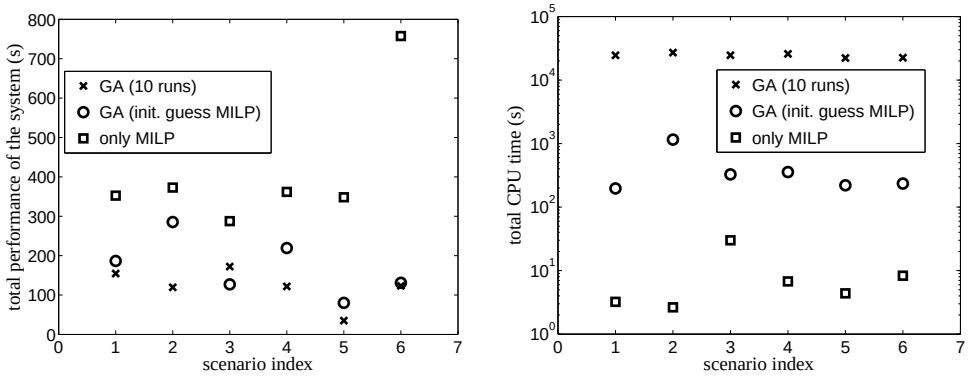
in queues of different length, and where $\rho_v = 25$ % for $m = 1,2$, and $\rho_3 = 50$ %. For this particular case study we consider $w_v = 1$ for $m = 1,2,3$. We simulate a period of 600 s, for a network where the capacity of each junction is 5 DCVs/s (this is a realistic value for the junction capacity when no switching occurs). The simulation time step τ_s is set to 20 s. We consider the bag arrival pattern for each loading station according to the three different classes of demand profiles sketched in Figure 4.32, where $T^{\text{load}} = 100$ s is the total loading time. The demand of each loading station equals 0 for $t \geq T^{\text{load}}$. These scenarios will involve very tight transportation since the time window for each unloading station is $[t_0 + 150\,\text{s}, t_0 + 350\,\text{s})$ — the last bag that enters the system can only arrive in time at the corresponding end point if the DCV travels the shortest route with maximum speed.

Results

Let us now compare the results obtained when using the proposed predictive control method with different formulations of the optimization problem.

In order to solve the MILP optimization problem (4.29) we have used the CPLEX solver implemented through the `cplex` interface function of the Matlab *Tomlab* toolbox, while to solve the original mixed integer nonlinear MPC optimization problem (4.28) we have chosen again the *genetic* algorithm with multiple runs and "bitstring" population of the Matlab optimization toolbox *Genetic Algorithm and Direct Search*. Note that typically the `ga` Matlab function starts the search from random initial points which have been set by the algorithm. However, this function has also the option to allow the user to set the initial search point. Then we can apply directly the results of the MILP optimization to the original nonlinear route choice problem, we can solve the nonlinear optimization problem starting from random initial points only, or we can use the solution of the MILP optimization problem as a good initial guess when solving the nonlinear optimization. As prediction horizon we have considered $N = 8$ for all MPC optimization problems.

Based on simulations we now compare, for the given scenarios, the results obtained for the proposed formulations of the optimization problem. The results of the simulations are reported in Figure 4.33. Note that the MPC performance index penalizes the absolute difference between the actual outflow profile and the desired outflow profile at each unloading station, and the queues in the network (as described in (4.27)), while the total performance of the system used to compare the proposed formulations penalizes both, the overdue time and the additional storage time at the end point (as described in (4.4)).



(a) closed-loop performance (b) computation time

Figure 4.33: Comparison of the results obtained using the proposed MPC formulations for the total closed-loop simulation. At each MPC step we consider three possible approaches: (1) we solve the MILP optimization only, (2) we use the MILP solution as good initial guess to solve the original MPC optimization problem, and (3) we solve the MPC original optimization starting from random initial points. The nonlinear MPC optimization problem is solved using a genetic algorithm (GA) with "bitstring" population.

These results confirm that computing the route choice using the original nonlinear formulation for the MPC optimization problem gives better performance than using only the MILP formulation. However, this happens at the cost of higher computational effort. Finally, we also compute the DCVs route choice using as initial feasible solution for the original nonlinear MPC problem the control sequence determined by solving the MILP optimization problem. As illustrated in Figure 4.33, the results indicate that this last method offers a good trade-off between performance and computational effort.

4.5.4 Route choice control using a hierarchical control framework

In this subsection we want to assess the efficiency of the hierarchical route choice control framework presented in Section 4.4.8. Recall that in Section 4.5.2 we have compared the results obtained when using a 1-layer route choice control framework. Therefore, in order to compare the efficiency of the proposed control frameworks, we will now select three representative methods of the 1-layer route choice control framework, namely:

1. centralized MPC presented in Section 4.4.2,
2. distributed MPC with a single round of downstream and upstream communication presented in Section 4.4.4,
3. distributed heuristics presented in Section 4.4.7.

These methods have been chosen since they are the first three methods (in Table 4.2) that give good performance for a DCV-based baggage handling system when the computation time is not an issue.

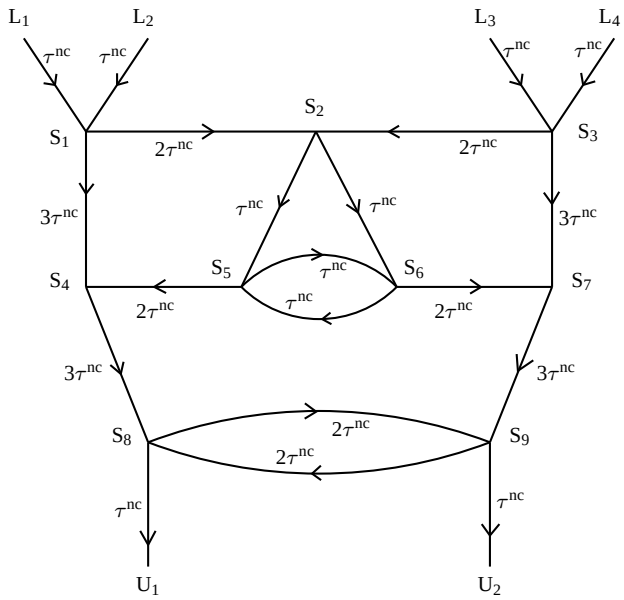


Figure 4.34: Case study for a DCV-based baggage handling system of Section 4.5.4.

Set-up

We consider the network of tracks depicted in Figure 4.34 with four loading stations, two unloading stations, nine junctions, and twenty unidirectional links, where the free-flow travel time is provided for each link. Note that the proposed hierarchical control allows the choice of routes containing loops.

We again assume $v^{\max} = 20\,\mathrm{m/s}$ and $\tau^{\mathrm{switch}} = 2\,\mathrm{s}$. As for the previous case studies, we assume that we do not start with an empty network but with a network already populated by DCVs transporting bags, as presented next.

Scenarios

In order to assess the performance of the proposed hierarchical control framework we define six scenarios where 2400 bags will be loaded into the baggage handling system (600 bags at each loading station). We consider three classes of demand profiles called "$\mathbf{dp}_1$", "$\mathbf{dp}_2$", and "$\mathbf{dp}_3$" hereafter. According to these classes, the bags arrive at each loading station in the time interval $[t_0, t_0 + 180\,\mathrm{s})$, the arrival times at a loading station being allocated randomly, using a uniform distribution according to the following cases:

$\mathbf{dp}_1$: the bags arrive at the loading station with a constant rate of 3.33 bags/s;

$\mathbf{dp}_2$: 50 bags arrive at a loading station during each of the time intervals $[t_0, t_0 + 60\,\mathrm{s})$ and $[t_0 + 120\,\mathrm{s}, t_0 + 180\,\mathrm{s})$, and the rest of 500 the bags arrives during $[t_0 + 60\,\mathrm{s}, t_0 + 120\,\mathrm{s})$;

$\mathbf{dp}_3$: 100 bags arrive during the time interval $[t_0, t_0 + 120\,\mathrm{s})$ and the rest of the bags, i.e., 500 bags, arrives after $t = t_0 + 120\,\mathrm{s}$, i.e., during $[t_0 + 120\,\mathrm{s}, t_0 + 180\,\mathrm{s})$.

We also consider two different initial states of the system where 60, and respectively 120 DCVs are already transporting bags in the network, running from loading stations $L_1, \ldots, L_4$ to junctions S_1 and S_3, from S_1 to S_2, and from S_3 to S_2. Their positions at t_0 are assigned such that between each 2 consecutive DCVs we have a minimum safe distance of 2 m, and between the DCV closest to the next to be passed junction and the junction we again have 2 m. The static priorities of these DCVs are assigned randomly in the set $\{1,2\}$ using a uniform distribution.

We assume that we have only two flights assigned to the unloading stations U_1 and U_2 (one flight assigned to one unloading station). Also, assuming that we start the simulation at time instant $t_0 = 0$ s, we consider that the time windows within which we need the bags at their end points are $[t_0 + 800\,\text{s}, t_0 + 1400\,\text{s})$ for U_1 and $[t_0 + 1000\,\text{s}, t_0 + 1600\,\text{s})$ for U_2.

