
Beta
Research School for Operations
Management and Logistics
GENERIC CONTROL OF MATERIAL
HANDLING SYSTEMS
Sameh Haneyah

GENERIC CONTROL OF MATERIAL HANDLING SYSTEMS

Sameh Haneyah

Dissertation committee

Chairman & secretary	Prof. dr. R.A. Wessel
Promotor	Prof. dr. W.H.M. Zijm
Assistant promotors	Dr. J.M.J. Schutten
	Dr. P.C. Schuur
Members	Prof. dr. J. van Hillegersberg
	Prof. dr. M.J. Uetz
	Prof. dr. S.S. Heragu
	Prof. dr. M.B.M. de Koster
	Dr. Dipl.-Ing D. Spee

This thesis is number D174 of the thesis series of the Beta Research School for Operations Management and Logistics. The Beta research school is a joint initiative of the departments of Technology Management and Mathematics and Computing Science at the Eindhoven University of Technology, and the Center for Telematics and Information Technology at the University of Twente. Beta is the largest research center in the Netherland in the field of operations management in technology-intensive environments. The mission of Beta is to carry out fundamental and applied research on the analysis, design, and control of operational processes.

The work described in this thesis was performed at the Industrial Engineering and Business Information Systems group, Centre for Telematics and Information Technology, Faculty of Management and Governance, University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands.

Printed by Wöhrmann Print Service

The book cover is based on the *Denver International Airport Abstraction* painting by the American artist *Sharon Schock*, who kindly gave the permission to use the painting.

ISBN 978-90-365-0737-0

GENERIC CONTROL OF MATERIAL HANDLING SYSTEMS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
Prof. dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Friday, the 27th of September, 2013 at 12.45

by

Sameh Haneyah,
born on the 1st of November, 1983
in Ramallah, Palestine

This dissertation is approved by the promotor,
Prof. dr. W.H.M. Zijm

and the assistant promotors,
Dr. J.M.J. Schutten
Dr. P.C. Schuur

Table of Contents

Chapter 1

Introduction[1]

The material handling world is broad and diverse. We can observe material handling in many facets of modern economies: mail delivered in a postal system, bags moved in an airport, parts moved in a manufacturing system, pallet loads moved in a warehouse, containers handled by cranes at a sea port, trash collected in a waste management system, and goods moved by train. This thesis focuses on industrial sectors where systems operate within a certain facility, with material handling being the key function. Therefore, we exclude manufacturing facilities where material handling is not the key function but rather a support function.

Material handling systems (MHSs)[2] are in general complex installations that comprise various processes, such as inbound, storage, batching, sorting, picking, and outbound processes. Both the design and the control of these systems have received a lot of attention in research in various industrial sectors. However, there are, to the best of our knowledge, no reports on *generic* (i.e., sector independent) planning and control architectures and modeling approaches. In the literature, the perspective of the system's user is dominant; we often encounter studies dealing with systems in a particular airport or a distribution center of certain characteristics. Less attention is paid to a generic, broader perspective, which is interesting for the MHSs' supplier. In this thesis, we take the perspective of the MHSs' supplier, who produces a broad range of MHSs for which achieving as much synergy as possible is vital to facilitate design. We attempt to bridge the gap between practical requirements for generic control approaches and existing theory. To this end, we address questions that are not typically posted by MHSs' users, and are in fact interesting in the first place for the producer, apart from their scientific merits. In this context, we stress that this research is heavily motivated by the collaboration with a major global company supplying MHSs in all industrial sectors discussed in this thesis.

This chapter proceeds as follows: Section 1.1 discusses the research motivation in concrete terms. Afterwards, Section 1.2 outlines the scope of our analysis and presents the industrial sectors that we study throughout this thesis. Next, Section 1.3 formulates the generic MHS control problem, by contrasting MHSs in the different industrial

[1] *This chapter is based on Haneyah et al. (2013a).*
[2] *The terminology and list of abbreviations are provided at the end of this book.*

sectors and then analyzing their practical requirements in view of the generic control problem. As Section 1.3 formulates the problem and lists the requirements, Section 1.4 addresses theory by conducting a literature review in a search for answers from existing theory to the requirements from practice. Section 1.5 weighs the practical requirements against the theoretical knowledge and sets the research agenda. Finally, Section 1.6 presents the structure of the remainder of this thesis.

1.1 Research motivation

Currently, planning and control of MHSs are highly customized and project specific, which has important drawbacks for at least two practical reasons. From a system user point of view, the environment and user requirements of systems may vary over time, yielding the need for adaptation of the planning and control procedures. An adaptation may include implementing new control strategies or adjusting existing ones. From a systems' supplier point of view, an overall planning and control architecture that exploits synergy between the different industrial sectors (and at the same time is flexible with respect to changing business parameters and objectives) may reduce design time and costs considerably. Moreover, from a scientific point of view, finding a common ground to model MHSs in totally different industrial sectors and developing a generic control architecture that can be applied to MHSs in these different sectors presents a true challenge.

This thesis focuses on generic planning and control of automated MHSs, where we pay attention to a set of MHSs in three different industrial sectors:

- Baggage handling at airports, which we simply refer to as baggage handling.
- Distribution in warehouses, which we refer to simply as distribution.
- Parcel & postal sorting.

Planning and control of MHSs need to be robust and yield close-to-optimal systems' performance. Typical performance indicators concern throughput, lead time, and reliability. The aim of this research is to design a planning and control architecture that clearly describes the hierarchical framework of decisions to be taken at various levels, as well as the required information for decisions at each level (e.g., from overall workload planning to local traffic control). The planning and control architecture should be at low costs, flexible, easy to maintain, easy to implement, allowing for easy adaptation to configuration changes, changes in performance criteria, different operational modes, and adjustment of the control strategies.

In this context, we emphasize that our focus is on control architectures and not on software architectures. Although the advantages and disadvantages of centralization versus decentralization in both domains are very much alike, we have to make a distinction because, e.g., a decentralized control architecture can be implemented by a single-tier software architecture and vice versa.

1.2 Scope of analysis and industrial sectors

In this section, we outline the scope boundaries (Section 1.2.1), and then describe the industrial sectors under study in more detail (Section 1.2.2).

1.2.1 Project scope

Figure 1.1 shows three possible scopes of analysis, along with the party mainly responsible for decision-making within each scope, i.e., the MHS's user or the MHS's supplier. Scope boundaries are as follows:

- Scope 3 is the widest, taking the whole logistic network into account. In this scope, decisions have a global impact and involve many stakeholders. An example of a problem within this scope is the facility location problem of depots within a logistic network, in order to optimize transportation costs. Another problem to deal with is how to plan the flow between network nodes in order to minimize costs while satisfying supply and demand constraints.
- Scope 2 focuses on a single site in the network. It includes inbound and outbound operations at the site of the MHS's user. Scheduling these operations is done by the MHS's user. However, the MHSs' suppliers may consider the extension of their services to offer scheduling tools to the MHS's user that can result in better system performance. An example is scheduling inbound containers that are waiting at a parcel sorting hub in order to make the operation of the MHS more efficient.
- Scope 1 focuses on the control of the MHS. The supplier is entirely responsible for decisions within this scope, as the (built-in) control architecture of the MHS is the relevant element here, i.e., the software running the automated MHS. Decisions within this scope have a local impact.

In this thesis we exclude Scope 3, because the focus then shifts towards network optimization. A shift towards network optimization will limit the attention paid to the internal system within a single facility in the network, i.e., the MHS, which is our main area of interest. The focus is on the control of large MHSs, which is mostly the analysis within Scope 1. However, we may have to deal with problems at Scope 2, which are closely related to the operation of the MHS, e.g., container scheduling to alleviate peak loads in MHSs. Solving this problem needs real-time information from the MHS, e.g. on the nature of the load in transport within the MHS. Chapter 2 provides more details on the problems we work on in this thesis.

