Delft University of Technology

Delft Center for Systems and Control

Technical report 11-036

Hierarchical model-based control for automated baggage handling systems*

A.N. Tarău, B. De Schutter, and H. Hellendoorn

If you want to cite this report, please use the following reference instead:

A.N. Tarău, B. De Schutter, and H. Hellendoorn, "Hierarchical model-based control for automated baggage handling systems," Chapter 15 in *Distributed Decision Making and Control* (R. Johansson and A. Rantzer, eds.), vol. 417 of *Lecture Notes in Control and Information Sciences*, London: Springer, ISBN 978-1-4471-2264-7, pp. 359–386, 2012.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: `https://www.dcsc.tudelft.nl`

*This report can also be downloaded via `https://pub.deschutter.info/abs/11_036.html`

Hierarchical Model-Based Control for Automated Baggage Handling Systems

A.N. Tarău, B. De Schutter, and H. Hellendoorn

Abstract This paper presents a unified and extended account of previous work regarding modern baggage handling systems that transport luggage in an automated way using destination coded vehicles (DCVs). These vehicles transport the bags at high speeds on a network of tracks. To control the route of each DCV in the system we first propose centralized and distributed predictive control methods. This results in nonlinear, nonconvex, mixed-integer optimization problems. Therefore, the proposed approaches will be expensive in terms of computational effort. As an alternative, we also propose a hierarchical control framework where at higher control levels we reduce the complexity of the computations by simplifying and approximating the nonlinear optimization problem by a mixed-integer linear programming (MILP) problem. The advantage is that for MILP problems solvers are available that allow us to efficiently compute the global optimal solution. To compare the performance of the proposed control approaches we assess the trade-off between optimality and CPU time for the obtained results on a benchmark case study.

1 Introduction

The state-of-the-art technology used by baggage handling systems at airports to transport the bags in an automated way incorporates scanners that scan the (elec-

A.N. Tarău
Delft University of Technology, Delft Center for Systems and Control, Mekelweg 2, 2628 CD Delft, The Netherlands, e-mail: a.n.tarau@tue.nl

B. De Schutter
Delft University of Technology, Delft Center for Systems and Control, Mekelweg 2, 2628 CD Delft, The Netherlands, e-mail: b.deschutter@tudelft.nl

H. Hellendoorn
Delft University of Technology, Delft Center for Systems and Control, Mekelweg 2, 2628 CD Delft, The Netherlands, e-mail: j.hellendoorn@tudelft.nl

Fig. 1 DCVs running on a network of tracks (Photo courtesy of Vanderlande Industries)

tronic) baggage tags on each piece of luggage, baggage screening equipment for security scanning, networks of conveyors equipped with junctions that route the bags through the system, and destination coded vehicles (DCVs). As illustrated in Figure 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. Note that the DCVs 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 very large (for these airports the conveyor systems are too slow, and therefore, a faster carrier is required for each bag).

In this chapter we consider a DCV-based baggage handling system. Higher-level control problems for such a system are route assignment for each DCV (and implicitly the switch control of each junction), line balancing (i.e., route assignment for each empty DCV such that all the loading stations have enough empty DCVs at any time instant), and prevention of buffer overflows. The velocity control of each DCV is a low-level control problem. Low-level controllers determine the velocity of each DCV so that a minimum safe distance between DCVs is ensured and so that the DCVs are held at switching points, if required. So, a DCV runs at maximum speed, $v^{\max}$, unless overruled by the local on-board collision avoidance controller. Other low-level control problems are coordination and synchronization when loading a bag onto a DCV (in order to avoid damaging the bags or blocking the system), and when unloading it to its end point. We assume the low-level controllers already present in the system, and we focus on the higher-level control problems of a DCV-based baggage handling system, in particular the route assignment of the DCVs.

Currently, the track networks on which the DCVs transport the baggage have a simple structure, with the loaded 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. 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 [3]. Also note that the first objective of a baggage handling system is to transport all the checked-in or transfer bags to the corresponding end points before the planes have to be loaded. However, due to the airport's logistics, an end point is allocated to a plane only

within a given time span before the plane's departure. Hence, 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. So, predefined routes are far from optimal. Therefore, we will not consider predefined preferred routes, but instead we will develop and compare efficient control methods to determine the optimal routing of the DCVs.

In the literature, the route assignment problem has been addressed to a large extent for automated guided vehicles (AGVs), see e.g., [10, 12]. Traditionally, the AGVs that execute the transportation tasks are controlled by a central server via wireless communication. Hence, the computational complexity of the centralized routing controller increases with the number of vehicles to be routed. In this context, [18] 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 routing 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 DCV routing problem has been presented in [4] where an analogy to data transmission via internet is proposed, and in [9] 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 [4], while the latter reference, [9], 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 [9] faces 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) for routing each DCV transporting bags to its end point. This paper integrates results of previous work [14, 16, 17], presents all the methods that previously proved to be efficient; and compares the obtained results for a benchmark case study over typical and extreme scenarios. Moreover, we address the trade-off between accuracy of the overall performance of the system and the total computational effort required to compute the optimal solution.

This chapter is structured as follows. In Section 2 we describe the automated baggage handling process. Next, in Section 3, we present the control objective which will be later on used when solving the DCV routing problem. Furthermore, in Section 4, we propose several control approaches for determining the optimal route for each bag through the baggage handling network. First we develop and compare centralized and distributed predictive methods that could be used to maximize the performance of the DCV-based baggage handling system. But these methods involve nonlinear, nonconvex, mixed-integer optimization problems that are 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 program-

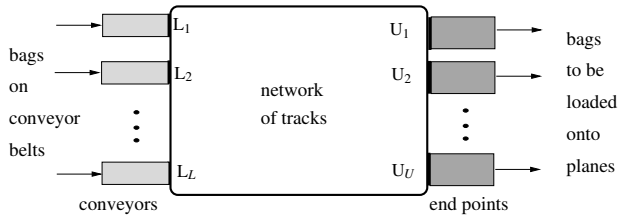

Fig. 2 Baggage handling system using DCVs

ming (MILP) optimization problem, for which solvers are available that allow us to efficiently compute the global optimal solution. This approach will then be incorporated in a hierarchical control framework for routing the DCVs. The analysis of the simulation results and the comparison of the proposed control methods and control frameworks are elaborated in Section 5. Finally, in Section 6, we draw conclusions and we present possible directions for future research.