We simulate a period of 40 minutes. The control time step for the network controller is set to 60 s, while the control time step for the switch controller is set to 2 s. Note that in these scenarios we also consider the occurrence of queues at origin.

Results

In this section we compare the results obtained when using the proposed hierarchical control framework and the approaches of a 1-layer control framework that have proved to give good performance in Section 4.5.2: centralized MPC, distributed MPC with a single round of downstream and upstream communication, and distributed heuristics.

In order to solve the MILP optimization of the network controller we have used the CPLEX solver of the Matlab optimization toolbox *Tomlab*, while to solve the nonlinear optimization problem of the switch controller we have chosen the *genetic* algorithm implemented in Matlab via the function `ga` with multiple runs (for these simulations we run the genetic algorithm three times for each optimization). Note that in order to keep the total computation time low, for both approaches — hierarchical MPC and centralized MPC — we shift the horizon with N, respectively $N^{\text{sc,max}}$ samples at each MPC step. Also, due to the same reason (computational requirements), we allow a limited amount of time for solving an optimization problem corresponding to the centralized route choice control and distributed MPC with a single round of downstream and upstream communication (the computation time allowed for each optimization is of 1 hour and 80 seconds, respectively).

As prediction horizon we consider $N = 6$ for the network controller and $N^{\text{sc,max}} = 15$ for the switch controller of the hierarchical control, $N = 40$ for the centralized MPC switch control, and $N = 5$ for the distributed MPC. We have chosen these values since simulations indicate that they give a good trade-off between the total computation time and performance.

Based on simulations we now compare, for the given scenarios, the results obtained for the proposed control frameworks. The results of the simulations are reported in Figure 4.35. For this comparison we consider the total performance of the system defined in Section 4.3 that penalizes both the overdue time and the additional storage time for each of the bags to be handled:

$$J^{\text{tot}}(\mathbf{t}) = \sum_{i=1}^{N^{\text{bags}}} J_i^{\text{pen}}(t_i^{\text{unload}})$$

with N^{bags} the number of bags to be handled.

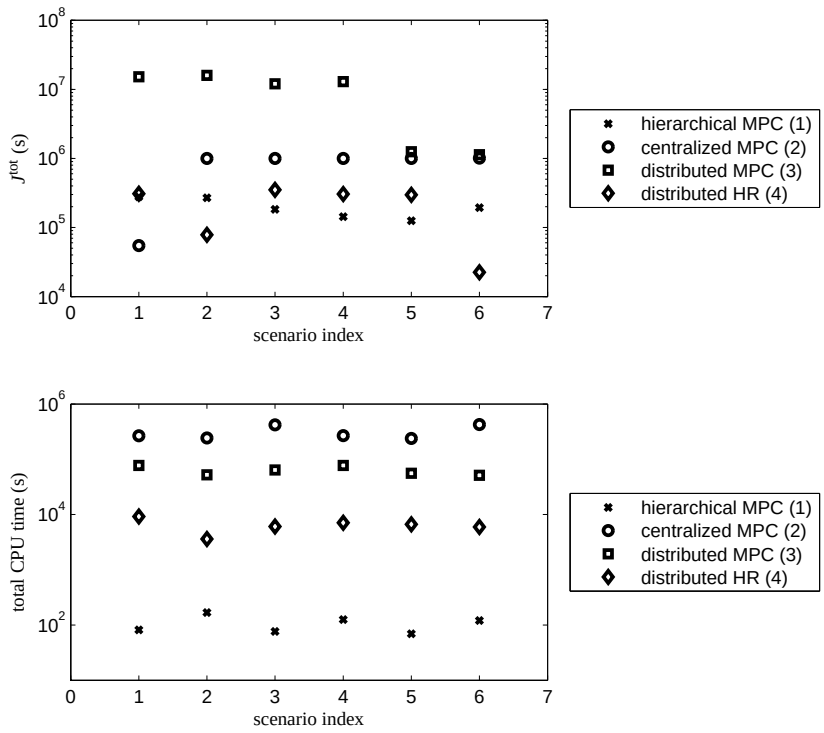


Figure 4.35: Comparison of the results obtained for the total closed-loop simulation when using (1) the hierarchical control framework, (2) the centralized route choice control approach presented in Section 4.4.2, (3) the distributed MPC with a single round of downstream and upstream communication (distributed MPC 2) presented in Section 4.4.4, and (4) the distributed heuristic approach presented in Section 4.4.7.

Recall that the lower the performance index J^{tot} is, the better the performance of the baggage handling system is. The simulation results indicate that using the hierarchical control framework typically yields a better system performance than using centralized MPC or distributed MPC with a single round of downstream and upstream communication, the solutions of which were returned by the prematurely terminated global and multi-start local optimization method. However, even with the computational restrictions mentioned above (we allow a limited amount of time for solving an optimization problem), the total computation time[21] of centralized MPC and of distributed MPC with a single round of downstream and upstream communication (over 17 hours) is much larger than the one of the hierarchical control (an average of 100 s per junction, plus 5 s for solving the MILP optimization problems).

Moreover, these results indicate that the performance index J^{tot} obtained when using the distributed heuristics (for $\tau^{\text{pred}} = 5$ s) is close to the one obtained when using the hierarchical control framework, and sometimes even lower (e.g., for scenario 2 and scenario 6), but the

[21] The simulations were performed on a 3.0 GHz P4 with 4 GB RAM.

total computation time required to determine the solution is also much larger when using the distributed heuristics.

Hence, the hierarchical control with MILP flow solutions offers a balanced trade-off between the performance of the system and the total computation time required to determine the route choice solution.

4.6 Summary

In this chapter we have considered the baggage handling process in large airports using destination coded vehicles (DCVs) running at high speeds on a network of tracks. Then, for a DCV-based baggage handling system, a fast event-driven model of the continuous-time bag handling process has been determined. Next, we have elaborated the performance criterion for this system. This performance criterion is then used to compare the control methods that we have proposed to use in order to optimally route the DCVs through the baggage handling system. In particular, we have developed and compared optimal control and centralized, decentralized, and distributed predictive methods for route choice control.

In practice, optimal control (OC) and centralized model predictive control (MPC) are not suitable for determining the optimal DCV route choice control due to the high computation time required to solve the route choice optimization problem. However, using *decentralized MPC* lowers the computation time. Furthermore, the results indicate that *distributed MPC* may give better performance than decentralized MPC, but at the cost of higher computational effort.

Since the MPC methods based on the event-driven route choice model involve solving a nonlinear, nonconvex, mixed integer optimization problem that is very expensive in terms of computational effort, we have also proposed an alternative approach for reducing the complexity of the computations by simplifying the nonlinear optimization problem and writing it as a mixed integer linear programming (MILP) optimization problem. The advantage is that for MILP problems solvers are available that allow us to efficiently compute the global optimal solution. The solution of the MILP problem can then be used as a good initial starting point for the original nonlinear optimization problem. Finally, in order to reduce the computational requirements, we have also proposed two heuristic methods and a hierarchical control framework. Simulations confirm that the decentralized and distributed heuristic approaches give typically very fast results, but the performance of the system when using the heuristic approaches is worse than when using the predictive methods. Finally, the results show that the hierarchical control with MILP flow solutions offers a balanced trade-off between the performance of the system and the total computation time required to determine the route choice solution when a limited amount of time is allowed for solving the optimization problems.

In future work we will also develop efficient control methods to solve the line balancing problem, and accordingly compute the optimal route choice for all (empty and loaded) DCVs in the network. We will also consider the early baggage storage area and analyze whether presorting the bags going out of the early baggage storage area can improve the performance of the system. Finally, we will apply these methods to more complex case studies.

Chapter 5

Conclusions and future research directions

In this chapter we will first present the summary and the conclusions of this thesis. Next, we will summarize the main contributions of the thesis. Finally, we will discuss the remaining open problems and we will give some recommendations for future research.

5.1 Summary and conclusions

In this thesis we have considered two specific applications of transportation systems for material handling, namely mail sorting machines in mail sorting centers and baggage handling systems in airports. Accordingly, this section presents the summary and the conclusions regarding each of the considered applications.

Postal automation We have considered mail sorting machines for large mail items such as newspapers, catalogs, and large letters, which have been shortly called “flats”. These sorting systems are then called “flat sorting machines”.

Regarding this application, we have first given a brief description of how flat sorting machines currently work. Afterwards, we have proposed a new set-up by making minor design changes, i.e., adding extra feeders and moving the bottom bin system. For the new set-up we have determined an event-driven model of the continuous-time process that has later on been used for model-based control. In order to ensure the optimal speed of the bin movements we have implemented and compared advanced control methods. In particular we have considered the following control approaches:

- different variants of optimal control with gradually decreasing complexity, namely:
    1. optimal control with a piecewise constant speed on time intervals of variable length,
    2. optimal control with a piecewise constant speed on time intervals of constant length,
    3. optimal control with a constant speed,

- model-based predictive control (MPC) with a piecewise constant speed on time intervals of constant length.