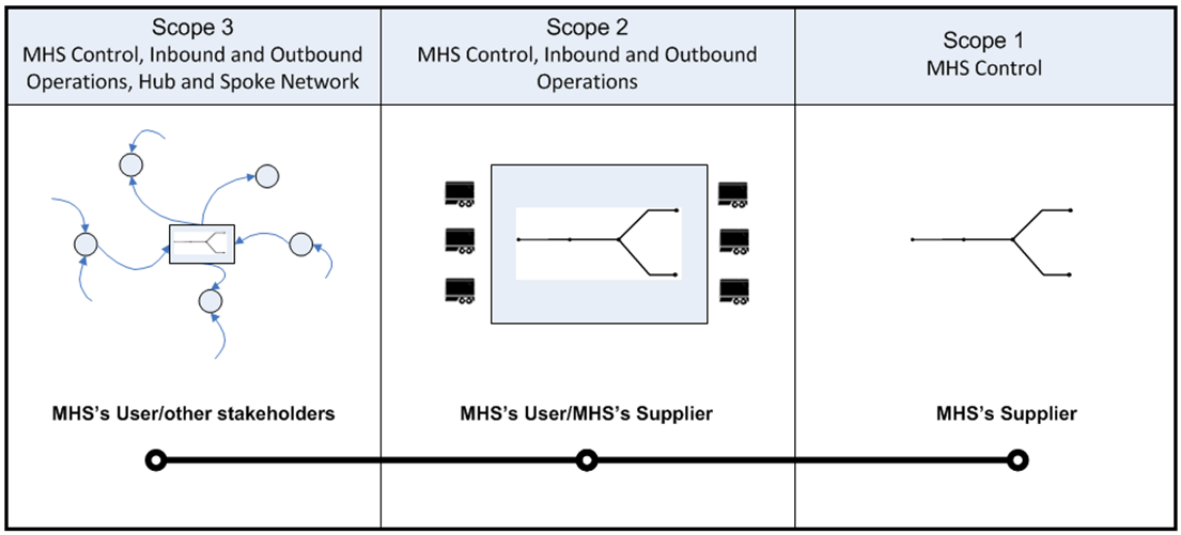


Figure 1.1. Scope boundaries.

In this thesis, we select MHSs from three industrial sectors to be the starting point of our analysis. The selected MHSs, which will be described in subsequent chapters, include the main (automated) operations of the logistic processes within the industrial sectors studied. We are actually interested in the intersection among industrial sectors where opportunities of generic control might be lost, and are less interested in the obvious differences. In other words, we focus on systems in different industrial sectors that are similar in terms of the equipment, but are using different control methods and work in different environments. Therefore, the analysis and findings are likely to be dependent on the initial selection of reference MHSs. However, we will analyze variants and extensions of the modeled MHSs, and try to propose flexible generic control methods that can apply to other MHSs than the ones this thesis concentrates on.

1.2.2 Industrial sectors

This section addresses three different industrial sectors using MHSs. The aim is to gain insight into the requirements and functionalities of MHSs in these sectors. Our scope of analysis is restricted to the built-in control of the MHS that is within the responsibility of the MHS supplier, not the MHS user.

1.2.2.1 Parcel & postal sorting

In parcel & postal sorting, systems are typically used by express parcel carriers, such as DHL[3], UPS[4], and TNT[5], to receive items coming to a hub from various sources, and then sort them according to destination, in preparation for further transport. In this business, as the quantities to be handled grow, manual operations fall short. Thus, the need for automated sorting systems, or simply *sorters*, is evident. Such systems can be seen in various forms and capabilities to meet the specific demands of system users. The term parcel is used throughout this thesis as the main item handled within these systems. However, other items, such as *totes*, can be handled by the same sorters as we clarify later on. Figure 1.2 shows the generic scheme of a simple sorter.

The process starts at the *unload area*, where containers carrying parcels arrive at the system via airplanes or trucks. Operators unload the containers and place the parcels on the infeed conveyors (or simply *infeeds*). These infeeds transport the parcels to the *main conveyor* represented by the big loop in Figure 1.2. The merge operation takes place when the parcels transported on the infeeds reach the main conveyor. Once they are on the main conveyor, the parcels are transported until they reach the *load area*. In this area, parcels are automatically directed to their destinations, based on parcel identification labels. Parcels are released into special *outfeed* conveyors called sorting *chutes* (see Figure 1.2). At the end of these chutes, operators gather the parcels in containers. In the layout given in Figure 1.2, some parcels may flow back into the

[3] *Acronym that stands for Dalsey, Hillblom and Lynn (surnames of the founders of this Company).*
[4] *Acronym that stands for United Parcel Service.*
[5] *Acronym that stands for Thomas Nationwide Transport.*

unload area, which means that they have passed the load area without being sorted. This may happen when the chutes are full or when there is some disruption in the system. Such a system is therefore referred to as a *closed-loop sorting system*, or *loop sorter*. Note that the system depicted in Figure 1.2 is a relatively simple one; larger and more complex systems can entail several load and unload areas, multiple loops, more complex layouts, etc. Such complex systems may provide alternative routes to reach a certain destination (chute).

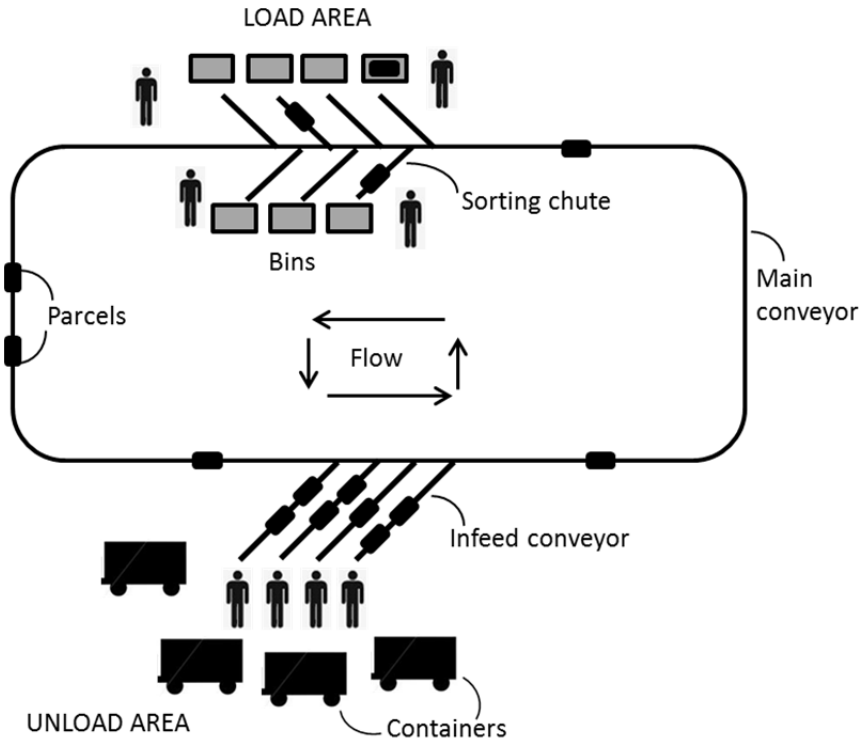


Figure 1.2. Generic scheme of a closed-loop parcels sorting system.

A parcel sorting hub operates at full power in specific time intervals, mostly during night-time. Normally, tons of parcels (and envelopes) are delivered, sorted, and transported within a few hours. In these rush-hour conditions, the main objective is to maximize throughput of the systems, in order to minimize the time period between the arrival and departure times of planes or trucks. This may result in some other functional requirements that may bring more efficiency to the process, e.g., balancing material flows within the system.

1.2.2.2 Baggage handling

We focus on baggage handling systems (BHSs) in airports. Baggage handling is a sector that differs from the other industrial sectors in the involvement of multiple stakeholders. These stakeholders include: the airport (main customer), airlines and handlers (parties using the BHS), security, and customs. The latter two are external parties that impose restrictions on the operation of the BHS. In other sectors, e.g., distribution, the warehouse operator is the main stakeholder. There, the MHS's supplier can build and deliver a system completely according to the stakeholder's

requests. However, in baggage handling the different stakeholders all influence the system design; this makes it challenging to satisfy the interests of all stakeholders.

In a BHS, the bag as the main item treated belongs to one of three possible categories (see Figure 1.3). On a generic level, first a bag may belong to a passenger who arrives at the airport and has a departing flight to catch. Second, it may belong to a transit passenger who lands on the airport and has a connecting flight to catch. Finally, a bag may belong to a passenger for whom the airport is his or her final destination. In a BHS, there is an Early Bags Storage (EBS), where bags that arrive early to the system are temporarily stored.