2 System description and original model

Now we briefly recapitulate the event-driven route choice model of a baggage handling system that we have developed in [13]. 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.

Consider the general DCV-based baggage handling system with L loading stations and U unloading stations sketched in Figure 2.

The DCV-based baggage handling system operates as follows: given a demand of bags 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 the performance of the system is optimized.

The model of the baggage handling system we have developed in [13] consists of a continuous part describing the movement of the individual vehicles transporting the bags through the network, and of the following discrete events: loading a new bag onto a DCV, unloading a bag that arrives at its end point, updating the position of the switches into and out of a junction, and updating the speed of a DCV. The state of the system consists of the positions of the DCVs in the network and the positions of each switch of the network. According to the discrete-event model of [13], as long as there are bags to be handled, the system evolves as follows: we shift the current time to the next event time, take the appropriate action, and update the state of the system.

Let DCV_i denote the DCV that transports the ith bag that entered the track network up to the current time instant. 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 consider two models of the baggage handling system which will be used for (1) route control — where 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 — where we determine switch positions over the simulation period — respectively:

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

or

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

where:

- $\mathbf{t} = [t_1^{\text{load}} \dots t_{N^{\text{bags}}}^{\text{load}}\, t_1^{\text{unload}} \dots t_{N^{\text{bags}}}^{\text{unload}}]^\top$ with N^{bags} the number of bags to be handled in the given simulation period.
- $\mathscr{T}$ is the tuple that consists of the arrival times at loading stations for all the bags to be handled.
- $\mathbf{x}(t_0)$ 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 $1, 2, \dots, R$. Let $r(i) \in \{1, 2, \dots, R\}$ 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 $\mathscr{U} = (\mathbf{u}_1, \dots, \mathbf{u}_S)$ with $\mathbf{u}_s = [u_s^{\text{sw\_in}}(1) \dots u_s^{\text{sw\_in}}(N^{\text{bags}})\, u_s^{\text{sw\_out}}(1) \dots u_s^{\text{sw\_out}}(N^{\text{bags}})]^\top$ for $s = 1, \dots, S$, where S is the number of junctions and where $u_s^{\text{sw\_in}}(j)$ is the position of the switch into junction S$_s$ when the jth bag crosses S$_s$ and $u_s^{\text{sw\_out}}(j)$ is the position of the switch out junction S$_s$ when the jth bag crosses S$_s$.
  Without loss of generality (i.e., by creating virtual junctions connected by virtual links of zero length) we can assume each junction to have at most 2 incoming links (indexed by the labels 0 and 1) and at most 2 outgoing links (also indexed by 0 and 1). We call the switch that makes the connection between a junction and its incoming links a switch-in, and the switch that makes the connection between a junction and its outgoing links a switch-out.

The operational constraints derived from the mechanical and design limitations of the system are the following: the speed of each DCV is bounded between 0 and $v^{\max}$, while a switch at a junction has to wait at least τ^{switch} time units between two consecutive toggles in order to avoid the quick and repeated back and forth movements of the switch which may lead to mechanical damage. We assume τ^{switch} to be an integer multiple of τ_{s} where τ_{s} is the sampling time. In this chapter we denote the operational constraints by $\mathscr{C}(\mathbf{t}) \leq 0$.

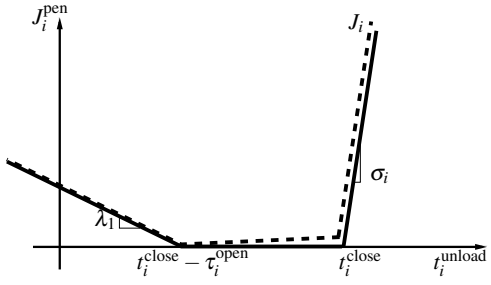

Fig. 3 Objective functions J_i^{pen} and J_i

3 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. Let N^{bags} be the number of bags that the baggage handling system has to handle. 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 i:

$$J_i^{\text{pen}}(t_i^{\text{unload}}) = \sigma_i \max(0, t_i^{\text{unload}} - t_i^{\text{close}}) + \lambda_1 \max(0, t_i^{\text{close}} - \tau_i^{\text{open}} - t_i^{\text{unload}}) \quad (1)$$

where t_i^{close} is the time instant when the end point of bag i closes and the bags are loaded onto the plane, σ_i is the static priority of bag 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 i is open for that specific flight. The weighting parameter $\lambda_1 > 0$ expresses the penalty for the additionally stored bags. Note that the control actions involved in $\mathbf{r}$ and $\mathscr{U}$ influence J_i^{pen} in the sense that they influence the time instant when bag i is unloaded. Moreover, all approaches that we propose have in common the fact that we are interested in t_i^{unload} with respect to the given unloading time window. Therefore, we have chosen t_i^{unload} as argument of J_i^{pen}.

Moreover, 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 3):

$$J_i(t_i^{\text{unload}}) = J_i^{\text{pen}}(t_i^{\text{unload}}) + \lambda_2(t_i^{\text{unload}} - t_i^{\text{load}}) \quad (2)$$

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:

$$J^{\text{tot}}(\mathbf{t}) = \sum_{i=1}^{N^{\text{bags,sim}}} J_i^{\text{pen}}(t_i^{\text{unload}}) \tag{3}$$

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 t_0 the initial simulation time and τ^{sim} is either the time instant when all the bags have been handled (and then $N^{\text{bags,sim}} = N^{\text{bags}}$) or $\tau^{\text{sim}} = \tau^{\text{max\_sim}}$ with $\tau^{\text{max\_sim}}$ the maximum simulation period.

4 Control methods

In this section we develop and compare centralized 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. The MILP approach will then be incorporated in a hierarchical control framework.

4.1 Centralized MPC

Since later on we will use model predictive control (MPC) for determining the routes of the DCVs in the network, in this section we first briefly introduce the basic concepts of MPC.