According to the obtained results we have concluded that MPC is the most appropriate control method to determine the velocity of the bottom system for the proposed flat sorting machine. To support this conclusion we have noted that MPC gives a throughput within 1% deviation of the throughput achieved when applying the most complex variant of optimal control that we have considered (optimal control with a piecewise constant speed on time intervals of variable length). Moreover, we have noted that MPC can compute real-time control actions, while optimal control requires an extremely large computation time.

We have also analyzed how the structural changes — the increased number of feeders, the variable position of inserting devices — and parameter changes — the increased maximal bound for the velocity of the top system of the flat sorting machine — influence the throughput of the automated flat sorting machine. Based on simulation results we draw the following conclusion: increasing the speed of the top system only, does not have as immediate consequence an increase in the throughput. Hence, determining the optimal bottom velocity is still needed in order to maximize the efficiency of the flat sorting machine.

Baggage handling We have considered the part of the baggage handling system which transports the bags in an automated way using destination coded vehicles (shortly called DCVs). These DCVs run at high speed on a network of tracks, transporting one bag at the time.

We have first given a brief description of how DCV-based baggage handling systems currently work. Next, we have determined a fast event-driven model of the continuous-time baggage handling process that has been later on used for model-based control. In order to maximize the efficiency of this system we have developed and implemented advanced control methods that could be used to optimally route the DCVs through the system. In particular, we have developed and compared efficient centralized, decentralized, and distributed predictive methods, and efficient decentralized and distributed heuristic approaches.

Based on the obtained simulation results we draw the following conclusions:

1. *Optimal control* becomes intractable in practice even for a network with a very small number of junctions due to the high computation time required to determine the optimal routing.

2. *Centralized MPC* still requires high computational effort to determine the control of DCV routes . The simulation results indicate that for a network with nine junctions centralized MPC requires more than one hour to compute the optimal routing for an horizon of 40 bags, while in practice, at rush hours, a large number of bags (over 2000 bags/hour) have to be handled within an hour.

3. *Decentralized MPC* lowers the computation time due to the parallel computation of the local control actions. However, this comes at the cost of reduction in the total performance.

4. *Distributed MPC* typically gives better performance than decentralized MPC, but at the cost of higher computational effort than decentralized MPC. This happens due to the required communication and coordination in computing the control actions.

5. *Decentralized* and *distributed heuristic approaches* determine the DCV routing while requiring much lower computational effort. However, these approaches give typically worse results than the predictive methods.

Since model-based predictive control methods involve solving a nonlinear, nonconvex, mixed integer optimization problem that is very expensive in terms of computational effort, we have also proposed an alternative approach for reducing the computational complexity. This alternative approach consists of simplifying the nonlinear optimization problem and then writing it as a mixed integer linear programming (MILP) optimization problem. The advantage is that for MILP problems solvers are available that allow us to efficiently compute the global optimal solution. This approach involves again a trade-off between the efficiency of the system and the total computation time since one can directly apply to the real system the solution obtained using the MILP formulation of the optimization problem, or use it as a feasible initial guess for the original (nonlinear) optimization problem.

Finally, we have also proposed a hierarchical control framework for computing the optimal routes. In this control framework switch controllers provide position instructions for each switch in the network. A collection of switch controllers is then supervised by a so-called network controller that mainly takes care of the route choice instructions for DCVs. Based on the obtained results we conclude that computing the optimal routes using the hierarchical control framework outperforms the centralized route choice control when a limited amount of time is allowed for solving the optimization problems of centralized route choice control.

5.2 Main contributions

In this section we present the main contributions of this research regarding the two applications that we have considered. Note that the control approaches that we have developed for these systems can also be applied to other transportation systems, e.g., distribution systems, automated guided vehicles in warehouses, port container terminals, or manufacturing systems.

The main contributions of this research with respect to postal automation are the following:

- We have proposed an event-driven model for the continuous-time flat sorting system which has been designed such that the destination bins can move bidirectionally with variable speed.
- We have developed and compared efficient model-based control methods to compute the speed profile of the destination bins that maximizes the throughput of the flat sorting machine. In particular, we have proposed variants of optimal control with gradually decreasing complexity and model predictive control.

The main contributions of this research with respect to baggage handling systems are the following:

- We have proposed an event-driven model for the continuous-time DCV-based baggage handling system.

- We have developed and compared efficient model-based control methods to compute the optimal routing of DCVs transporting bags from a given origin to a given destination such that the performance of a DCV-based baggage handling system is maximized. In particular we have considered centralized, decentralized, and distributed model predictive control, and heuristic approaches. We have also proposed a hierarchical route choice control framework for the DCV-based baggage handling system.

5.3 Open problems and recommendations for future research

In this section we briefly present some of the open problems that still have to be tackled with respect to the applications considered in this thesis. Additionally, we give some recommendations for future research.

Postal automation

Regarding this application one could further develop control methods to compute the speed profile of the bottom part for a flat sorting system. Next, we will develop efficient control methods for higher-level control problems — such as optimally assigning identification codes to the destination bins, sorting the flats in end-delivery sequence — that are currently not optimized. Finally, we will present other design changes for a flat sorting system, and other means to increase the efficiency of the postal services.

New control methods to set the speed of the bottom system In this thesis we have compared several model-based control methods that could be used to determine the optimal velocity of the bottom system of an augmented design for a flat sorting machine. In future work also other control methods will be considered such as fast rule-based approaches, neural networks, see, e.g., [36], and fuzzy-based approaches, see, e.g., [63]. These approaches can use the receding horizon principle. Then, at each time step, the following steps are involved:

- **automatic feature extraction:** Recall that before entering the sorting phase, we know, for a buffer of flats, the identification codes which were also printed on each item in form of a bar code. Then, given the buffer of identification codes, we will compute one or more feature measures for the stream of identification codes (such as entropy).
- **automatic speed calculation:** In order to determine the speed of the destination bins we will use neural networks and fuzzy-based approaches, or rule-based approaches.
- **use of speed:** Then we will apply the computed speed to the system for a given time period.

Also note that one can use the solution of these approaches as initial feasible solution when solving the MPC optimization problem described in Section 3.5.2.

In future work we will also determine an approximation for the minimum number of scenarios that we have to use when comparing control methods such that the relative error is smaller than a given bound ϵ ($0 < \epsilon < 1$) with a given probability δ ($0 < \delta < 1$), see [51].

Control methods for higher-level control problems A higher-level control problem for a flat sorting system is to assign destinations to the bins that collect the sorted mail so that the overall performance is optimized when considering a given set of scenarios. Currently, this is done by assigning destinations to the bins before the process starts (several destination addresses and postal codes can be assigned to one bin if the corresponding addresses are close on the route that a post man takes for the mail delivery). However, one can increase the performance of the flat sorting machine by optimally assigning identification codes to the destination bins. This can be done beforehand, by solving off-line a mixed integer optimization problem over a given set of scenarios. This optimization problem will have multiple objectives (to determine the optimal speed of the bins for a given destination assignment, and to determine the optimal destination assignment). Therefore, one can solve, in an inner loop, the optimization problem that has the goal to determine the optimal speed using optimal control or model predictive control. Then, this speed will be used when solving, in an outer loop, the optimization problem that will determine the optimal destination assignment. Both optimizations will be solved with respect to the model of the flat sorting system and so that the operational constraints are satisfied.

Another open problem for a flat sorting system is to ensure the correct order of the mail not only with respect to postal codes and street names, but also with respect to the street and house number. Then this would save time when actually delivering the mail items. The sequence of mail sorted with respect to the street and house number is called "end-delivery sequence" of the sorted mail. Currently, to obtain this order, the mail is sorted several times. Therefore, designing a sorting machine that would ensure the fast mail sorting in end-delivery sequence is a big challenge. To this aim, one can develop intelligent control methods to dynamically allocate destinations to the destination bins and to the intermediary pockets collecting the mail, so that the number of sorting rounds necessary to ensure the end-delivery sequence of the sorted mail is minimized.

In the future all the big companies that use the postal service to deliver their catalogs or advertisement brochures should also print the address and postal code according to the end-delivery sequence that mail sorting centers use. This could additionally decrease the time needed for sorting the flats in end-delivery sequence.

Changes and optimizations In order to further increase the efficiency the flat sorting system one can also make other design changes such as:

- Augment the flat sorting system with more intermediate levels of transport pockets between the top part of the system and the bottom part. Then we will not have only transport boxes and destination bins, but also several layers of intermediate transportation. Then for this new set-up one can develop efficient control methods to dynamically allocate destinations for each intermediate pocket so that the items are sorted in end-delivery sequence as fast as possible.
- Augment the flat sorting system with smaller sorting systems at each destination bin. Then these smaller secondary systems would sort the items in end-delivery sequence.
- Augment the system with a weight sensor. Then, knowing the weight of each flat, we can determine more accurately the time instant for correct dropping and stacking. As a consequence, we can increase the maximum relative velocity between the top and

the bottom system, and this, combined with determining the optimal speed profile for the bottom system, will increase the throughput of a flat sorting machine.