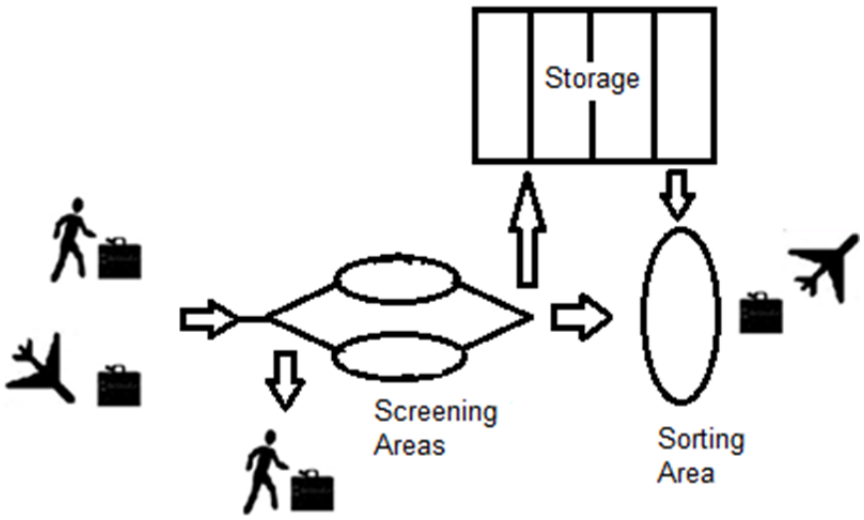


Figure 1.3. Generic scheme of a baggage handling system.

The purpose of a BHS is to deliver each bag from some source point A to some destination point B, within a specific time limit. However, the airport environment of a BHS is highly dynamic and stochastic, which complicates the delivery job, and generates additional challenges. Moreover, every stakeholder has its own desires, which affect its criteria for assessing the BHS. A main performance measure for BHS is the *irregularity rate*. The irregularity rate is the number of bags (per 1000) that are supposed to be on a certain plane but are not (luggage that missed the correct plane, and lost luggage). From a practical point of view, minimizing the irregularity rate is most challenging when dealing with connecting flights. This is because several things may go wrong when trying to correctly deliver an arriving bag to the next connecting plane within a given (often short) time window. Problems may arise from: wrong or corrupted bag tags, planes arriving late, disruptions in the BHS causing bags to miss their connecting flight, etc. As a result, the main objective for a BHS is to minimize the irregularity rate. An important system design parameter is the *in-system time*. This is the time a bag needs to travel along the longest path between the input and output points that are farthest apart in the BHS. This measure does not account for manual operations such as manual coding of bags when bag tags are found corrupted.

Within the BHS, an important attribute of each bag is the urgency measure in terms of the time left for the departure of its corresponding flight. Urgent bags have the highest priority to move to the intended destination as the time window available for them is the smallest. As time goes by, non-urgent bags become urgent. Business class bags

have a priority when loading and unloading the plane, but they do not affect the urgency classification.

A BHS is a complex system consisting of several routes of transportation by different possible means such as conveyors and *Destination Coded Vehicles (DCVs)*. The system includes different resources, e.g., screening machines, and redundant transport systems to ensure high availability. Therefore, there are different possible routings to realize the transport operation. The logistic control of this system must use the resources in a way that optimizes the bag's flow time in the system (Section 1.3.2 discusses other relevant requirements). To sum up, the general high level objective for the control architecture of BHSs is to minimize the irregularity rate. This is done by completing the overall transportation operation within the time limits, which requires a smooth process that is able to avoid disruptions or congestion that may result in bags missing their corresponding flights.

1.2.2.3 Distribution

The distribution sector concerns the MHSs used in warehouses and distribution centers to handle various types of products for various customers. In distribution, projects vary considerably in terms of user requirements and the variety of system designs and operational approaches that can be implemented. However, for all systems the generic set of ordered activities in a distribution center (DC) are as follows: Receiving, Storage, Order Picking, Consolidation, and Shipping. Moreover, Cross Docking is an operation in which the DC acts merely as a material handler without intermediate storage. Figure 1.4 shows a schematic view of a warehouse with a goods receiving area, a storage area, an order picking area (with three *pick station*s), and a consolidation area. For storage areas, automated storage and retrieval systems (ASRSs) are often used. An ASRS consists of a number of parallel racks and a number of cranes operating in the aisles between these racks. We will study these systems in more detail in subsequent chapters (see Section 3.1.1 for a more detailed illustration). In order to study MHSs with common equipment among the three industrial sectors, the distribution systems we study use mainly ASRSs and loop conveyors with pick stations.

In this sector, the general purpose is to satisfy the orders in time and with good quality, given time, cost, and other operational constraints. In order to satisfy orders properly within a certain time frame, a high throughput of the MHSs is a main objective. At each process stage in these systems, there normally is a set of parallel stations performing the same tasks, for example, parallel order pick stations, parallel cranes, etc. Therefore, it is crucial to balance the workloads within the system. There should be a generic control approach that entails generic algorithms, allowing for applications in different types of systems. However, the current control of MHSs in distribution centers is highly customized and often includes quite a number of relatively complicated rules to realize as much throughput as possible at the MHS.

As a general remark, according to observations from practice, there is an increasing interest from system architects, towards control solutions that are more robust and generic, at the expense of sacrificing the maximum attainable throughput from MHSs.

This is due to certain design and operational requirements that we explain in Section 1.3.2.

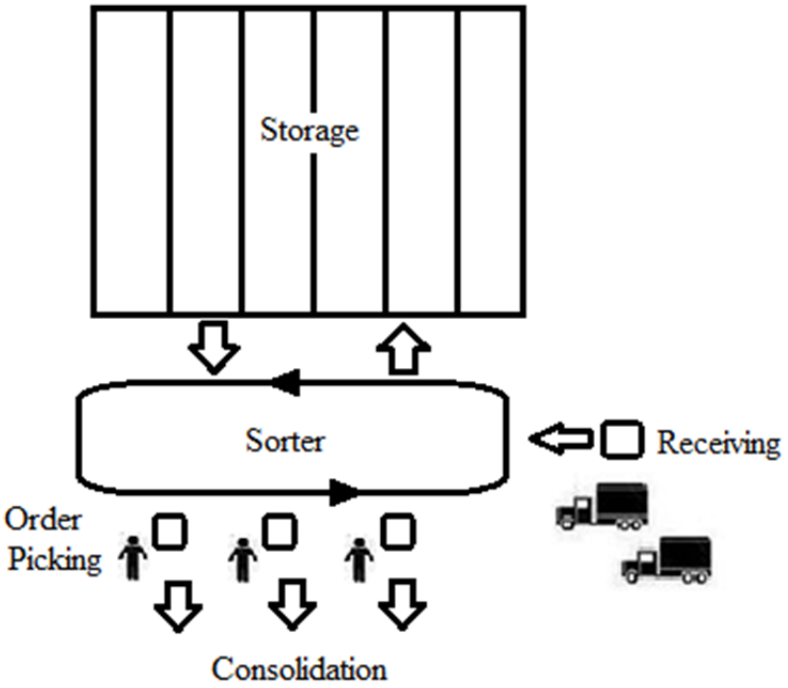


Figure 1.4. Generic scheme of a distribution center/warehouse.

1.3 Problem formulation

In this section, we contrast MHSs in the different industrial sectors (Section 1.3.1) and then we define the common requirements for a generic control architecture (Section 1.3.2).

1.3.1 Contrasting MHSs in the different industrial sectors

Different industrial sectors imply different MHS's user environments and requirements. However, we take the challenge to deal with the differences in order to model the MHSs in different sectors in a generic way that maximally exploits synergies. A first impression from the general study of these different sectors tends to suggest a certain level of synergy among them. MHSs in baggage handling and parcel & postal sorting seem to have more similarity with each other than with MHSs in distribution. In the following, we list the main similarities of these two sectors, and at some points we indicate how the distribution sector differs:

- Routing parcels or bags within the system can be complex and with more than one route to go from one point to another.
- Compared to MHSs that we study in distribution, the time pressure is higher in BHSs and parcel & postal sorting systems, as is reflected in the necessity to deliver the items to their intended destinations in time to meet strict deadlines.
- Unpredictable arrivals: in baggage handling, there is no information ahead on the type, number or weight of bags from check-in passengers. For parcel & postal sorting and transit bags, information is in the network but not used to

plan the operations. In distribution, there are planned goods to receive with known quantities and arrival times, so the distribution center can plan operations ahead.

- Item integrity: the bag or parcel enters and leaves a BHS or a parcel & postal sorting system in the same form and with the same characteristics or attributes. On the other hand, in distribution, pallets are broken into *product totes*, and these product totes are handled within the material handling system. The unit transported by the MHS may be the same, i.e., totes, but the characteristics of the tote change. A product tote changes, e.g., when some items are picked from it, and becomes part of a reverse flow that goes back from pick stations to the storage area.
- Items uniqueness: a parcel or a bag is a unique item in a BHS or a parcel & postal sorting system and is required for a certain plane or truck. However, in distribution there are multiple alternatives for a certain item. If an order requires one unit from item x, there may be several totes containing item x. There is a choice from which tote to pick.
- Unit handled: in baggage handling and parcel & postal sorting, the bag or parcel is normally picked, stored, and transported throughout the MHS. In this sense, bags or parcels are single *unit loads*. However, in distribution, there may be a different definition of the unit load, which implies a number of items to be handled together and usually supported by a handling device such as a pallet, case or tote.
- Heterogeneous items: bags and parcels may be of different shapes, weights, dimensions, which affects the conveyability on an MHS. However, in a distribution center there are normally standardized unit loads.