MPC is an on-line model-based control design method, see e.g., [11], that uses a receding horizon principle. In the basic MPC approach, given an horizon N, at step $k \geq 0$, where k is integer-valued, corresponding to the time instant $t_k = k\tau_{\text{s}}$ with τ_{s} the sampling time, the future control sequence $u(k), u(k+1), \ldots, u(k+N-1)$ is computed by solving a discrete-time optimization problem over the period $[t_k, t_k + N\tau_{\text{s}})$ so that a performance criterion defined over the considered period $[t_k, t_k + N\tau_{\text{s}})$ is optimized subject to the operational constraints. After computing the optimal control sequence, only the first control sample is implemented, and subsequently the horizon is shifted. Next, the new state of the system is measured or estimated, and a new optimization problem at time t_{k+1} is solved using this new information.

We define now a variant of MPC, where k is not a time index, but a bag index. If $k > 0$, then bag step k then corresponds to the time instant t_k^{load} when the kth bag has just entered the track network, while bag step $k = 0$ corresponds to the initial simulation time 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 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 changed later on.

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:

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

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.

Centralized MPC can compute on-line the route of each DCV in the network, but it requires a large computational effort as will be illustrated in Section 5. Therefore, we also propose 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.

4.2 Distributed MPC

One can decrease the computation time required by the *centralized* control approach proposed above by implementing a *distributed* approach that computes local control actions by solving local optimization problems similar to those that we have detailed in [15].

4.2.1 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 approaches, data is communicated between neighboring junctions, which are characterized by the concept of level of influence. The levels of influence are defined as follows.

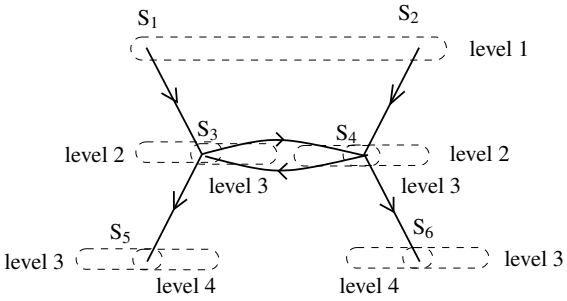

Fig. 4 Levels of downstream influence for parallel computation

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 $\kappa_{\mathrm{d}}^{\max}$ downstream influence levels are assigned to a given junction[1]. For example see Figure 4 where we define maximum 2 levels of downstream influence for each junction in the network ($\kappa_{\mathrm{d}}^{\max} = 2$). For this example we have considered the junctions S_1 and S_2 to have been assigned downstream influence level 1. Then S_3 and S_4 are assigned level 2 (since these junctions are connected to S_1 and S_2 via outgoing links). Next, we assign influence level 3 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 two levels of downstream influence: 2 and 3. Therefore, S_5 and S_6 are also assigned influence level 4 (since they are connected to S_3 and S_4 with influence level 3).

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 $\kappa_{\mathrm{u}}^{\max}$ levels of upstream influence are assigned to a given junction.

4.2.2 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 1 solves the local optimal switch control problem.

After computing the optimal switch control sequence, each junction with influence level 1 then communicates to its neighboring junctions at level 2 which bags

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

(out of all the bags over which we make the prediction for the corresponding junction with influence level 1) will enter the incoming link of the junction at level 2 and at which time instant they will do so. Next, we iteratively consider the junctions at levels $2,3,\ldots,K^{\text{downstream}}$, were $K^{\text{downstream}}$ is the largest level of downstream influence assigned in the network. Then, for each junction with influence level larger than 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 $\kappa_{\mathrm{d}} > 1$. Let $\mathrm{S}_{s,l}^{\text{prev}}$ denote the neighboring junction of S_s connected via the incoming link[2] $l \in \{0,1\}$ of S_s (so, $\mathrm{S}_{s,l}^{\text{prev}}$ 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

$$N_s = \min\left(N^{\max}, \sum_{l=0}^{1}\left(n_{s,l}^{\text{horizon}} + n_{s,l,0}^{\text{pred\_cross}} + n_{s,l,1}^{\text{pred\_cross}}\right)\right) \tag{4}$$

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 on link $l \in \{0,1\}$ going into S_s at the time instant when we start optimizing, 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 to continue their journey towards S_s ($n_{s,l,m}^{\text{pred\_cross}} < n_{s,l}^{\text{horizon}}$).

Let us now index[3] 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.

Recall 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 $t_{s,k}^{\text{cross}} \leq t_s^{\text{crt}} < t_{s,k+1}^{\text{cross}}$, where $t_{s,k}^{\text{cross}}$ 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 $k = 0$.

When solving the local MPC optimization problem for junction S_{s}, we will use a local objective function $J_{s,k,N_s}^{\text{Distr\_MPC}}$. 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:

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

where

- N_s^{cross} is the number of DCVs that actually cross junction S_s during the prediction period,

[2] Recall that we may assume without loss of generality that each junction has at most 2 incoming links.

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

- $\hat{t}^{\text{unload},*}_{s,k+j}$ is the predicted unloading time instant of bag $b_{s,k+j}$,
- 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 instant of bag $b_{s,k+j}$

The second term $\lambda^{\text{pen}}(N_s - N_s^{\text{cross}})$ 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 some DCVs travel on link $1-l$. If this term would not be considered, then $J^{\text{Distr\_MPC}}_{s,k,N_s}(\mathbf{t})$ would be minimal 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.

The MPC optimization problem at junction S_s for bag k is then defined as follows:

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

with N_s given by (4). Note that in this approach $\mathscr{M}^{\text{local,switch\_ctrl}}\big(\mathscr{T}, \mathbf{x}_s(t^{\text{cross}}_{s,k}), \mathbf{u}_s(k)\big)$ 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.

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 to nodes already present in the subtree. So, only the control of the switch-in and switch-out of the junctions in this subtree have to be updated.

4.2.3 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 one round of downstream and upstream communication.