Furthermore, the scope of the research can be broadened. Then, in order to increase the efficiency of the postal services, while minimizing the costs, one can:

- Remove more human work from the postal services by using, e.g., automated guided vehicles or conveyor systems to transport the bins with sorted mail to the postal vehicles.
- Optimize the number and the position of mail sorting centers and post offices.
- Optimize also the routes and the number of postal vehicles and postal trucks needed when delivering the mail, or when transporting mail from one mail sorting center to another.

Baggage handling

Regarding this applications we will further improve the control methods already developed in this thesis. Next, we will continue developing other efficient control methods for the DCV route choice problem. We will also develop advanced control methods for other control problems — such as presorting the bags that leave the early baggage storage area, line balancing, and empty cart management— which are currently not optimized. Finally, we will present other means to increase the efficiency of a DCV-based baggage handling system.

Improving the developed control methods Regarding the DCV-based baggage handling systems, one can further develop and analyze control methods that could be used to efficiently route the DCVs on the network of tracks. Therefore, in future work we will also consider several extensions of the current approaches:

- We will improve the heuristic approaches (e.g., we will take into account more features (such as density of DCVs on the links of the network) when computing the control action for the switch out of a junction, and also optimize the number of links that one will look farther when estimating the time that a bag will spend in the system.
- We will combine the model-based predictive control with heuristics (e.g., one can use the solution of the heuristic approach as good initial guess for the optimization algorithm).
- Regarding the distributed control we will:
  - use multiple up and down rounds of optimizations,
  - extend the range of communication exchange to more than one level,
  - extend the local control area to more than one node.

New route choice control methods We can develop new efficient control approaches to determine the DCV route choice. These approaches involve the use of neural networks and fuzzy control.

Furthermore, one can introduce the concept of *platooning*, [92]. In this framework the groups of DCVs will travel closely spaced together with short intervehicle distances and

larger distances between platoons. So, we can develop model-based methods for optimally creating, routing, and splitting platoons of DCVs. The biggest advantage of routing platoons of DCVs instead of individual DCVs will be a much lower computation time than the one obtained when using the distributed approaches developed in this thesis. This happens because we will then compute routes for platoons instead of computing routes for individual DCVs.

Finally, we will compare the efficiency of these methods with the performance of the control methods already developed in this thesis.

New control problems Note that state-of-the-art baggage handling systems also have an early baggage storage area where the bags that have been checked-in too early can be additionally stored. In this thesis we have not considered the early baggage storage area, but in order to emulate its presence, one or more loops were included in which the bags that entered the network of tracks too early were kept. In busy airports, the order in which the bags leave the early baggage storage area also has high importance. Therefore, in future work we will consider *presorting* the bags that leave the early baggage storage area so that the overall performance of the DCV-based baggage handling systems is maximized. In order to optimally presort the bags leaving the early baggage storage area one can design a local MPC controller to solve on-line an integer optimization problem. Hence, for a given prediction period, we will compute the vector of bag identification codes that leave the early baggage storage area during the prediction period so that the performance of the DCV-based baggage handling system is optimized with respect to the dynamics of the system and its safety and operational constraints.

Next, one can also use the concept of *platooning* for the bags leaving the early baggage storage area, and develop efficient control methods for optimally creating the platoons.

Also, recall that in this thesis we have assumed a sufficient number of DCVs to be present in the system so that when a bag is at the loading station there is a DCV ready to transport it. In practice, we also have to efficiently manage the empty DCVs in order to ensure a balanced service to all loading stations. Hence, we have to develop intelligent control methods that will dynamically assign loading stations to each empty DCV and efficiently route the DCV through the network so that all loading stations have sufficient DCVs in their buffer. Note that the problem of dynamically assigning loading stations to each empty DCV is also called the “line balancing” problem, while the problem of routing the empty DCVs through the network is also called “empty cart management”. In order to solve these problems, one can develop control methods similar to the ones already developed in this thesis or the ones proposed as recommendations for future research.

Changes and optimizations Finally, one can optimize the number of DCVs required for an efficient baggage handling system, and minimize the energy consumption. Also, one can investigate whether the layout of the network can be optimized while minimizing the costs of the infrastructure and maximizing the overall performance of the DCV-based baggage handling system. Moreover, one could investigate how the network of tracks should be equipped with sensors so that events such as congestion or jams can be determined in time and, as a consequence, how to maximize the efficiency of this system while minimizing the number of sensors in the network (and, consequently, minimizing the costs).

Other recommendations

For both applications (postal automation and baggage handling) one can also include more complex dynamics of the system than those that have been considered in this thesis (e.g., one can include the acceleration and deceleration of the bottom system of a flat sorting machine and of the DCVs respectively, instead of considering a piecewise constant speed of their movements), friction, characteristics of motors, and the effects and the limitations of distributed actuation.

The control approaches developed in this thesis can be successfully used also in controlling and optimizing other applications of transportation systems such as:

- Roller belts for people. These systems are often used in airports or in buildings where people have to travel (on foot and in a short time) large walking distanced between important points of the building. These systems are very attractive due to their continuously transport capacity during operation. An important problem of such systems is to determine the optimal speed of a roller belt that minimizes the energy consumption, while maximizing the people's satisfaction, and while quaranteeing some level of safety and passenger comfort. To this aim optimization problems can be solved similar to how we proceed for determining the optimal speed for the bottom part of a flat sorting system.
- Automated guided vehicles (AGVs). The AGVs are completely automated vehicles that can load, unload, and transport goods in warehouses, port container terminals, or manufacturing systems. Typically, they navigate from a point to another along fixed pathways by following some markers. Hence, an efficient use of AGVs can increase the performance of transportation in the production, trade, and service sector, while minimizing the energy consumption. The systems consisting of AGVs deal with planning, routing, and scheduling problems just as the DCV-based baggage handling system. In particular, for large-scale AGV-based systems, the efficient planning, routing, and scheduling is difficult. Then, one can apply the control methods presented in this thesis.
- Traffic systems. Advanced technologies from the field of control theory, communication, and information technology are currently being combined with the existing road transportation infrastructure and equipment. Hence, soon, intelligent vehicles will be driving on roads in an automated way. The routing and speed problems of these intelligent vehicles can also be controlled using the methods presented in this thesis. Similar approaches can be used for unmanned aerial vehicles (UAVs). The UAVs could fly freely on optimal routes between an origin and a destination. Then distributed controllers could compute the routes and speeds of UAVs flying on a 3-D network such that a smooth, efficient, reliable, and safe automated flight control is ensured.
- Power distribution and water management. The network infrastructure of the systems dealing with power distribution and water management can be very complex. The control problem of these systems can be stated as follows: compute the optimal power flow or water level, respectively, such that the overall performance of the system is maximized. Then lower level controllers could try to achieve the optimal power flow or water level. Therefore, one can use decentralized and distributed control methods similar to the ones presented in this thesis.

Bibliography

[1] L. Acar. Some examples for the decentralized receding horizon control. In *Proceedings of the 31st IEEE Conference on Decision and Control*, volume 2, pages 1356–1359, Tucson, Arizona, USA, December 1992.

[2] T. Le Anh. *Intelligent Control of Vehicle-Based Internal Transport Systems*. PhD thesis, Erasmus University Rotterdam, 2005.

[3] A. Atamtürk and M. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140(1):67–124, November 2005.

[4] C. Audet and J.E. Dennis. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, February 2003.

[5] P. Banerjee and Y. Zhou. Facilities layout design optimization with single loop material flow path configuration. *International Journal of Production Research*, 32(1): 183–203, January 1995.

[6] L.D. Baskar, B. De Schutter, and H. Hellendoorn. Hierarchical traffic control and management with intelligent vehicles. In *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium (IV'07)*, pages 834–839, Istanbul, Turkey, June 2007.

[7] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[8] R. Berber, editor. *Methods of Model Based Process Control*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[9] P.R. Bhave and R. Gupta. *Analysis of Water Distribution Networks*. Alpha Science International, Oxford, UK, 2006.

[10] G. Black and V. Vyatkin. On practical implementation of holonic control principles in baggage handling systems using IEC 61499. In M.A. Crew and P.R. Kleindorfer, editors, *Holonic and Multi-Agent Systems for Manufacturing*, pages 314–325. Springer, Berlin, Germany, 2007.

[11] E.F. Camacho and C. Bordons. *Model Predictive Control in the Process Industry*. Springer-Verlag, Berlin, Germany, 1995.

[12] E. Camponogara and S.N. Talukdar. Distributed model predictive control: synchronous and asynchronous computation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A*, 37(5):732–745, September 2007.

[13] E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, February 2002.

[14] C.R. Cutler and B.L. Ramaker. Dynamic matrix control – a computer control algorithm. In *Proceedings of Joint American Control Conference*, San Francisco, California, USA, August 1980. Paper WP5-B.

[15] C.F. Daganzo. *Fundamentals of Transportation and Traffic Operations*. Pergamon Press, New York, New York, USA, 1997.