In the distribution sector, the synergy on a higher level may be less apparent, especially due to the high variety in implemented systems. However, based on the study of some distribution centers in practice, we observe synergy on a subsystem level in terms of physical components. Direct examples are:

- The storage in the ASRS system is analogous to the Early Bags Storage in baggage handling (note that such systems are not used in parcel & postal sorting due to the absence of a storage function). The physical system is similar in these two sectors, but there are storage rules in distribution centers that determine where an item is stored, based on criteria such as item availability in aisles. On the other hand, for baggage handling during peak times, the main concern is to store all bags that need storage as fast as possible without considering storage rules and anticipating the balance of picking from different *storage aisles*, where a storage aisle is defined between two storage racks. These functional issues raise challenges for developing a generic storage and retrieval strategy that can be used by both sectors. Finally, the unit of storage in baggage handling is a bag, whereas in distribution there are storage concepts for totes, pallets, cartons, etc. and the picking operation differs accordingly.
- Sorting systems: the backbone of the MHSs in parcel & postal sorting is the sorting system, consisting of sorters, which are generally characterized by a few inputs, many outputs, and high speed. However, similar systems may be a sub-

system in the other two sectors. We will call similar systems also sorters for modeling purposes. In distribution, products arriving to be stored are normally merged on a conveyor loop that leads totes to storage aisles. In this context, guiding a tote to its destined storage aisle is a sorting operation that is similar to guiding the parcel to its destined sorting chute. *Broken totes*, which are totes that are picked from but still contain items, return from order pick stations and subsequently merge on the conveyor loop that leads totes back to storage, which is again similar to the merge operation in parcel & postal sorting. In the other direction, totes leave the storage aisles to go to the pick stations; this transport operation sorts totes to destined pick stations as well. In baggage handling, sorters are also used for sorting bags to, e.g., parallel screening machines or *laterals*[6].

We believe it makes sense to provide a generic material flow model to explain the processes in the different sectors. The model entails generic process stages, which should cover all possible operations of MHSs in practice. Therefore, we propose the material flow terminology of the most complex sector in terms of operations or process stages, which is distribution. MHSs in distribution entail some complex and more detailed operations than the other two sectors, e.g., the order picking operation that changes the characteristics of handled items. Our claim is that any operation in the other two sectors can be mapped to one of the operations in the distribution sector. Transportation channels may be more complex in BHSs, but this is a matter of transportation complexity, not operational variety. Figure 1.5 presents a generic material flow model, together with a tabulated description of process stages, based on the analysis of selected reference sites from the different industrial sectors in practice. The model divides the physical flow into six process stages. In each stage, there is a set of resources modeled in abstract terms as workstations. This model lists resources and indicates transportation possibilities, but no explicit transportation routes.

1.3.2 Common requirements of MHSs/control architecture

The objective of this thesis is to develop a generic control architecture that can be applied to various types of MHSs. The challenge for a generic control architecture lies in its ability to satisfy the objectives of different sectors. Therefore, we first look at the objectives of MHSs in different sectors to decide whether a generic control architecture can be achieved.

We define a set of generic requirements for an appropriate control architecture, in which we discern *functional* and *design* requirements. Functional requirements are the key performance indicators *(KPIs)* for MHSs. Design requirements are the basic characteristics of a control architecture from development, implementation, and maintenance perspectives. In this section, we first discuss functional requirements, followed by design requirements. At a system level, there are two important functional objectives that serve as KPIs for MHSs in all sectors:

[6] *A type of outfeed conveyors used in baggage handling to gather bags in preparation for loading on planes.*

Stage	Sector	Description
Receiving	Parcel & Postal Sorting	Containers are broken into parcels. In our scope of analysis, the arrival of parcels to the system is uncontrollable. Workstations (depending on the level of aggregation) are merge areas or infeed conveyors on which parcels are loaded.
	Baggage Handling	Containers/ULDs coming from arriving planes are broken into bags. The arrival of bags to the system is uncontrollable. Workstations are: (a) check-in desks where bags are brought by departing passengers, (b) loading belts for transfer bags coming from arriving planes.
	Distribution	Containers are broken into totes or items. Workstation is the pallet(s) unloading point(s) of incoming trucks. Unloading can be done by an operator, but can also be automated (e.g., robots).
Quality Control	Parcel & Postal Sorting	In the MHSs of some air hubs, each parcel has to pass through one of a set of parallel screening machines. In this case, workstations are screening machines.
	Baggage Handling	Each bag has to pass through one of a set of screening machines, which are the workstations.
	Distribution	Quality control occurs in distribution centers to check the conformance of incoming materials to the requirements and specifications. Workstations are operators checking the incoming materials.
Storage	Parcel & Postal Sorting	Normally, there is no storage operation in parcel & postal sorting systems.
	Baggage Handling	Early bags are stored in the ASRS. Workstations here can be cranes operating in storage aisles where bags are stored.
	Distribution	The ASRS represents the storage area. Workstations can be the cranes operating in storage aisles where items are stored.
Order Picking	Parcel & Postal Sorting	Parcels to be sorted to different destinations are picked at the chutes, so (depending on the level of aggregation) we model sort areas or the chutes as order picking stations. Parcels for different destinations are transported simultaneously and then sorted to the assigned chutes.
	Baggage Handling	Bags for departing flights are picked at the laterals, so we model the laterals as order picking stations.
	Distribution	In goods-to-man systems, (manned) pick stations represent the workstations. Totes for different orders are retrieved simultaneously and then sorted to the assigned pick station(s). An item can be picked from a tote to fulfill a customer order. If the item is not the last one in the tote, then the tote is routed back to the ASRS.
Consolidation	Parcel & Postal Sorting	Several chutes can be assigned to a certain destination. Parcels with same destination are consolidated and assigned to containers. Thereafter, containers going to the same destination by a truck/plane are consolidated. Workstations are loading operators (robots).
	Baggage Handling	ULDs for the same flight and coming from different laterals are consolidated. Workstations are loading operators (robots).
	Distribution	Totes for the same order and coming from different pick stations or zones, are consolidated and prepared in (pallets) for shipment. Workstations are loading operators (robots).
Shipping	All Sectors	Consolidated ULDs/containers for a certain truck/plane are loaded for shipping.

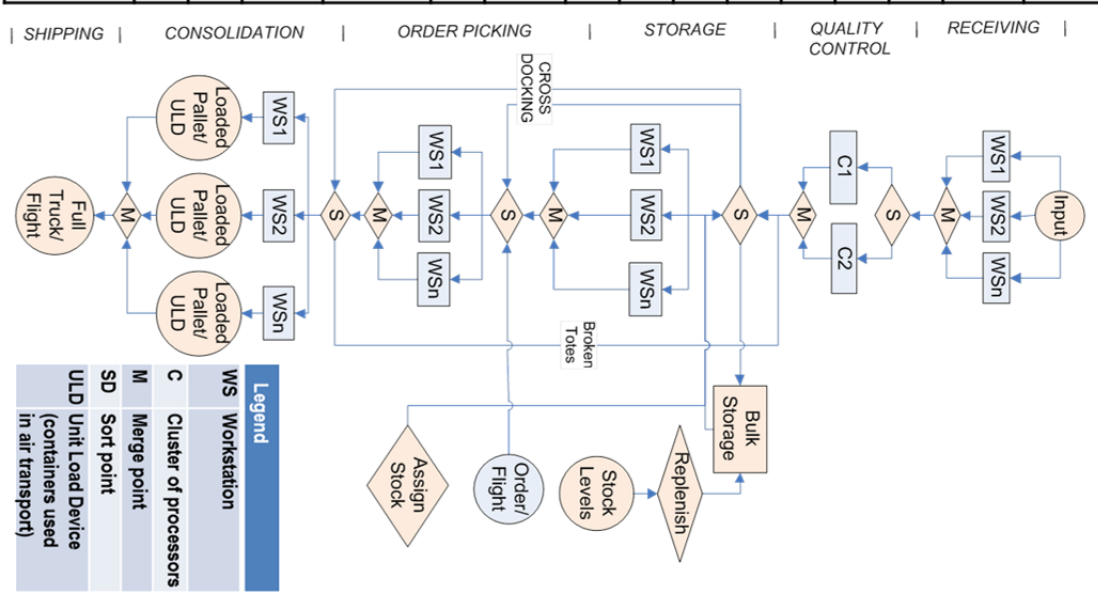


Figure 1.5. Generic material flow model with the description of process stages per sector.