So, every time a bag has crossed a junction we compute the local control sequences according to the downstream levels of influence as explained above. Then 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 $\kappa_{\text{u}} = 1$. 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 in the downstream direction. Let $t_s^{\text{last},*}$ be the time instant at which the last bag crossed S_s (out of all

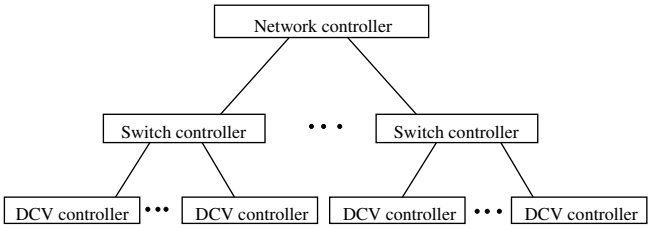

Fig. 5 Hierarchical control for DCV-based baggage handling systems

the bags over which we make the prediction for S_s). 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. Then if $t_s^{\text{last},*} < t_0 + \tau^{\text{rate}}$ we set $\zeta_{s,l} = \zeta^{\max}$ for $l = 0, 1$ with $\zeta^{\max}$ the maximum number of DCVs per time unit that can cross a junction using maximum speed. Otherwise, if $n_{s,l}^{\text{rate}} > 0$ with $n_{s,l}^{\text{rate}}$ the number of DCVs that left the outgoing link l of S_s within the time window $[t_s^{\text{last},*} - \tau^{\text{rate}}, t_s^{\text{last},*})$, we set $\zeta_{s,l} = \dfrac{n_{s,l}^{\text{rate}}}{\tau^{\text{rate}}}$. Finally, if $n_{s,l}^{\text{rate}} = 0$ we set $\zeta_{s,l} = \varepsilon$ with $0 < \varepsilon \ll 1$. Now we solve the local MPC problem presented in Section 4.2.2 using the updated release rates and we compute the local control of all junctions at upstream level $\kappa_u + 1$. Recursively, we compute the local control until level K^{upstream} where K^{upstream} is the largest level of upstream influence assigned in the network.

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.

4.3 Hierarchical MPC

In order to efficiently compute the route of each DCV we propose a hierarchical control framework that consists of a multi-level control structure, see Figure 5. The layers of the framework can be characterized as follows:

- The *network controller* considers flows of DCVs instead of individual DCVs. Moreover, the network controller determines reference DCV 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.

- 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 the switch of which 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.

4.3.1 Route control

We now focus on the network controller. In Section 5 it will be shown that when each DCV is considered individually, 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. So, since considering each individual DCV is too computationally intensive we will now consider streams of DCVs instead (characterized by real-valued demands and flows expressed in vehicles per second). In this paper the routing problem will then be recast as the problem of determining 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. This corresponds to blocking flows before a junction whenever necessary and possible, and routing the DCVs towards the outgoing links.

In the literature one can find extensive work addressing the flow-over-time problem, see e.g., [8]. However, in this paper we propose a non-standard, but efficient approach to model the flows of DCVs as presented next.

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. 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.

Furthermore, let the time instant t_k be defined as

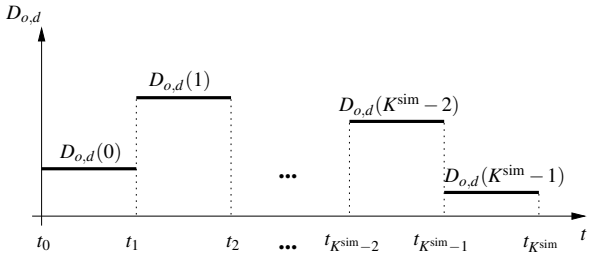

Fig. 6 Piecewise constant demand profile $D_{o,d}$

$$t_k = t_0 + k\tau^{\text{nc}}$$

with t_0 that time when we start the simulation and 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 6 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 $k = 0, 1, \ldots, K^{\text{sim}} - 1$ with K^{sim} the simulation horizon (we assume that beyond $t_{K^{\text{sim}}}$ 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 $d \in \mathscr{D}$ we have $\mathscr{L}_d^{\text{out}} = \emptyset$. Also, assume each origin node to have only one outgoing link and each destination node to have only one incoming link[4]. 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.

[4] If a loading station has more than one outgoing link, then one can virtually expand that loading station into a loading station connected via a link of length 0 to a junction with 2 outgoing links, etc.; similarly, one can virtually expand an unloading station with more than one incoming link.

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

$$\tau_\ell = \kappa_\ell \tau^{\mathrm{nc}} \quad \text{with } \kappa_\ell \text{ an integer.} \tag{5}$$

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:

$$u_{\ell,d}(k) \leqslant D_{o,d}(k) + \frac{q_{o,d}(k)}{\tau^{\mathrm{nc}}} \quad \text{for } \ell \in \mathscr{L}_o^{\mathrm{out}} \cap \mathscr{L}_d \tag{6}$$

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

$$q_{o,d}(k+1) = \max\left(0, q_{o,d}(k) + \left(D_{o,d}(k) - \sum_{\ell \in \mathscr{L}_o^{\mathrm{out}} \cap \mathscr{L}_d} u_{\ell,d}(k)\right)\tau^{\mathrm{nc}}\right) \tag{7}$$

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 in vertical queues before crossing the junction. 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 $d \in \mathscr{D}$. 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:

$$F_{v,d}^{\mathrm{out}}(k) \leqslant F_{v,d}^{\mathrm{in}}(k) + \frac{q_{v,d}(k)}{\tau^{\mathrm{nc}}} \tag{8}$$

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

$$F_{v,d}^{\mathrm{in}}(k) = \sum_{\ell \in \mathscr{L}_v^{\mathrm{in}} \cap \mathscr{L}_d} u_{\ell,d}(k - \kappa_\ell) \tag{9}$$

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

$$F_{v,d}^{\mathrm{out}}(k) = \sum_{\ell \in \mathscr{L}_v^{\mathrm{out}} \cap \mathscr{L}_d} u_{\ell,d}(k) \ . \tag{10}$$

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:

$$q_{v,d}(k+1) = \max\left(0, q_{v,d}(k) + \left(F_{v,d}^{\text{in}}(k) - F_{v,d}^{\text{out}}(k)\right)\tau^{\text{nc}}\right) \tag{11}$$

Moreover, for each origin $o \in \mathscr{O}$ and for each junction $v \in \mathscr{I}$ we have the following constraints:

$$\sum_{d\in\mathscr{D}} q_{o,d}(k+1) \leq q_o^{\max} \tag{12}$$

$$\sum_{d\in\mathscr{D}} q_{v,d}(k+1) \leq q_v^{\max} \tag{13}$$

where $q_o^{\max}$ and $q_v^{\max}$ express respectively the maximum number of DCVs the conveyor belt transporting bags towards loading station o can accommodate and the maximum number of DCVs the track segments of the incoming links of junction v can accommodate.