[16] M.C. De Guzman, N. Prabhu, and J.M.A. Tanchoco. Complexity of the AGV shortest path and single-loop guide path layout problems. *International Journal of Production Research*, 35(8):2083–2092, August 1997.

[17] R. de Neufville. The baggage system at Denver: Prospects and lessons. *Journal of Air Transport Management*, 1(4):229–236, December 1994.

[18] K.A. Dowsland. Simulated annealing. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, chapter 2, pages 20–69. John Wiley & Sons, Inc., New York, New York, USA, 1993.

[19] W.B. Dunbar and R.M. Murray. Model predictive control of coordinated multi-vehicle formations. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 4, pages 4631–4636, Las Vegas, Nevada, USA, December 2002.

[20] W.B. Dunbar and R.M. Murray. Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42(4):549–558, April 2006.

[21] K. Duzinkiewicz, M.A. Brdys, and T. Chang. Hierarchical model predictive control of integrated quality and quantity in drinking water distribution systems. *Urban Water Journal*, 2(2):125–137, June 2005.

[22] P.J. Egbelu and J.M.A. Tanchoco. Potentials for bi-directional guide-path for automated guided vehicle based systems. *International Journal of Production Research*, 24(5):1075–1097, September 1986.

[23] W.R. Esposito and C.A. Floudas. Deterministic global optimization in nonlinear optimal control problems. *Journal of Global Optimization*, 17(1–4):97–126, September 2000.

[24] P. Falcone, F. Borrelli, H.E. Tseng, J. Asgari, and D. Hrovat. A hierarchical model predictive control framework for autonomous ground vehicles. In *Proceedings of the 2008 American Control Conference*, pages 3719–3724, Seattle, Washington, USA, June 2008.

[25] A. Fay. Decentralized control strategies for transportation systems. In *Proceedings of the 2005 IEEE International Conference on Control and Automation*, pages 898–903, Budapest, Hungary, June 2005.

[26] W Findeisen, F.N. Bailey, M.Brdys, K. Malinowski, P. Tatjewoki, and A. Wcrznial. *Control and Coordination in Hierarchical Systems*. John Wiley & Sons, New York, New York, USA, 1980.

[27] R. Fletcher and S. Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM Journal on Optimization*, 8(2):604–616, May 1998.

[28] R. Fletcher and M.J.D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, May 1963.

[29] M. Fuhrman and G. Tessitore. Existence of optimal stochastic controls and global solutions of forward-backward stochastic differential equations. *SIAM Journal on Control and Optimization*, 43(3):813–830, March 2004.

[30] R.J. Gaskins, J.M.A. Tanchoco, and F. Taghaboni. Virtual flow paths for free-ranging automated guided vehicle systems. *International Journal of Production Research*, 27 (1):91–100, January 1989.

[31] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London, UK, 1981.

[32] F. Glover and F. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1997.

[33] F. Glover, D.D. Klingman, and N.V. Phillips. A new polynomially bounded shortest path algorithm. *Operations Research*, 33(1):65–73, January 1985.

[34] W.G. Goetz and P.J. Egbelu. Guide path design and location of load pick-up/drop-off points for an automated guided vehicle system. *International Journal of Production Research*, 28(5):927–941, May 1990.

[35] I.E. Grossmann. Mixed-integer nonlinear programming techniques for the synthesis of engineering systems. *Research in Engineering Design*, 1(3–4):205–228, September 1990.

[36] K. Gurney. *An Introduction to Neural Networks*. UCL Press, London, UK, 1997.

[37] W.W. Hager and P.M. Pardalos, editors. *Optimal Control: Theory, Algorithms, and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[38] K. Hallenborg and Y. Demazeau. Dynamical control in large-scale material handling systems through agent technology. In *Proceedings of the 2006 IEEE /WIC/ACM International Conference on Intelligent Agent Technology*, pages 637–645, Hong Kong, China, December 2006.

[39] J. Huang, U.S. Palekar, and S.G. Kapoor. A labeling algorithm for the navigation of automated guided vehicles. *Journal of Engineering for Industry*, 115(3):315–321, August 1993.

[40] H. Kaneko and E. Fujiwara. A class of m-ary asymmetric symbol error correcting codes for data entry devices. *IEEE Transactions on Computers*, 53(2):159–167, February 2004.

[41] M. Kaspi and J.M.A. Tanchoco. Optimal flow path design of unidirectional AGV systems. *International Journal of Production Research*, 28(6):1023–1030, June 1990.

[42] T. Keviczky, F. Borrelli, and G.J. Balas. Hierarchical design of decentralized receding horizon controllers for decoupled systems. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 1592–1597, New Orleans, Louisiana, USA, December 2004.

[43] T. Keviczky, F. Borrelli, and G.J. Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42(12):2105–2115, December 2006.

[44] U. Kiencke and L. Nielsen. *Automotive Control Systems: For Engine, Driveline, and Vehicle*. Springer-Verlag, New York, New York, USA, 2000.

[45] C.W. Kim and J.M.A. Tanchoco. Operational control of a bidirectional automated guided vehicle system. *International Journal of Production Research*, 31(9):2123–2138, September 1993.

[46] D.E. Kirk. *Optimal Control Theory: An Introduction*. Dover Publications, New York, New York, USA, 2004.

[47] A.W.J. Kolen, A.H.G. Rinnooy-Kan, and H.W.J.M. Trienekens. Vehicle routing with time windows. *Operations Research*, 35(2):266–274, March 1987.

[48] P. Kouvelis and M. Kim. Unidirectional loop network layout problem in automated manufacturing systems. *Operations Research*, 40(3):533–550, May 1992.

[49] P. Kundur. *Power System Stability and Control*. McGraw-Hill, New York, New York, USA, 1994.

[50] A. Langevin, D. Lauzon, and D. Riopel. Dispatching, routing, and scheduling of two automated guided vehicles in a flexible manufacturing system. *International Journal of Flexible Manufacturing Systems*, 8(3):247–262, July 1996.

[51] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis*. McGraw Hill, New York, New York, USA, 2000.

[52] J. Leimer. Design factors in the development of an optical character recognition machine. *IEEE Transactions on Information Theory*, 8(2):167–171, February 1962.

[53] F.L. Lewis and V.L. Syrmos. *Optimal Control*. John Wiley & Sons, New York, New York, USA, 1995.

[54] S. Leyffer. Integrating sqp and branch-and-bound for mixed integer nonlinear programming. *Research in Engineering Design*, 18(3):295–309, March 2001.

[55] G. Lodewijks. *Dynamics of Belt Systems*. PhD thesis, Delft University of Technology, Delft, The Netherlands, October 1996.

[56] B. Lohmann. Throughput control for a transport process and an application in postal automation machines. *Control Engineering Practice*, 4(11):1503–1509, November 1996.

[57] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, UK, 2002.

[58] A.S. Matveev and A.V. Savkin. *Hybrid Dynamical Systems: Controller and Sensor Switching Problem*. Springer-Verlag, New York, New York, USA, 2001.

[59] F.H. Mossman and N. Morton. *Logistics of Distribution Systems*. Allyn and Bacon, Boston, Massachusetts, USA, 1965.

[60] R.R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Engineering Applications of Artificial Intelligence*, 21(3):353–366, April 2008.

[61] G.C. Onwubolu and B.V. Babu. *New Optimization Techniques in Engineering*. Springer-Verlag, Berlin, Germany, 2004.

[62] C. Park, D.J. Scheeres, V. Guibout, and A. Bloch. Global solution for the optimal feedback control of the underactuated Heisenberg system. *IEEE Transactions on Automatic Control*, 53(11):2638–2642, December 2008.

[63] G. Chen T.T. Pham. *Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*. CRC Press, Boca Raton, Florida, USA, 2000.

[64] F. Pla, J.M. Sanchiz, and J.S. Sanchez. An integral automation of industrial fruit and vegetable sorting by machine vision. In *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 6, pages 541–546, Antibes, France, October 2001.

[65] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, Wisconsin, USA, 2009.

[66] J.B. Rawlings and B.T. Stewart. Coordinating multiple optimization-based controllers: New opportunities and challenges. *Automatica*, 18(9):839–845, October 2008.

[67] C.R. Reeves and J.E. Rowe. *Genetic Algorithms – Principles and Perspectives: A Guide to GA Theory*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 2002.

[68] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(1):413–428, December 1978.

[69] A. Richards and J. How. Decentralized model predictive control of cooperating UAVs. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 4, pages 4286–4291, Paradise Island, Bahamas, December 2004.

[70] R. Scattolini and P. Colaneri. Hierarchical model predictive control. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4803–4808, New Orleans, Louisiana, USA, December 2007.

[71] Riccardo Scattolini. Architectures for distributed and hierarchical model predictive control – A review. *Journal of Process Control*, 19(5):723–731, May 2009.

[72] K. Schmidt, T. Moor, and S. Perk. Nonblocking hierarchical control of decentralized discrete event systems. *IEEE Transactions on Automatic Control*, 53(10):2252–2265, November 2008.