- *Throughput:* this is a measure concerned with the capacity of systems. Throughput has to conform to the *functional capacity* requirements that specify the number of items the MHS is able to handle per unit of time while operating, according to design specifications. This presents a constraint to be met by the MHS. Moreover, throughput may be directly related to the overall operation time. For example, a transfer operation in an express parcel sorting system refers to the operation of unloading all arriving containers, sorting all parcels, and finally loading all sorted parcels. When this operation is performed in less time, the throughput is higher since throughput is measured in terms of parcels sorted per hour.
- *Response time:* this is a measure of the promptness in coping with dynamic operational requirements such as the completion of an urgent order in a distribution center, or the handling of a batch of urgent bags arriving at an airport.

The time dimension may suggest an overlap in the definition of these two main KPIs. However, a crucial difference is that throughput is measured at some point and as an average value, e.g., number of parcels passing the output chute per hour. On the other hand, response time covers the variation in the operational requirements by providing a time frame within which to respond, measured at a system level.

In addition to response time and throughput, we mention a KPI that has to do with operators working at the MHS. This KPI is *labor efficiency*, from the following perspective: wherever an interaction between the MHS and operators occurs, the MHS should function in a way that ensures efficient task allocation to operators even if inefficient allocation does not hamper throughput or response time. An example is when several operators load parcels onto parallel infeed conveyors in a sorting system (see Section 2.1). In this case, the speeds of the infeeds should be synchronized in a way that results in an even demand for parcels to be loaded by operators. In other words, having an infeed moving at a slow pace (e.g., due to a blocked output point), and another infeed moving at a fast pace, would require the operator on the fast infeed to load parcels at a higher rate than his peer on the slow infeed. This results in unfair workload distribution among operators. We summarize the aforementioned requirements in the following model:

Minimize Response time
Subject to
Throughput>= prescribed target (functional capacity)
Labor Efficiency>= prescribed target

The decision variables in the model above are basically the control rules implemented in the architecture. Examples of such rules are how to determine in which aisle to store a certain item, on which workstation to activate a certain order, when to release bags from storage to destination, and which route to take to the destination.

As a matter of fact, our collaboration with experts from industry resulted in a long list of functional requirements for MHSs. However, we claim that the model above presents a compact set of functional requirements, in which all other functional

requirements are implicitly involved. In the following, we present a list of the other functional requirements for the MHS, which are implicit in the model above:

- *Starvation avoidance*: starvation to material in an active resource/workstation is caused by delays in delivery from other resources or improper workload balancing. This phenomenon is implicitly handled as a means to reduce response time, or to aim at a higher throughput.
- *Blocking avoidance*: blocking occurs when an item is unable to get service from a workstation/resource, because it is still occupied or its buffer is full. Blocking is an obstacle to throughput, and may cause response times to be unnecessarily long. Therefore, blocking avoidance plays a role in the model.
- *Deadlock avoidance*: A deadlock is a condition in which items do not move on a certain transportation resource or are blocked at a certain workstation as a result of overloading the system resources.
- *Saturation management*: it is known in practice, especially in BHSs, that the capacity of the system decreases dramatically if the load on the system exceeds a certain threshold value. This state is called *saturation*. Undesired resource allocation may lead to saturation, which in turn leads to longer response times, and eventually may lead to a deadlock situation.
- *Prevention of imbalanced queues and recirculation* as they cause a decline in throughput.
- *Management of buffers*: in all systems there can be buffers. It is critical to deal with buffers properly; where, when, and how much to buffer in order to minimize response time and to satisfy throughput requirements.
- *Dealing with urgent items* (e.g., critical bags). This is directly related to optimizing response times.
- *Dealing with disruptions*: the control architecture should be able to respond to disruptions. E.g., it should divert bags in a BHS to a less occupied cluster of screening machines when another cluster suffers from an accumulation of workload. Moreover, the control architecture should respond to failures of physical equipment by proceeding the operation on the active equipment. E.g., when a crane fails in a distribution system then the retrieval tasks of the crane should be reassigned to the (active) cranes. These issues are related to the overall objective of response time minimization.
- *Operational flexibility*: this perspective of flexibility refers to the ability to cope with a changing operational environment. This requirement may be involved in response time minimization and throughput maximization simultaneously. For example, bags coming towards the Early Bags Storage have to be distributed evenly among parallel storage aisles. In this way, we gain higher throughput in the storage operation, and later in the retrieval operation as cranes can retrieve bags from all aisles simultaneously (assuming there is at least one crane at each aisle). Moreover, the time needed to retrieve all bags for a certain flight is minimized when bags of this flight are distributed among different aisles,

allowing several cranes to work on retrievals for the same flight. When the load in the system is high, incoming bags can be allocated to the first available aisle, i.e., the *water fall principle*. This strategy would result in even quantities across all aisles when the load is high enough to fill all aisles. However, when the load in the system is low, the water fall principle results in the first aisle to have a high load, whereas the load in aisles decreases as we go downstream. This happens when the load in the system is not high enough to fill all aisles evenly using the water fall principle. Therefore, we have to implement a smarter balancing strategy that reacts to changes in the operational environment (in this case low load in transport). In this context, operational flexibility is a functional requirement to be handled.

So far we discussed the functional requirements. At this point, we present the design requirements for a generic control architecture. Obviously, the main objective we seek is the design of a generic control architecture that may apply to MHSs in different industrial sectors. Moreover, we find that, in practice, other design requirements are necessary for a generic control architecture. In the following, we list these design requirements and make use of some descriptions presented by Zimran (1990) to define them formally:

- ***Flexibility***: the flexibility of a control architecture from the design perspective is the ability to introduce changes in the system layout with minor modifications in the control architecture.
- ***Modularity***: a modular design allows to build the architecture gradually through the use of a decomposed structure, and to have the architecture capable of introducing or removing some applications based on case-specific details.
- ***Scalability***: a scalable design allows the control architecture to control a wide range of system sizes.
- ***Robustness***: a robust design entails: first, *graceful degradation*, which is a term used often in practice and refers to the ability of the control architecture to keep functioning, and keep the MHS up and running when some units of the physical system fail. Second, it entails the ability to take action when disruptions occur.

Section 1.4 presents the results of a systematic literature review carried out to look for useful studies, which may help in synthesizing a control architecture that is in line with the requirements presented in this section.

1.4 Literature

In this section, we first present the basic forms of control in order to define the scope of the literature study and to position the studies in the literature review in a certain theoretical framework. In Section 1.4.1, we define this theoretical framework. Thereafter, in Section 1.4.2, we discuss the main literature contributions and position these studies using the reference framework of Section 1.4.1.

1.4.1 The basic forms of control

We focus on high levels of control that deal with decision-making functions and not on implementation issues, e.g., configuration of hardware elements, equipment instructions, and conveyor movements. Therefore, due to our functional rather than software implementation focus, we may exclude some basic principles of collaborative control theory. For example, Conflict and Error Diagnostics and Prognostics (CEDP) is a basic principle that is often studied in literature, e.g., in Chen and Nof (2007). However, CEDP focuses on software-related issues, i.e., the prediction and detection of errors in the software code, which is beyond our scope. As a matter of fact, we believe that on this level (e.g., machine interfaces, equipment control), standardization independent of specific applications is already the rule rather than the exception. Therefore, we explore whether a similar standardization may be achieved at higher, more abstract, decision-making levels. For examples of studies dealing with low levels of control and configurability, we refer to Alsafi and Vyatkin (2010) who present a methodology to integrate the high level planning with low level control of a mechatronic system, and to Furmans et al. (2010) who propose a plug-and-work MHS.

We use a theoretical framework that is based on the basic forms of control that have been suggested in the literature. We provide a description based on Dilts et al. (1991), who review the evolution of control architectures grouped in the major four forms of control, as follows (see Figure 1.6, where control units are represented by squares and resources by circles).