We also have the following constraint for every link ℓ:

$$\sum_{d\in\mathscr{D}} u_{\ell,d}(k) \leqslant U_\ell^{\max} \tag{14}$$

where $U_\ell^{\max}$ is the maximum flow of DCVs that can enter link ℓ.

Then, at time step k, the model of the DCV flows through the network of tracks describing (6)–(14) can be written as a system of equalities and a system of inequalities as follows:

$$\mathbf{q}_{k+1} = \mathscr{M}^{\text{eq}}(\mathbf{q}_k, \mathbf{u}_k)$$
$$\mathscr{M}^{\text{ineq}}(\mathbf{q}_{k+1}, \mathbf{u}_k) \leq 0$$

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 criterion

Next we define the performance 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

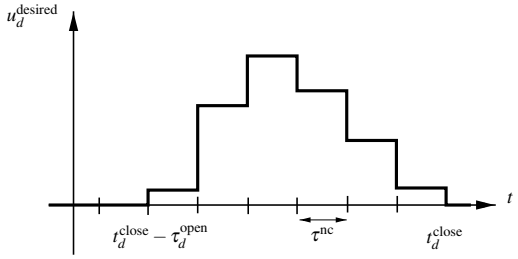

Fig. 7 Desired arrival profile at destination d

Figure 7, 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 $d \in \mathscr{D}$:

$$J_{d,k}^{\text{pen}} = \left| u_d^{\text{desired}}(k) - u_{\ell_d,d}(k + \kappa_{\ell_d}) \right|$$

where ℓ_d is the incoming link of destination d.

Later on we will include the penalty term $\sum_{i=k}^{k+N-1-\kappa_{\ell_d}} J_{d,i}^{\text{pen}}$ 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 $\sum_{i=k}^{k+N-1-\kappa_{\ell_d}} J_{d,i}^{\text{pen}}$ 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[5] 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:

$$J_{\ell,d,k}^{\text{flow}} = \begin{cases} u_{\ell,d}(k) \text{ if } k_d^{\text{open}} - \kappa_{l,d} \leq k < k_d^{\text{close}} - \kappa_{l,d} \\ 0 \qquad \text{otherwise} \end{cases}$$

Later on this penalty will be 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[5] time required for

[5] These durations are determined based on historical data.

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:

$$J_{v,d,k}^{\text{overdue}} = \begin{cases} d_{v,d}^{\min} q_{v,d}(k) & \text{if } k \geq k_d^{\text{close}} - \kappa_{v,d} \\ 0 & \text{otherwise} \end{cases}$$

where $d_{v,d}^{\min}$ represents the length of the shortest route from junction v to destination d. Note that $J_{v,d,k}^{\text{overdue}}$ is nonzero only for steps that are larger than or equal to $k_d^{\text{close}} - \kappa_{v,d}$. Moreover, for these steps $J_{v,d,k}^{\text{overdue}}$ is proportional to $d_{v,d}^{\min}$. The reason for this is that we want to penalize more the queues at junctions that are further away from destination d since the DCVs in those queues will need a longer time to travel to d.

Finally, let $\mathscr{L}^{\text{dest}}$ denote the set of links directly connected to unloading stations. Then the MPC performance criterion is defined as follows:

$$J_{k,N} = \sum_{d \in \mathscr{D}} \left(\sum_{i=k}^{k+N-1-\kappa_{\ell_d}} \lambda_d J_{d,i}^{\text{pen}} + \beta \sum_{i=k}^{k+N-1} \sum_{v \in \mathscr{I}} J_{v,d,i}^{\text{overdue}} - \alpha \sum_{i=k}^{k+N-1} \sum_{\ell \in (\mathscr{L} \setminus \mathscr{L}^{\text{dest}}) \cap \mathscr{L}_d} J_{\ell,d,i}^{\text{flow}} \right)$$

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:

$$\begin{aligned} \min_{\mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}, \mathbf{q}_{k+1}, \ldots, \mathbf{q}_{k+N}} \quad & J_{k,N} \\ \text{subject to} \quad & \\ & \mathbf{q}_{k+1} = \mathscr{M}^{\text{eq}}(\mathbf{q}_k, \mathbf{u}_k) \\ & \quad \vdots \\ & \mathbf{q}_{k+N} = \mathscr{M}^{\text{eq}}(\mathbf{q}_{k+N-1}, \mathbf{u}_{k+N-1}) \\ & \mathscr{M}^{\text{ineq}}(\mathbf{q}_{k+1}, \mathbf{u}_k) \leq 0 \\ & \quad \vdots \\ & \mathscr{M}^{\text{ineq}}(\mathbf{q}_{k+N}, \mathbf{u}_{k+N-1}) \leq 0 \end{aligned}$$

The nonlinear MPC optimization problem defined above is typically complex and it requires large computational effort to solve. Therefore, in the next section we will recast this problem into a MILP one for which efficient and fast solvers are available.

MILP optimization problem for the network controller

The general formulation of a mixed-integer linear programming problem (MILP) is:

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

$$\mathbf{x}^{\text{low}} \leq \mathbf{x} \leq \mathbf{x}^{\text{up}}$$
$$x_i \text{ is an integer, for each } i \in I$$

where $\mathbf{c}$, $\mathbf{x}$, $\mathbf{x}^{\text{low}}$, $\mathbf{x}^{\text{up}}$, $\mathbf{b}^{\text{eq}}$, and $\mathbf{b}$ are vectors, with $\mathbf{x}^{\text{low}}$ the lower bound for $\mathbf{x}$ and $\mathbf{x}^{\text{up}}$ the upper bound, and where $\mathbf{A}^{\text{eq}}$ and $\mathbf{A}$ are matrices (all these vectors and matrices have appropriate size), and $I \subset \{1,\ldots,n\}$ where n is the number of variables.