[73] F. Taghaboni and J.M.A. Tanchoco. Comparison of dynamic routeing techniques for automated guided vehicle systems. *International Journal of Production Research*, 33 (10):2653–2669, October 1995.

[74] L. Talbot. *Design and Performance Analysis of Multistation Automated Guided Vehicle Systems*. PhD thesis, Universite Catholique de Louvain, 2003.

[75] A. Tarău, B. De Schutter, and H. Hellendoorn. Multi-agent controllers for large-scale transportation systems — Application to postal automation. In *Proceedings of the 9th TRAIL Congress 2006 — TRAIL in Motion — CD-ROM*, Rotterdam, The Netherlands, November 2006.

[76] A. Tarău, B. De Schutter, and H. Hellendoorn. Route choice control for DCVs in baggage handling systems — Comparison between centralized and decentralized approaches. In *Proceedings of the 10th TRAIL Congress 2008 — TRAIL in Perspective — CD-ROM*, Rotterdam, The Netherlands, October 2008.

[77] A. Tarău, B. De Schutter, and J. Hellendoorn. Travel time control of destination coded vehicles in baggage handling systems. In *Proceedings of the 17th IEEE International Conference on Control Applications*, pages 293–298, San Antonio, Texas, USA, September 2008.

[78] A. Tarău, B. De Schutter, and H. Hellendoorn. Centralized versus decentralized route choice control in DCV-based baggage handling systems. In *Proceedings of the 2009 IEEE International Conference on Networking, Sensing and Control*, pages 334–339, Okayama, Japan, March 2009.

[79] A.N. Tarău, B. De Schutter, and J. Hellendoorn. Throughput optimization of automated flats sorting machines. In *Proceedings of the 17th IFAC World Congress*, pages 51–56, Seoul, Korea, July 2008.

[80] A.N. Tarău, B. De Schutter, and H. Hellendoorn. Receding horizon approaches for route choice control of automated baggage handling systems. In *Proceedings of the European Control Conference 2009*, pages 2978–2983, Budapest, Hungary, August 2009.

[81] A.N. Tarău, B. De Schutter, and H. Hellendoorn. Distributed route choice control in DCV-based baggage handling systems. In *Proceedings of the 18th IEEE International Conference on Control Applications*, pages 818–824, Saint Petersburg, Russia, July 2009.

[82] A.N. Tarău, B. De Schutter, and H. Hellendoorn. Hierarchical route choice control for baggage handling systems. In *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems*, pages 679–684, St. Louis, Missouri, USA, October 2009.

[83] A.N. Tarău, B. De Schutter, and H. Hellendoorn. Model-based control for route choice in automated baggage handling systems. Technical Report 09-048, Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, November 2009. Accepted for publication in *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*.

[84] A.N. Tarău, B. De Schutter, and J. Hellendoorn. Model-based control for throughput optimization of automated flats sorting machines. *Control Engineering Practice*, 17 (6):733–739, June 2009.

[85] A.N. Tarău, B. De Schutter, and J. Hellendoorn. Route choice control of automated baggage handling systems. *Transportation Research Record*, (2106):76–82, 2009.

[86] A.N. Tarău, B. De Schutter, and J. Hellendoorn. Centralized, decentralized, and distributed model predictive control for route choice in automated baggage handling systems. *Journal of Control Engineering and Applied Informatics*, Special Issue on Distributed Control in Networked Systems, 11(3):24–31, 2009.

[87] A.N. Tarău, B. De Schutter, and J. Hellendoorn. Predictive route choice control of destination coded vehicles with mixed integer linear programming optimization. In *Proceedings of the 12th IFAC Symposium on Control in Transportation Systems*, pages 64–69, Redondo Beach, California, USA, September 2009.

[88] A.N. Tarău, B. De Schutter, and J. Hellendoorn. Decentralized route choice control of automated baggage handling systems. In *Proceedings of the 12th IFAC Symposium on Control in Transportation Systems*, pages 70–75, Redondo Beach, California, USA, September 2009.

[89] A.N. Tarău, B. De Schutter, and J. Hellendoorn. Route choice control of automated baggage handling systems. In *Proceedings of the 88th Annual Meeting of the Transportation Research Board*, Washington DC, USA, January 2009. Paper 09-0432.

[90] A.J. van der Schaft and J.M. Schumacher. *An Introduction to Hybrid Dynamical Systems, Lecture Notes in Control and Information Sciences*, volume 251. Springer-Verlag, London, UK, 2000.

[91] Vanderlande Industries. Baggage handling. URL www.vanderlande.com.

[92] P. Varaiya. Smart cars on smart roads. *IEEE Transactions on Automatic Control*, 38 (2):195–207, February 1993.

[93] A.N. Venkat, I.A. Hiskens, J.B. Rawlings, and S.J. Wright. Distributed MPC strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, 16(6):1192–1206, November 2008.

[94] D.D. Šiljak. *Decentralized Control of Complex Systems*. Academic Press, San Diego, California, USA, 1991.

[95] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts, USA, 2000.

[96] D. Weyns and T. Holvoet. Architectural design of a situated multiagent system for controlling automatic guided vehicles. *International Journal on Agent Oriented Software Engineering*, 2(1):90–128, January 2008.

[97] A.P. Whichello and H. Yan. Locating address blocks and postcodes in mail-piece images. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 716–720, Vienna, Austria, August 1996.

Glossary

Terminology

Below we present the specific terminology used in this thesis:

Flats: Large mail items such as large letters, journals, magazines, and newspapers.

DCV: Metal cart with a plastic tub on top, used to transport at high speed one bag at the time on a network of tracks. These carts are propelled by linear induction motors similar to roller coasters.

List of abbreviations

The following abbreviations are used in this thesis:

DCV	Destination coded vehicle
AGV	Automated guided vehicles
OC	Optimal Control
MPC	Model Predictive Control
MILP	Mixed Integer Linear Programming
HR	Heuristics
GA	Genetic Algorithm

Conventions

The following conventions are used in this thesis for notation and symbols:

- A lower case character typeset in boldface, e.g., $\mathbf{x}$, represents a column vector. The transpose of a vector is denoted by the superscript $\top$. For instance, the transpose of $\mathbf{x}$ is $\mathbf{x}^\top$.
- The number of elements of a set Λ is indicated by $|\Lambda|$.
- The absolute value of a scalar variable x is denoted by $|x|$.

Samenvatting

In dit proefschrift wordt gefocused op twee specifieke transportsystemen, namelijk geautomatiseerde postsorteermachines en bagage-afhandelingssystemen.

Geautomatiseerde postsorteermachines

In de laatste decennia is de hoeveelheid tijdschriften, catalogi en andere in plastic verpakte poststukken die worden verwerkt in geautomatiseerde sorteercentra aanzienlijk toegenomen. Om deze grote stroom aan post te kunnen verwerken zijn state-of-the-art sorteercentra uitgerust met geautomatiseerde sorteermachines. De productiviteit van een sorteermachine is gedefinieerd als het aantal gesorteerde poststukken gedeeld door de tijd die nodig is om deze te sorteren. In dit proefschrift beperken we ons tot poststukken in A4 formaat enveloppen. Poststukken van deze afmetingen worden flats genoemd.

Kort samengevat werkt een geautomatiseerde flat sorteermachine als volgt: de flats worden door een invoegmachine in transportbakken geplaatst; de transportbakken bewegen met een constante snelheid en leveren poststukken af op stationaire bestemmingen volgens een vooraf bepaalde sorteerstrategie. De doorlooptijd van het hierboven geschetste basissysteem kan nog worden verkort door het systeem zo te ontwerpen dat de ontvangstbakken in twee richtingen kunnen meebewegen met de transportbakken.

Voor een continu sorteerproces wordt een event-driven model opgesteld met behulp van simulatie. Om de optimale snelheid van het ontvangstsysteem te berekenen wordt een aantal geavanceerde regelsystemen geïmplementeerd en vergeleken. De verschillende varianten van *optimal control* die worden vergeleken zijn, in volgorde van afnemende complexiteit: optimal control voor constante snelheden over tijdsintervallen met variabele lengte, optimal control met een constante snelheid over tijdsintervallen met constante lengte en optimal control met een constante snelheid. Vervolgens worden deze methoden vergeleken met een regelsysteem gebaseerd op *model predictive control* (MPC) voor een constante snelheid over tijdsintervallen van gelijke lengte. De voorgestelde methoden worden vergeleken in verschillende scenario's.

Vervolgens wordt geanalyseerd hoe structurele veranderingen zoals een toename in het aantal feeders, een variabele positie van het aantal invoersystemen en parametrische veranderingen zoals een verhoogde maximumsnelheid van het bovensysteem van de sorteermachine de doorlooptijd van de sorteermachine beïnvloeden.