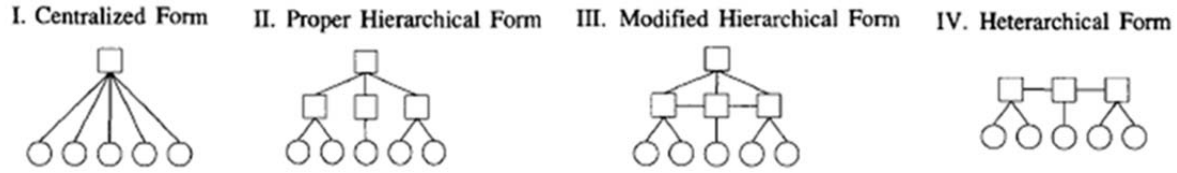


Figure 1.6. Evolution of control architectures (Dilts et al., 1991).

I. ***Centralized form***: Here a central control unit performs all planning and control functions for all resources in the system. Moreover, it uses a global database that contains all types of detailed information about the system. The main advantages of centralized control are: access to global information, possibility of global optimization, and a single source for system-status information. The disadvantages include: a single point of failure, where any problem with the central unit causes the whole system to stop functioning, slow and inconsistent speed of response, high dependency in the structure, i.e., single control unit, and complex software that is difficult to modify. The authors state that such control mechanisms are no longer common as they cannot deal with the requirements of today's complex systems.

II. ***Proper hierarchical form:*** In this form, there are multiple control units, and a rigid master-slave relation between decision-making levels. The control unit in an upper hierarchy acts as a supervisor for resources in the subordinate level. Decisions made by the supervisor have an aggregate view on the system, and do not prescribe detailed low level actions. Subordinate control units have to comply with tasks imposed by controls in the upper level, but as tasks are

delegated, subordinates make more detailed decisions for their actions. We notice that control decisions are executed top-down, while status reporting goes bottom-up. The main advantages of this form are: adequacy for gradual implementation of software, with less room for problems compared to the central control, fast response times, and last but not least delegation of lower level decisions to lower levels in the hierarchy so that not all details are at the highest level. The disadvantages include: making future modifications in the design is difficult, because the structure tends to be rigid and fixed in the early design stages (Dilts et al., 1991), an increased number of inter-level communication links (compared to the centralized form), and computational limitations of local controllers.

III. ***Modified hierarchical form:*** This form evolved in order to deal with some shortcomings in the proper hierarchical form, mainly the rigid master-slave relationship. It differs from the proper hierarchical form primarily through the degree of autonomy of subordinates. In the modified hierarchical form, there is some degree of coordination among subordinates on the same hierarchical level. This loosening of the master-slave relation brings additional advantages: more robustness to disturbances if the supervisor unit fails, because there is less need for continuous supervision, and subordinates have the ability to coordinate tasks among them. Some disadvantages are: connectivity problems among subordinates and with supervisors, capacity limitation of low-level controllers, and increased difficulty of the control system design.

IV. ***Heterarchical form:*** This form is the extreme of decentralized control, which became popular recently. An example is a multi-agent system (MAS). In this form, control structures have distributed locally autonomous entities. These entities communicate with each other to make decisions in cooperation. The master-slave relationship is totally abandoned and not just loosened as in the modified hierarchical form. In this control form, decision-making is distributed in some manner within the system. This distribution can be based on functions, geographical areas, task sequence, etc. Each control unit has its own rules and objectives, and communicates with other units to fulfill its own requirements. This notion is the general form of the agent-based systems. The main advantages of the heterarchical form are: full local autonomy, reduced software complexity, implicit fault-tolerance, high modularity, and faster diffusion of information as subordinates have smarter controllers. The disadvantages are primarily due to technical limits of controllers, lack of standards for communication protocols, and the likelihood of local optimization.

As a final remark, we emphasize again the distinction between our focus on *control architectures* (such as described by Dilts et al., 1991), and *software architectures*, because a decentralized control architecture can be implemented, in principle, by a monolithic software architecture and vice versa. However, advantages and disadvantages of centralization versus decentralization in both domains run in parallel to each other and are often mixed up.

1.4.2 Literature review

In this section, we list studies that are relevant to planning and control of MHSs in general, and to the industrial sectors in which we are interested. We make an attempt to classify the reviewed studies based on the framework for the basic forms of control (Section 1.4.1).

1.4.2.1 Centralized control

Tařau et al. (2009a) study route control in BHSs. They compare centralized and decentralized route choice in BHSs, particularly in systems using *Destination Coded Vehicles (DCVs)* as a transport mechanism. They implement centralized control approaches, but find them computationally expensive and not robust. Furthermore, they develop decentralized control rules for *Merge* and *Divert* switches, where each switch has its own controller. A merge switch is basically a piece of equipment that combines two inflows (of items) from two input sources, i.e., conveyor routes, into one outflow. On the other hand, a divert switch is a piece of equipment where items from a single inflow source can be routed to one of two possible outflow directions. We will study these elements further in Chapters 3 and 4.

Mo et al. (2009) study flow diversion to multiple paths in integrated automatic shipment handling systems. The authors take a network optimization perspective and formulate a nonlinear multi-commodity flow problem. They develop a mathematical programming model to propose routing strategies with the objective of minimizing the total shipment travel time in the system. However, they do not apply their theoretical framework to a business case and they make assumptions that may not hold in many practical settings. For example, they assume independent waiting times at different pieces of equipment and do not include time constraints for special shipments.

Zimran (1990) presents a commercial generic controller for material handling systems. His design is mostly based on hardware and software linkages and communication. The routing decision function is supported by tree graph algorithms. Tree graphs have only one path between every pair of origin and destination. These tree graphs change while the system is running (based on system state), by adding or removing arcs. Since the algorithm is computationally expensive, simpler algorithms are used for low level controllers.

1.4.2.2 Hierarchical control

The concept of *Cooperation Requirements Planning (CRP)* is a hierarchical decision-making strategy that stems from collaborative control theory. Rajan and Nof (1996) define CRP as "the process of generating a consistent and coordinated global execution plan for a set of tasks to be completed by a multi-machine system based on the task cooperation requirements and interactions". CRP is divided into two steps. The first step (CRP I) generates the *cooperation requirements matrix* whose elements represent the capabilities of machine sets for processing the tasks. CRP I also generates processing constraints. Next, the second step (CRP II) determines the assignment of tasks to machine sets for processing. These two steps may include advanced search algorithms to generate plans and to make assignments. In general,

CRP is unnecessarily complicated for our MHSs control problem. It is more adequate for a manufacturing environment such as the robots and machine cells application presented by Rajan and Nof (1996). In such environments, it is challenging to deal with jobs that need several processing tasks, which are not standardized. On the contrary, in the MHSs we study, items follow standardized routes and processes, but the challenge lies in the control and balance of material flows within the systems.

Amato et al. (2005) state that control systems of warehouses have three main hierarchical levels: a Planning level, a Management level, and a Handling level. The authors introduce the *Optimizer System* as a new level to bridge the gap between planning and management on the one hand, and shop floor control systems on the other hand, by improving the realization of decisions by handling devices such as the cranes and a shuttle handling device.

Faber et al. (2002) study the complexity in warehouses in relation to the warehouse planning and control structures. The authors focus on *warehouse management systems (WMSs)* and analyze the tradeoffs between tailor-made and standard WMSs. The authors present a holistic view on planning and control in warehouses. They describe a structure with different levels of planning and control. The main levels are the order management system, the WMS, and the technical control system. In this thesis, the focus is on planning and control activities within the technical control system, from the perspective of the MHS manufacturer. However, in order to understand the environment in which warehouses operate and to understand the dynamics that can influence the operation of the technical control system in warehouses, we refer to Faber et al. (2013). The authors investigate how warehouse management is organized and driven by task complexity and market dynamics, develop a multi-variable conceptual model based on the literature, and test it in 215 warehouses using a survey. Faber et al. (2013) suggest that task complexity and market dynamics are the main drivers of warehouse management. They assess how these drivers impact specificity of WMS using predefined measurement criteria. They also show how planning in production warehouses differs from distribution centers. We emphasize that the authors focus on the management of warehouses from a system user perspective.

In baggage handling, Tařau et al. (2009b) address hierarchical control for route choice. To this end, they design a control architecture with three levels of hierarchy: network controller, switch controller, and DCV controller. In the same study, they examine multi-agent systems, but find them hard to implement due to the extensive communication required between the agents. In general, Tařau et al. (2009a, 2009b) focus on BHSs and only on routing by controlling switches within BHSs, but they do not consider the storage operation.