Next we transform the dynamic optimal route choice problem presented above into an MILP problem, for which efficient solvers have been developed [5]. To this aim we use the following equivalences, see [2], where f is a function defined on a bounded set X with upper and lower bounds M and m for the function values, δ is a binary variable, y is a real-valued scalar variable, and ε is a small tolerance (typically the machine precision):

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

$$\begin{cases} f(x) \leqslant M(1-\delta) \\ f(x) \geqslant \varepsilon + (m-\varepsilon)\delta \end{cases},$$

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

$$\begin{cases} y \leqslant M\delta \\ y \geqslant m\delta \\ y \leqslant f(x) - m(1-\delta) \\ y \geqslant f(x) - M(1-\delta) \end{cases}.$$

As example we will show how equation (7) of the nonlinear route choice model presented in the previous section can be transformed into a system of linear equations and inequalities by introducing some auxiliary variables. For the other equations of the route choice model we apply a similar procedure.

We consider now (7). This is a nonlinear equation and thus it does not fit the MILP framework. Therefore, we will first introduce the binary variables $\delta_{o,d}(k)$ such that

$$\delta_{o,d}(k) = 1 \text{ if and only if}$$
$$q_{o,d}(k) + \Big(D_{o,d}(k) - \sum_{l \in \mathscr{L}_o^{\text{out}} \cap \mathscr{L}_d} u_{\ell,d}(k)\Big)\tau^{\text{nc}} \leqslant 0 \tag{15}$$

and rewrite (7) as follows:

$$q_{o,d}(k+1) = \big(1-\delta_{o,d}(k)\big) \cdot \Big(q_{o,d}(k) + \big(D_{o,d}(k) - \sum_{l \in \mathscr{L}_o^{\text{out}} \cap \mathscr{L}_d} u_{\ell,d}(k)\big)\tau^{\text{nc}}\Big). \tag{16}$$

Condition (15) is equivalent to (cf. Property **P1**):

$$\begin{cases} f(k) \leqslant (q_o^{\max} + D_{o,d}^{\max}\tau^{\text{nc}})(1-\delta_{o,d}(k)) \\ f(k) \geqslant \varepsilon + (-U^{\max}\tau^{\text{nc}} - \varepsilon)\delta_{o,d}(k) \end{cases},$$

where $f(k) = q_{o,d}(k) + \big(D_{o,d}(k) - \sum_{l \in \mathscr{L}_o^{\text{out}} \cap \mathscr{L}_d} u_{\ell,d}(k)\big)\tau^{\text{nc}}$, $q_o^{\max}$ is the maximal queue length at origin o, and where $D_{o,d}^{\max} = \max_k D_{o,d}(k)$ is the maximal demand for origin-destination pair (o,d).

However, (16) is still nonlinear since it contains a multiplication of a binary variable $\delta_{o,d}(k)$ with a real-valued (linear) function. However, by using Property **P2** this equation can be transformed into a system of linear inequalities.

The rest of the model equations can be transformed, in a similar way, into a system of MILP equations. Next we will transform the MPC performance criterion into its MILP form.

The problem

$$\min \sum_{i=k}^{k+N-1} \sum_{d \in \mathscr{D}} \lambda_d \left| u_d^{\text{desired}}(i) - u_{\ell_d,d}(i + \kappa_{\ell_d}) \right|$$

can be written as:

$$\begin{aligned} \min \; & \sum_{i=k}^{k+N-1} \sum_{d \in \mathscr{D}} \lambda_d u_d^{\text{diff}}(i) \\ \text{s.t.} \; & \\ & u_d^{\text{diff}}(i) \geqslant u_d^{\text{desired}}(i) - u_{\ell_d,d}(i + \kappa_{\ell_d}) \\ & u_d^{\text{diff}}(i) \geqslant -u_d^{\text{desired}}(i) + u_{\ell_d,d}(i + \kappa_{\ell_d}) \\ & \text{for } i = k, \ldots, k+N-1. \end{aligned}$$

which is a linear programming problem.

If we add the MILP equations of the model, the nonlinear optimization problem of Section 4.3.1 can be written as an MILP problem.

Several efficient branch-and-bound MILP solvers [5] are available for MILP problems. Moreover, there exist several commercial and free solvers for MILP problems such as, e.g., CPLEX, Xpress-MP, GLPK, or lp_solve, see [1] for an overview. In principle, — i.e., when the algorithm is not terminated prematurely due to time or memory limitations, — these algorithms guarantee 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 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 will be illustrated in Section 5).

4.3.2 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

$u^{\text{opt}}_{\ell,d}(k),\ldots,u^{\text{opt}}_{\ell,d}(k+N-1)$ 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 $k=0,\ldots,K^{\text{sim}}$.

Recall that the optimal flows $u^{\text{opt}}_{\ell,d}(k),\ldots,u^{\text{opt}}_{\ell,d}(k+N-1)$ are determined for the time window $[t_k,t_{k+N})$ with $t_k=t_0+k\tau^{\text{nc}}$. In order to determine the switch control action during the time window $[t_k,t_{k+N})$ we will now again use 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[6] 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 a prediction period of N^{sc} steps. Hence, at each MPC step k^{sc}, the switch controller solves the following optimization problem:

$$\min_{\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}} \quad J^{\text{sw}}_{v,k^{\text{sc}},N^{\text{sc}}} \tag{17}$$

with $\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}=[s^{\text{in}}_v(k^{\text{sc}})\ldots s^{\text{in}}_v(k^{\text{sc}}+N^{\text{sc}}-1)\ldots\ s^{\text{out}}_v(k^{\text{sc}})\ldots\ s^{\text{out}}_v(k^{\text{sc}}+N^{\text{sc}}-1)]^\top$ if junction v has 2 incoming and 2 outgoing links and with $\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}}$ containing only switch-in or only switch-out positions if junction v has only 1 outgoing or only 1 incoming link respectively, and where the local MPC performance criterion $J^{\text{sw}}_{v,k^{\text{sc}},N^{\text{sc}}}$ is defined as:

$$\begin{aligned}J^{\text{sw}}_{v,k^{\text{sc}},N^{\text{sc}}}=\sum_{\ell\in\mathscr{L}^{\text{out}}_v}&\left|X^{\text{opt}}_{\ell,k,k^{\text{sc}},N^{\text{sc}}}(\mathbf{u}^{\text{opt}}_\ell)-X_{\ell,k^{\text{sc}},N^{\text{sc}}}(\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}})\right|\\&+\gamma\left(n^{\text{sw\_in}}_{k^{\text{sc}},N^{\text{sc}}}(\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}})+n^{\text{sw\_out}}_{k^{\text{sc}},N^{\text{sc}}}(\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}})\right)\end{aligned}$$

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 $u^{\text{opt}}_{\ell,d}(k),\ldots,u^{\text{opt}}_{\ell,d}(k+N)$ 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 (18)).
- $X_{\ell,k^{\text{sc}},N^{\text{sc}}}(\mathbf{s}_{v,k^{\text{sc}},N^{\text{sc}}})$ is the actual number of DCVs entering link ℓ during the prediction period. The variable $X_{\ell,k^{\text{sc}},N^{\text{sc}}}$ is determined via simulation for a nonlinear

[6] 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}.

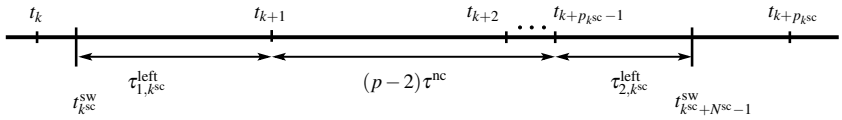

Fig. 8 Prediction window $\left[t^{\mathrm{sw}}_{k^{\mathrm{sc}}}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1}\right)$ over which we solve the MPC optimization problem (17) illustrated with respect to the window $[t_k, t_{k+p_{k^{\mathrm{sc}}}})$ for $p_{k^{\mathrm{sc}}} > 2$

(event-based) model similar to the one used in the distributed MPC approach of Section 4.2 (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_s DCVs to cross a junction);

- $n^{\mathrm{sw\_in}}_{k^{\mathrm{sc}},N^{\mathrm{sc}}}(\mathbf{s}_{v,k^{\mathrm{sc}},N^{\mathrm{sc}}})$ and $n^{\mathrm{sw\_out}}_{k^{\mathrm{sc}},N^{\mathrm{sc}}}(\mathbf{s}_{v,k^{\mathrm{sc}},N^{\mathrm{sc}}})$ represent the number of toggles of the switch-in and of the switch-out respectively during the time period $\left[t^{\mathrm{sw}}_{k^{\mathrm{sc}}}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}}\right)$ that are obtained from the simulation;
- 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 (17) 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 8 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 $[t_k, t_{k+p_{k^{\mathrm{sc}}}})$.

The variable $X^{\mathrm{opt}}_{\ell,k,k^{\mathrm{sc}}}(\mathbf{u}^{\mathrm{opt}}_{\ell})$ is given by:

$$\begin{aligned} X^{\mathrm{opt}}_{\ell,k,k^{\mathrm{sc}},N^{\mathrm{sc}}}(\mathbf{u}^{\mathrm{opt}}_{\ell}) = & \tau^{\mathrm{left}}_{1,k^{\mathrm{sc}}} \sum_{d\in\mathscr{D}} u^{\mathrm{opt}}_{\ell,d}(k) + \tau^{\mathrm{nc}} \sum_{i=k+1}^{k+p_{k^{\mathrm{sc}}}-2} \sum_{d\in\mathscr{D}} u^{\mathrm{opt}}_{\ell,d}(i) \\ & + \tau^{\mathrm{left}}_{2,k^{\mathrm{sc}}} \sum_{d\in\mathscr{D}} u^{\mathrm{opt}}_{\ell,d}(k+p_{k^{\mathrm{sc}}}-1) \end{aligned} \tag{18}$$

where $\sum_{i=k+1}^{k+j} x(i) = 0$ by definition for $j < 1$ and where

$\tau^{\mathrm{left}}_{1,k^{\mathrm{sc}}} = \min(t_{k+1}, t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1}) - t^{\mathrm{sw}}_{k^{\mathrm{sc}}}$ and $\tau^{\mathrm{left}}_{2,k^{\mathrm{sc}}} = \begin{cases} t^{\mathrm{sw}}_{k^{\mathrm{sc}}+N^{\mathrm{sc}}-1} - t_{k+p_{k^{\mathrm{sc}}}-1} & \text{if } p_{k^{\mathrm{sc}}} > 1 \\ 0 & \text{otherwise.} \end{cases}$

5 Simulation results

In this subsection we compare the performance of the centralized, distributed, and hierarchical MPC based on a simulation example.

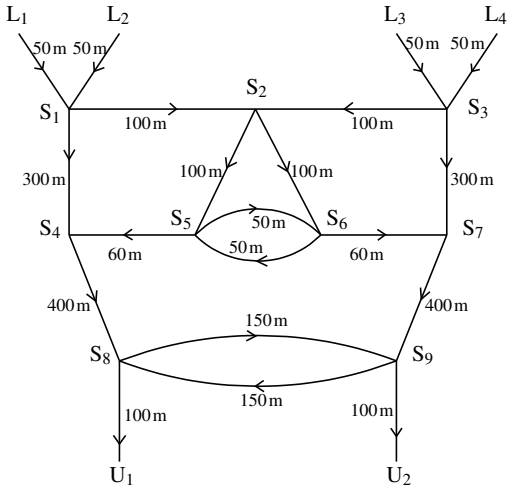

Fig. 9 Case study for a DCV-based baggage handling system

5.1 Set-up

We consider the network of tracks depicted in Figure 9 with four loading stations, two unloading station, nine junctions, and twenty unidirectional links. Note that this network allows 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[7], 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 9.