Bagage-afhandelingssystemen

De continue vraag naar kostenbesparing in de luchttransportsector en de toename van goedkope vluchten vereist een effectievere werking van luchthavens. Deze doelstelling kan mede bereikt worden door het intelligenter afhandelen van de bagage door middel van automatisering en het gebruik van geïntegreerde sensoren, actuatoren en intelligente regeleenheden. Moderne bagage-afhandelingssystemen op grote luchthavens transporteren de bagage door middel van bestemmingsgecodeerde transportmiddelen (DCV's), dit zijn onbemande eenheden die met grote snelheden bagage via een netwerk van rails vervoeren. Zo'n bagage-afhandelingssysteem bestaat uit laadplekken, losplekken, een eenrichtingsnetwerk van transportsmiddelen met verschillende (lokale) lussen voor het laden, lossen en de opslag van DCV's, en de vroegtijdige bagage-opslag waar de bagage die vroegtijdig gearriveerd is enkele uren kan worden opgeslagen. Dit resulteert in een complexe infrastructuur met vele interacties tussen de verschillende componenten en plaatsen waar regelbeslissingen moeten worden genomen, waardoor een adaptieve, on-line management en regelstructuur vereist is.

Typische zaken in een geautomatiseerd bagage-afhandelingssysteem zijn de coördinatie van de "processing units", tijdsplanning, planning van alle middelen, routekeuze, het aansturen van het transport tussen verschillende vervoersmiddelen (bijvoorbeeld DCV's en lopende band) en het voorkomen van deadlocks en het vollopen van buffers. Tegelijkertijd moet er worden gestreefd naar een optimale doorstroom- en verwerkingstijd zodanig dat andere eisen en condities gerespecteerd worden (bijvoorbeeld het niet beschadigen van bagage en de bagage af te leveren binnen de gegeven tijd). Het bedienen van een DCV's-gebaseerd bagage-afhandelingssysteem vraagt daarom om het oplossen van eenvoudige bedieningsproblemen bijvoorbeeld de coördinatie en synchronisatie van het laden en lossen van de bagage op een DCV, de snelheidscontrole van elke DCV alswel het oplossen van hogere orde problemen bijvoorbeeld de route van alle DCV's door het netwerk. In deze studie leggen we de nadruk op de hogere orde problemen en veronderstellen we dat de lagere orde regelsystemen aanwezig zijn om hun problemen effectief oplossen. Met name concentreren we ons op het vraagstuk van het effectief transporteren van de DCV's door het netwerk. De vroegtijdig bagage-opslag is nog niet meegenomen, echter de aanwezigheid van zo'n opslag is wel gesimuleerd door het introduceren van enkele lussen waarin de vroegtijdig aan het netwerk aangeboden bagage opgeslagen kan worden.

Het op DCV gebaseerde bagage-afhandelingssysteem werkt als volgt: voor een gegeven dynamische vraag naar bagage en lege DCV's voor elk oplaadpunt, samen met een eenrichtingsnetwerk van rails, wordt de route voor elk DCV zodanig berekend onder de van toepassing zijnde veiligheids-en operationele condities, dat elk bagagestuk binnen een vooraf bepaalde tijd op het eindpunt arriveert. Op dit moment hebben de netwerken een simpele structuur, de DCV's worden door het netwerk vervoerd door middel van route schema's gebaseerd op "geprefereerde" routes. De schema's kunnen gewijzigd worden indien een vooraf gedefinieerde gebeurtenis optreedt. Echter, de hoeveelheid bagage verschilt van moment tot moment afhankelijk van externe factoren bijvoorbeeld het jaargetijde, tijdstip van de dag, het vliegtuigtype aan de gate en het aantal passagiers op de vlucht. In dit onderzoek wordt niet uitgegaan van vooraf gespecificeerde routes maar ontwikkelen en vergelijken we efficiënte regelmethodes die de optimale route tijdens veranderende omstandigheden rechtstreeks bepalen. We bestuderen met name de *voorspellende* en *heuristische* methodes die geïmplementeerd zijn in een *gecentraliseerde*, *gedecentraliseerde* en *gedistributeerde* me-

thode.

Men spreekt van een *gecentraliseerde* aanpak als er één oplossing wordt bepaald voor het hele systeem, bij een *gedecentraliseerde* aanpak worden de oplossingen lokaal bepaald zonder onderlinge communicatie en coördinatie. De aanpak is *gedistribueerd* indien de beslissingen lokaal worden genomen maar er ook sprake is van communicatie en coördinatie tussen aangrenzende regelaars. Verder wordt de suggestie gedaan om de route van elke DCV te bepalen aan de hand van een *hierarchische* regelstructuur bestaande uit twee lagen bestaande uit lokale schakelaars op het lagere niveau en een surveillerende regelaar op een hoger niveau. In dit raamwerk voorzien de schakelregelaars positie-instructies aan de schakelaars in het netwerk. De verzameling van schakelregelaars staat onder toezicht van een netwerkregelaar die als hoofdtaak de verschillende schakelregelaars van stroom instructies voorziet.

Het bepalen van een optimale route resulteert in een niet-linear, niet-convex, “mixed-integer” optimalisatie probleem. De rekentijd om deze vraagstukken op te lossen is dermate hoog dat het onoplosbaar is, “intractable”. Derhalve presenteren we een alternatieve methode om de complexiteit van het probleem te reduceren door het niet-lineare probleem als een lineair programmerings probleem met reële en integer variabelen (*mixed integer linear programming* — MILP) te definiëren. Het voordeel van deze MILP-problemen is dat de globale, optimale oplossing gevonden kan worden door efficiënte software algorithmen. De oplossing van het MILP-probleem kan dan direct dienen als beginconditie voor het oorspronkelijke optimalisatie probleem.

De prestaties van deze aanpak worden getoetst aan de hand van een “benchmark case study” waar de verschillende methodes toegepast en vergeleken worden.

Summary

In this thesis we focus on two specific transportation systems, namely postal automation and baggage handling.

Postal automation

During the last decades the volume of magazines, catalogs, and other plastic wrapped mail items that have to be processed by mail sorting centers has increased considerably. In order to be able to handle the large volumes of mail, state-of-the-art mail sorting centers are equipped with dedicated mail sorting machines. The throughput of a mail sorting machine is defined as the number of sorted mail items divided by the time needed to sort them. In this thesis we consider large letters of A4 size envelopes. Such mail items are called flats.

Briefly, a state-of-the-art automated flat sorting machine operates as follows: the flats are inserted into transport boxes by feeding devices; the boxes carry the pieces with constant speed and sort them into static destination bins according to the selected sorting scheme. The throughput of a basic system sketched above can be augmented by designing a system where the bottom part consisting of destination bins can move bidirectional with piecewise constant speed.

For the continuous sorting process we determine an event-driven model using simulation. In order to compute the speed of the bottom system that maximizes the throughput of this machine, we implement and compare several advanced control methods. In particular we first consider different variants of *optimal control* with gradually decreasing complexity, namely: optimal control with a piecewise constant speed on time intervals of variable length, optimal control with a piecewise constant speed on time intervals of constant length, optimal control with a constant speed. Next we also consider *model-based predictive control* (MPC) with a piecewise constant speed on time intervals of constant length. The proposed control methods are then compared for several scenarios.

Furthermore, we also analyze how the structural changes — the increased number of feeders, the variable position of inserting devices — and parameter changes — the increased maximal bound for the velocity of the top system of the flat sorting machine — influence the throughput of the automated flat sorting system.

Baggage handling

The continuous need for reduction of costs in the air transport industry and the rise of low-cost carriers require a cost effective operation of the airports. To this aim major efforts are now being invested in making the baggage handling systems at airports more intelligent by increasing automation and by including embedded sensors, actuators, and intelligent control units. As a result, modern baggage handling systems at large airports transport luggage in an automated way using destination coded vehicles (DCVs), which are unmanned carts that transport the bags at high speeds on a network of tracks. A baggage handling system consists of several parts: loading stations, unloading stations, a network of conveyors of single-direction tracks with several (local) loops (for loading, unloading, and temporary storage of DCVs), and the early baggage storing area, where the bags that enter the system too early can be stored for longer time periods (e.g., hours). All this results in a complex infrastructure with many interacting components and points at which control decisions have to be taken, requiring an adaptive, on-line management and control structure.

Typical issues in automated baggage handling systems are coordination of the processing units, time scheduling, scheduling of resources, route choice, controlling the transfers between different modes of transportation (e.g., conveyor belts and DCVs), and prevention of deadlocks and buffer overflows. At the same time, the control should aim at optimal throughput and processing times subject to various operational and other constraints (e.g., the bags should not be damaged, bags should arrive at unloading stations within prescribed time windows). Therefore, the operation of a DCV-based baggage handling system involves solving both low-level control problems e.g., coordination and synchronization when loading a bag onto a DCV and when unloading it to its end point, or velocity control of each DCV and higher-level control problems e.g., routing DCVs through the network. In this thesis, we focus on the higher-level control problems for DCV-based systems where we assume that the low-level controllers are present and efficiently solve the low-level control problems. In particular, we only focus on routing DCVs transporting bags through the network such that the performance of the system is maximized. The early baggage storage area is not yet considered, but in order to emulate its presence, one or more loops were included in which the bags that entered the network of tracks too early were kept.