1.4.2.3 Modified hierarchical control

Kim et al. (2003) propose a hybrid scheduling and control architecture for warehouse management, mainly for order picking. We classify their architecture as modified hierarchical, although they implement it using multi-agents software. In their architecture, they have three hierarchical levels of control: high level optimizer agent, medium level guide agent, and low level agents. The latter agents have a degree of autonomy that allows them to negotiate with each other and propose changes (to the

assigned tasks) to higher level agents. The authors claim that this architecture becomes a purely heterarchical architecture when the optimizer agent and the guide agents are eliminated, whereas it becomes purely hierarchical when communications among low level agents are prohibited. However, the fact that this architecture is tailored to order picking in a warehouse, limits it applicability as a generic control architecture for MHSs.

1.4.2.4 Heterarchical control

As a matter of fact, heterarchical forms of control are a recent trend in research. Babiceanu et al. (2004) present a framework for the control of MHSs as part of the so-called holonic manufacturing approach. Holons are units that act as parts and as wholes at the same time, meaning that they have a high degree of autonomy but operate as part of a more general system. Therefore, holons have two main properties: autonomy in making decisions and cooperation with other holons for mutually acceptable plans. The authors state that from the significant number of papers in the area of agent-based and holonic manufacturing, only a few consider material handling problems. They present a case study focusing on a material handling system.

Van Brussel et al. (1998) present a reference architecture for holonic manufacturing systems. Their architecture has 3 main holons:

- Product holon: represents a model of a product type, which basically acts as an information server to other agents.
- Resource holon: represents a production resource in the system.
- Order holon: represents a task with requirements and a due date. It manages a physical product being produced.

In addition, staff holons are optional holons that can aid other holons in decision-making. An example is a central scheduling unit. The architecture is called PROSA, which stands for Product-Resource-Order-Staff-Architecture. PROSA focuses primarily on manufacturing operations rather than transport operations. In this thesis, however, we do not aim for an architecture that is generic for MHSs and for manufacturing systems; we focus solely on MHSs and the operations within the industrial sectors we analyze. The complexity of decision-making in the MHSs that we study is less than that for a flexible manufacturing cell and, more importantly, is of a different nature. PROSA is an example of a completely heterarchical control approach, whereas we opt, for good reasons, for another form of control (see Chapter 2).

The holonic paradigm is similar to the agent paradigm in many aspects, but there are some differences. Giret and Botti (2004) conduct a thorough study to provide a comprehensive comparison of holons and agents. Their main conclusion is that a holon is a special case of an agent. A holonic system is a manufacturing-specific approach for distributed intelligent control. On the other hand, a multi-agent system is a broad software approach, where one of its uses is distributed intelligent control. For more details, we refer to Giret and Botti (2004). However, we note that holonic systems are heterarchical in the context of the systems that we address in this thesis, but they may have hierarchical characteristics when applied to other types of systems that are beyond the scope of this thesis.

Gue et al. (2013) study a high-density storage system, which has a modular physical structure. In this system, they present a conveyor-based material movement in a *puzzle* architecture that is analogous to popular board games such as the 15-puzzle and rush hour. They describe a decentralized control structure of this physical storage system in which each of the physical modules has an independent logic controller that is identical to the controllers of other modules. The study is based upon an earlier study (Gue and Kim, 2007) in which they present the puzzle-based storage system and analyze the tradeoffs between storage density and retrieval time based on a specific control algorithm. These studies focus mainly on a theoretical storage system in the distribution sector. However, Gue et al. (2013) emphasize the value of decentralized control for flexibility and scalability, and state that within material handling, decentralized control has been confined almost exclusively to the control of *Automated Guided Vehicles (AGVs)* or shuttles.

Vrba and Mařík (2006) focus on software implementation and the use of simulation in agent-based control systems. In their control architecture, they use a basic set of agents for conveyor-based transportation: work cell, divert, and conveyor belt. In this work, we find useful control mechanisms such as the dynamic routing tables used by the diverters. We stress that the main objective of our research is to propose a generic control architecture for MHSs that is applicable in different industrial sectors, where not every element within this architecture is necessarily a novel application.

Lau and Woo (2008) develop an agent-based dynamic routing strategy for MHSs. They emphasize that existing routing strategies in theory often use static routing information based on shortest path, least utilization, etc. In their study, they map the MHS to a network with node agents connected by unidirectional links. Control points of a network of MHS components are modeled as cooperating node agents. To make routing decisions, they define the best route in terms of: cycle time of material, workload balancing, and degree of tolerance to unexpected events. In their architecture, each agent is responsible for its zone of coverage. They implement their architecture in a simulation environment of a DC. The authors outline a generic classification of routing strategies and classify their approach as *distributed real-time state-dependent*.

Johnstone et al. (2010) study status-based routing in baggage handling. In their approach, the status of the bag determines its processing requirements and triggers computation of the route to be followed depending on the states of required resources ahead. The authors study two main algorithms: the first one based on learning agents, while the second uses a graph representation of the network to find all possible routes at switches via Dijkstra's shortest path algorithm (Dijkstra, 1959). They find learning agents more efficient in larger systems, as they make use of information from operations performed on the bags upstream. With this information, they limit the possible routing options downstream.

Hallenborg and Demazeau (2006) use multi-agent technology in a BHS to construct generic software components to replace traditional system-specific centralized control software. In their approach, when the bag enters the system, the first agent on the route can make an agreement with all agents on the route to the bag's destination. However,

it is also possible to make an agreement only with the next agent on the route. This raises the distinction between routing by static shortest path and routing on the way. We also refer to Hallenborg (2007a) for a case study of a large airport hub in Asia, in which a centralized control architecture is replaced by an agent-based solution.

Some of the advanced control designs generate forecasts in order to prevent congestions and to facilitate proactive rather than reactive decisions. Studies in this context include Hadeli et al. (2004) who present a control architecture that is a combination of PROSA and concepts inspired by ant colony coordination mechanisms. Weyns et al. (2007) use delegate MASs, inspired by food foraging in ant colonies, to anticipate road conditions to make routing decisions. Claes et al. (2011) present an MAS for anticipatory vehicle routing, which allows directing vehicle routes by accounting for traffic forecast information. Finally, Parunak (2010) presents the concept of swarming agents that interact through digital pheromones. However, note that we focus on internal transport, as distinguished from external transport that is dealt with in these studies. Chapter 2 further describes other control approaches and anticipation techniques, which we employ to take precautions in order not to create congestions and in order to maintain a balanced material flow in the system.

In this thesis we do not study autonomous vehciles. However, we refer to Kamagaew et al. (2011) and Wurman et al. (2008) for control approaches for autonomous vehicles. Moreover, we mention Mayer (2009) who develops a decentralized control system for modular continuous conveyors. The latter study, however, focuses on the equipment level (i.e., the mechatronics of the system) whereas we take higher functional control levels as our main focus.

1.4.2.5 Other studies

Some simulation-based studies in the area of MHSs are worth mentioning. Meinert et al. (1999) present a modular simulation approach for the evaluation of MHSs. Babiceanu and Chen (2005) use simulation to justify the use of a decentralized agent-based approach in materials handling and assess its performance compared to conventional scheduling systems. Jahangirian et al. (2010) conduct a broad review of simulation studies in manufacturing. A trend they notice concerns the increasing interest in hybrid modeling as an approach to cope with complex enterprise-wide systems. Hunter (1994) presents a model evolution analysis for simulating MHSs. Finally, we mention Van den Berg (1999), Rouwenhorst et al. (2000), and Gu et al. (2010) as useful literature reviews in the distribution and warehousing area.

In parcel & postal sorting, we could hardly find any studies discussing control architectures. McWilliams et al. (2005) introduce the *Parcel Hub Scheduling Problem (PHSP)*; this problem concerns the scheduling of a set of inbound trailers to a fixed number of unload docks at an express parcel sorting hub. The objective is to minimize the makespan (i.e., total required time) of the transfer operation, i.e., sorting all unloaded parcels to the required destinations. In his studies, McWilliams deals with the MHS as a black box and does not interfere with the inner control. His studies include simulation-based genetic algorithms and dynamic load balancing heuristics. From his work on the PHSP, we mention the development of a dynamic load-balancing scheme for the PHSP (McWilliams, 2009b). A useful result of his studies is

that a balanced flow within the system results in minimizing the time required to accomplish the transfer operation.

1.4.3 Concluding remarks

As a general remark, there are few studies that attempt to build a generic control architecture for MHSs operating in different industrial sectors. From the studies we reviewed, we observe that a control architecture normally targets a specific sector or deals with material handling as part of a manufacturing environment. From our point of view, the most relevant study is the holonic architecture proposed by Babiceanu et al. (2004). Although this architecture is based on a manufacturing system, it does suggest a framework for material handling. However, the MHSs in the sectors we address are far more complex and diverse than the MHS modeled by Babiceanu et al. (2004). We conclude that their study misses an in-depth treatment of practical requirements of complex MHSs as they do not show how decision-making processes can be employed to achieve functional requirements. However, we may make use of their findings in the architectural design aspects. In general, many authors favor distributed control when dealing with complex systems.