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.

We consider 6 typical scenarios where 2400 bags will be loaded into the baggage handling system (600 bags arrive at each loading station in the time interval $[t_0, t_0 + 100\,\mathrm{s})$). These scenarios include different classes of demand profiles for each loading station, different initial states of the system, queues on different links, and different time criticality measures (e.g., cases where the transportation of the bags is very tight, i.e., the last bag that enters the system can only arrive in time at the corresponding end point if the shortest path is used and its DCV is continuously running with maximum speed, or cases where the timing is more relaxed).

[7] The proposed control approaches allow the choice of routes containing loops.

Table 1 Comparison of average performance of the system and total computation time

Control approach	$J^{\text{approach,avg}}(s)$	total CPU time (s)
Centralized MPC	$1.13 \cdot 10^6$	$1.95 \cdot 10^5$
Distributed MPC downstream communication	$2.27 \cdot 10^7$	$3.90 \cdot 10^4$
Distributed MPC communication back & forth	$1.90 \cdot 10^7$	$1.46 \cdot 10^5$
Hierarchical MPC	$1.98 \cdot 10^5$	$1.06 \cdot 10^2$

5.2 Discussion

In order to solve the nonlinear, nonsmooth MPC optimization problem, one may use specialized search algorithms [6, 7] such as sequential quadratic programming algorithms, pattern search, genetic algorithms, and so on. We have chosen the genetic algorithm `ga` of the Matlab optimization toolbox *Genetic Algorithm and Direct Search* with multiple runs and "bitstring" population, since simulations show that this optimization technique gives good performance, with the shortest computation time.

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 considerably larger total computation time.

Let $J_j^{\text{tot,approach}}$ denote the performance of the baggage handling system corresponding to scenario index j and the considered control approach. Moreover, let $J^{\text{approach,avg}}$ denote the average performance:

$$J^{\text{approach,avg}} = \frac{1}{|\Delta|} \sum_{j \in \Delta} J_j^{\text{tot,approach}}$$

with Δ the set of considered scenarios. Then in Table 1 we list the average results.

Theoretically the performance of the baggage handling system obtained when using the *centralized* predictive switch control is better than when using the hierarchical approach. However, to obtain the true performance of centralized MPC would require extremely high computational time, see e.g., [13] where the CPU time is over 2 hours for routing 25 bags (on a network with only 2 junctions) when using a prediction horizon $N = 2$. Hence, to obtain the true optimum with centralized MPC requires a too high computational burden — centralized control becomes intractable in practice when the number of junctions is large due to the high computation time required. Therefore, the need to limit the computation time is required. 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).

The simulation results indicate that *distributed MPC* gives worse performance than centralized MPC. But this happens due to the time limitations that we have

imposed when solving the nonlinear optimization problems. Note that when using distributed MPC we ran the genetic algorithm once for each local optimization problem, while allowing a maximum of 3 generations of population. We have chosen these options in order to have a balance between the overall performance and the total computation time.

Regarding the hierarchical control approach we note that we have set the control time step for the network controller to 60 s, and the control time step for the switch controller was set to 2 s. The simulation results indicate that using the hierarchical control framework yields a better system performance than using the other predictive methods. But, recall that the solutions of centralized or distributed MPC 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 of centralized MPC and of distributed MPC with a single round of downstream and upstream communication is much larger than (over 40 hours) the one of the hierarchical control (an average of 100 s per junction, plus 6 s for solving the MILP optimization problems).

Hence, the advantage of using hierarchical MPC is clear: much better performance and much lower computational effort. To compute centralized or distributed MPC solutions in a more precise way one should route only a few bags on a very simple network. But then all approaches show the same performance, the only advantage of using the hierarchical framework is the low computation time.

6 Summary

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, we have developed and compared efficient control methods to determine the optimal DCV routing. In particular, we have developed and compared centralized, distributed, and hierarchical predictive methods to control the DCV routing.

In practice, centralized model predictive control (MPC) is not suitable for determining the optimal DCV routing due to the high computation time required to solve the route choice optimization problem. The simulation results indicate that distributed MPC yields a worse performance than centralized MPC when hard computational restrictions are imposed in solving the nonlinear optimizations. Simulation results also indicate that the hierarchical control with MILP flow solutions outperforms the other predictive control approaches where the multi-start local optimization method has been terminated prematurely.

In future work we will perform extensive simulations in order to assess the efficiency of these control approaches. Moreover, we will further improve the performance of the distributed MPC by considering multiple up and down rounds of optimizations and by extending the range of communication exchange to more than

one level. Also, in order to account for the increased computation and communication time of such an approach, we will extend the local control area to more than one node and assess the efficiency and the balance between performance and computation and communication requirements of such an alternative approach.

Acknowledgements This research has been supported by the VIDI project "Multi-Agent Control of Large-Scale Hybrid Systems" (DWV.6188) of the Dutch Technology Foundation STW, by the BSIK project "Next Generation Infrastructures (NGI)", by the Transport Research Centre Delft, by the Delft Research Center Next Generation Infrastructures, and by the European 7th framework STREP project "Hierarchical and Distributed Model Predictive Control of Large-Scale Systems" (contract number INFSO-ICT-223854).

References

1. A. Atamtürk and M. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140(1):67–124, November 2005.
2. A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.
3. R. de Neufville. The baggage system at Denver: Prospects and lessons. *Journal of Air Transport Management*, 1(4):229–236, December 1994.
4. 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.
5. 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.
6. C.A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, New York, USA, 1995.
7. P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London, UK, 1981.
8. A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science*, 379(3):387–404, June 2007.
9. 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.
10. 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.
11. J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, UK, 2002.
12. 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.
13. 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.
14. 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.
15. 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.

16. 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.
17. A.N. Tarău, B. De Schutter, and J. Hellendoorn. DCV route control in baggage handling systems using a hierarchical control architecture and mixed integer linear programming. In *Proceedings of the 3rd International Conference on Information Systems, Logistics and Supply Chain (ILS 2010)*, Casablanca, Morocco, April 2010.
18. 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.