The DCV-based baggage handling operates as follows: given a dynamic demand of bags and a buffer of empty DCVs for each loading station, together with the network of single-direction tracks, the route of each DCV has to be computed subject to operational and safety constraints such that each of the bags to be handled arrives at its given end point within a specific time window.

Currently, the networks have a simple structure, the DCVs being routed through the system use routing schemes based on preferred routes. These routing schemes can be adapted to respond to the occurrence of predefined events. However, the load patterns of the system are highly variable, depending on, e.g., the season, time of the day, type of aircraft at each gate, or the number of passengers for each flight. Therefore, in this thesis, we do not consider predefined preferred routes, but instead we develop and compare efficient control methods to determine the optimal routing in case of dynamic demand. In particular, we consider *predictive* and *heuristic* approaches implemented in a *centralized*, *decentralized*, and *distributed* manner — the control approach is said to be *centralized* if the overall solution is determined by a single controller, the control approach is said to be *decentralized*

if local control actions are computed by local controllers without any communication or coordination between these controllers, the control approach is said to be *distributed* if the local control actions are computed while considering also communication and coordination between neighboring controllers. Furthermore, in order to efficiently determine the route choice of each DCV we also propose a *hierarchical* control framework that consists of a 2-level control structure with local switch controllers at the lowest level and one higher supervisory controller. In this control framework, switch controllers provide position instructions for each switch in the network. The collection of switch controllers is then supervised by a so-called network controller that mainly takes care of the flow instructions for the switch controllers.

Computing the optimal route choice yields a nonlinear, nonconvex, mixed integer optimization problem. The computational efforts required to determine the optimal route choice are high, and therefore, solving this optimization problem becomes intractable in practice. Consequently, we also present an alternative approach for reducing the complexity of the computations by writing the nonlinear optimization problem as a *mixed integer linear programming* (MILP) problem. The advantage is that for MILP optimization problems solvers are available that allow to efficiently compute the global optimal solution. The solution of the MILP problem can then be used directly or as an initial starting point for the original optimization problem.

To assess the performance of the proposed control approaches and control frameworks, we consider a benchmark case study, in which the methods are compared for several scenarios.

Curriculum vitae

Alina N. Tarău was born on 23rd of June 1981, in Tecuci, Romania. She finished her pre-university education in 2000, at the National College "Bogdan Petriceicu Hasdeu", in Buzau, Romania, Computer Science specialization. After this, Alina N. Tarău studied Control Engineering and Automatic Control at the Technical University of Bucharest, in Romania, where she obtained the 5-year Engineer Diploma in 2005. Her graduation thesis was entitled "Methods of Controlling the Congestion in TCP Networks" and was carried out during the final semester of her studies, under the supervision of Dr.ir. Radu Ştefan. During the last year of her studies Alina N. Tarău also assisted in teaching the courses "Systems Theory" and "Optimization Techniques" at the faculty of "Control and Computer Science" of the Technical University of Bucharest.

Since 2006, Alina N. Tarău has been working on the Ph.D. project "Multi-Agent Control for Large-Scale Transportation Systems" at the Center for Systems and Control of the Delft University of Technology, in Delft, the Netherlands. The research of her Ph.D. project focused on two applications of transportation systems, namely postal automation and baggage handling. For both applications, after determining fast event-driven models, model-based control methods have been analyzed such that the overall performance of the mentioned systems is increased. This research has been performed under the supervision of Prof.dr.ir. Hans Hellendoorn and Prof.dr.ir. Bart De Schutter.

During her Ph.D. research, Alina N. Tarău obtained the DISC certificate for fulfilling the course program requirements of the Dutch Institute for Systems and Control. Since 2006, Alina N. Tarău has been a member of the Netherlands Research School for Transport, Infrastructure, and Logistics (TRAIL).

The research interests of Alina N. Tarău include hybrid systems, distributed control, predictive and model-based control, and optimization techniques.

TRAIL Thesis Series

The TRAIL Thesis Series is a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics.

Tarău, A.N., *Model-based Control for Postal Automation and Baggage Handling*, T2010/1, January 2010, TRAIL Thesis Series, the Netherlands

Knoop, V.L., *Road Incidents and Network Dynamics: Effects on driving behaviour and traffic congestion*, T2009/13, December 2009, TRAIL Thesis Series, the Netherlands

Baskar, L.D., *Traffic Control and Management with Intelligent Vehicle Highway Systems*, T2009/12, November 2009, TRAIL Thesis Series, the Netherlands

Konings, J.W., *Intermodal Barge Transport: Network Design, Nodes and Competitiveness*, T2009/11, November 2009, TRAIL Thesis Series, the Netherlands

Kusumaningtyas, I., *Mind Your Step: Exploring aspects in the application of long accelerating moving walkways*, T2009/10, October 2009, TRAIL Thesis Series, the Netherlands

Gong, Y., *Stochastic Modelling and Analysis of Warehouse Operations*, T2009/9, September 2009, TRAIL Thesis Series, the Netherlands

Eddia, S., *Transport Policy Implementation and Outcomes*: *the Case of Yaounde in the 1990s*, T2009/8, September 2009, TRAIL Thesis Series, the Netherlands

Platz, T.E., *The Efficient Integration of Inland Shipping into Continental Intermodal Transport Chains. Measures and decisive factors*, T2009/7, August 2009, TRAIL Thesis Series, the Netherlands

Tahmasseby, S., *Reliability in Urban Public Transport Network Assessment and Design*, T2009/6, June 2009, TRAIL Thesis Series, the Netherlands

Bogers, E.A.I., *Traffic Information and Learning in Day-to-day Route Choice*, T2009/5, June 2009, TRAIL Thesis Series, the Netherlands

Amelsfort, D.H. van, *Behavioural Responses and Network Effects of Time-varying Road Pricing*, T2009/4, May 2009, TRAIL Thesis Series, the Netherlands

Li, H., *Reliability-based Dynamic Network Design with Stochastic Networks*, T2009/3, May 2009, TRAIL Thesis Series, the Netherlands

Stankova, K., *On Stackelberg and Inverse Stackelberg Games & their Applications in the Optimal Toll Design Problem, the Energy Markets Liberalization Problem, and in the The-*

ory of Incentives, T2009/2, February 2009, TRAIL Thesis Series, the Netherlands

Li, T., *Informedness and Customer-Centric Revenue*, T2009/1, January 2009, TRAIL Thesis Series, the Netherlands

Agusdinata, D.B., *Exploratory Modeling and Analysis. A promising method to deal with deep uncertainty*, T2008/17, December 2008, TRAIL Thesis Series, the Netherlands

Kreutzberger, E., *The Innovation of Intermodal Rail Freight Bundling Networks in Europe. Concepts, Developments, Performances*, T2008/16, December 2008, TRAIL Thesis Series, the Netherlands

Taale, H., *Integrated Anticipatory Control of Road Networks. A game theoretical approach*, T2008/15, December 2008, TRAIL Thesis Series, the Netherlands

Li, M., *Robustness Analysis for Road Networks. A framework with combined DTA models*, T2008/14, December 2008, TRAIL Thesis Series, the Netherlands

Yu, M., *Enhancing Warehouse Performance by Efficient Order Picking*, T2008/13, October 2008, TRAIL Thesis Series, the Netherlands

Liu, H., *Travel Time Prediction for Urban Networks*, T2008/12, October 2008, TRAIL Thesis Series, the Netherlands

Kaa, E.J. van de, *Extended Prospect Theory. Findings on Choice Behaviour from Economics and the Behavioural Sciences and their Relevance for Travel Behaviour*, T2008/11, October 2008, TRAIL Thesis Series, the Netherlands

Nijland, H., *Theory and Practice of the Assessment and Valuation of Noise from Roads and Railroads in Europe*, T2008/10, September 2008, TRAIL Thesis Series, the Netherlands

Annema, J.A., *The Practice of Forward-Looking Transport Policy Assessment Studies*, T2008/9, September 2008, TRAIL Thesis Series, the Netherlands

Ossen, S.J.L., *Theory and Empirics of Longitudinal Driving Behavior*, T2008/8, September 2008, TRAIL Thesis Series, the Netherlands

Tu, H., *Monitoring Travel Time Reliability on Freeways*, T2008/7, April 2008, TRAIL Thesis Series, the Netherlands

D'Ariano, A., *Improving Real-Time Train Dispatching: Models, Algorithms and Applications*, T2008/6, April 2008, TRAIL Thesis Series, the Netherlands

Quak, H.J., *Sustainability of Urban Freight Transport. Retail Distribution and Local Regulations in Cities*, T2008/5, March 2008, TRAIL Thesis Series, the Netherlands

Hegeman, G., *Assisted Overtaking. An assessment of overtaking on two-lane rural roads*, T2008/4, February 2008, TRAIL Thesis Series, the Netherlands

Katwijk, R.T. van, *Multi-Agent Look-ahead Traffic Adaptive Control*, T2008/3, January 2008, TRAIL Thesis Series, the Netherlands