From the studies we reviewed, we observe that a control architecture is initially designed and then applied to some sector, often to a distribution center. For baggage handling, there are few studies on control architectures. Most of the studies focus on route planning through divert and merge switches and do not take the storage operation into account. On the other hand, the relatively abundant number of studies on warehousing systems emphasize either the design aspects or throughput optimization of the system through the use of advanced algorithms for warehousing activities such as: storage and retrieval sequencing and order pick concepts. From our experience with industry we however learned that other requirements are necessary to make the control architecture applicable in a practical setting. For example, experts from industry value a robust control architecture that provides satisfactory solutions higher than an architecture that provides near optimal solutions but is less robust. Finally, we could hardly find studies for parcel & postal sorting that discuss a control architecture, probably because MHSs in this sector are of less complexity, i.e., they are basically sorters. In this sector, related studies deal with inbound and outbound operations. Most relevant in this context is the parcel hub scheduling problem introduced by McWilliams et al. (2005), which we address in Chapter 5.

1.5 Theory versus practice

This section confronts the theoretical studies with practical requirements (Section 1.5.1), and based on this confrontation defines the agenda of our research (Section 1.5.2).

1.5.1 Confronting literature studies with practical requirements

As mentioned briefly in Section 1.4.2, there is a lack of in-depth studies dedicated to the generic control of complex MHSs. There are studies addressing MHSs from different perspectives. A few studies claim that they propose a generic control

architecture or framework. However, we find them lacking due to one or more of the following reasons:

- *Being applicable only to a specific sector:* when an architecture is based on one sector, it becomes impractical for other sectors as it normally misses relevant problems, constraints, and objectives in a different operational environment.
- *Lacking an in-depth treatment of practical requirements:* the functional requirements listed in Section 1.3.2, present necessary conditions for a comprehensive control architecture. Moreover, the architecture has to control all possible subsystems of a complex MHS, e.g., ASRSs and divert switches. We conclude that a comprehensive coverage of these requirements is still lacking because the current studies are limited in several ways. First, they model simple material handling systems where no complex decision-making is required. Second, they focus on certain problems and subsystems, e.g., they deal with urgent items and with routing at diverts and do not address other problems, such as management of buffers and ASRS control, in the same architecture.
- *Limiting the role of MHSs to be merely a support to a manufacturing environment:* there is limited focus on complex MHSs that are functioning for the sake of material handling and not merely as part of a manufacturing environment. The latter trend generally results in simplified MHS problems.
- *Missing the combination of design requirements and functional requirements in a unified architecture:* there is a need for a comprehensive control architecture that is designed according to the design requirements, but that also entails control rules and algorithms implemented to satisfy the functional requirements. Studies on control architectures normally address design requirements (modularity, robustness, scalability, and flexibility). Yet, we could hardly find any study with proven implementation potential on MHSs in different industrial sectors.

At a lower level of analysis, we find studies addressing specific problems or sub-systems within MHSs. Moreover, we find sector-specific studies (e.g., control of BHSs). Therefore, results of specific problems can be used as building blocks in a new generic control architecture. However, having subsystems functioning properly on their own does not mean that the combination of subsystems functions properly as well. Therefore, a top-down design approach makes sense, because it allows to deal with the system dynamics at an early stage. Finally, there may be a need to adapt solutions for subsystems in certain sectors to be generic for similar subsystems in all sectors.

1.5.2 Research agenda

In this thesis, we aim at developing a comprehensive generic control architecture that satisfies design requirements and controls the operation of the MHSs in a way that satisfies the functional requirements. Both sets of requirements are defined based upon the research we performed at a major global company supplying material handling systems in all sectors discussed in the thesis. Based on our study, we conclude that there are still contributions needed for literature to answer questions in practice. The

missing points in current studies provide starting points to propose an agenda for this research. In addition, this research differs from other studies in addressing three different sectors from practice and using their requirements simultaneously to develop a generic control architecture. Current studies either develop control approaches and then apply them to a certain sector or use cases from one specific sector as a starting point. This thesis aims to handle the following elements:

1. *Proposing a concept for a control architecture:* the concept may be based on the basic forms of control (see Section 1.4.1). We may decide upon the most appropriate form or propose a hybrid of several basic forms.
2. *Detailing the concept in terms of control levels (hierarchies) and control units*: in particular, we have to address the relations between these different decision-making bodies and the spans of control for each. This point has to satisfy the design requirements (see Section 1.3.2).
3. *Translating the concept into a concrete control architecture:* this requires proposals for control rules and algorithms at the control levels and units. We have to define the links between control levels or control units in terms of information transmitted and the way information is reacted upon and communicated. This point has to satisfy the functional requirements (see Section 1.3.2).
4. *Validating the generic control architecture:* this requires the modeling and testing of operational scenarios of MHSs in different industrial sectors.
5. *Proving the adequacy of the control architecture:* this requires implementation on a business case to prove its adequacy to serve as a generic control architecture.

Section 1.6 outlines the remainder of this thesis, where it refers to the aforementioned points on the research agenda. Moreover, Section 1.6 illustrates how each of these points fit in the content of the subsequent chapters.

1.6 Summary and thesis outline

This chapter introduced the research problem on generic planning and control of MHSs that occur in different industrial sectors. Section 1.1 motivated this research, while Section 1.2 defined its scope and described the industrial sectors involved. Section 1.3 analyzed the synergy among the different sectors. Furthermore, the process flows in the different sectors were modeled in an analogous way given a certain level of abstraction. This analysis, partly based on close experience with the material handling industry, led to a list of general requirements for a generic control architecture. These requirements are valuable for all industrial sectors and concern both the design and functionality of the control architecture. Subsequently, Section 1.4 reviewed the literature to investigate the availability of answers to the requirements from practice. Consequently, Section 1.5 weighed the requirements from practice against the existing literature and highlighted the missing links to propose an agenda for this research in the field of planning and control of MHSs. This section presents the organization of the remainder of this thesis.

Chapter 2 deals with the first two points on the research agenda, i.e., proposing a concept control architecture and detailing it (see Section 1.5.2). It builds upon our conclusions so far to propose a concept control architecture. We propose a variant of the modified hierarchical form of control (see Section 1.3.2) and motivate our design choices. In this concept control architecture, we propose a set of generic control units distributed over three levels of control, i.e., planning, scheduling, and local traffic control. We argue that the planning and scheduling levels of control are the main determinants of the control structure and these are the levels covering communication links and cooperative decision-making processes. Local traffic problems have a minor global effect on the system performance. These problems occur at certain physical areas of the system at a low level of control, where no communications with other system areas are needed. Therefore, at this level we can derive separate traffic problems. For one of these problems, which is the most challenging, we develop a solution in Chapter 6, but for the rest, available solutions in literature can be implemented. Chapter 2 presents all decision-making processes encountered in the control architecture and describes each of them.

Chapter 3 deals with the third and fourth points on the agenda, i.e., developing the *concept* into a *concrete* control architecture and validating it. The control architecture is divided into modules. We mainly implement planning and scheduling control modules and include local traffic control modules in an aggregate manner. Chapter 3 presents a generic MHS model, in a simulation environment, which can be tuned to simulate MHSs in different industrial sectors. We model different operational scenarios and analyze the *generality* of control. Chapter 3 also presents an implementation of the routing module (at the scheduling level), using the aforementioned generic MHS model with a modified system base.

Chapter 4 deals with the fifth point on the research agenda, i.e., proving the adequacy of the control architecture. It presents a comprehensive application of the control architecture on a business case, in which we study a major European airport that entails challenging system elements and business rules to be handled by the generic control architecture. In this large implementation, we face new system areas that need to be handled in a generic manner and we show how we face standardization challenges not only among different industrial sectors, but also within the same sector and the same MHS. Moreover, we deal with a routing module, for which we attempt to maintain the generic control structure as proposed for such problems in Chapter 2 and as implemented in another system and another industrial sector in Chapter 3. Finally, given this comprehensive implementation, we compare the performance of the generic control architecture to current practice. This chapter provides a proof-of-concept for the applicability of generic control on practical cases.

As the control architecture is designed, implemented, and confronted with more challenges in a business case, we take a step to extend our analysis in Chapter 5 from Scope 1 (i.e., MHS control) to Scope 2 (i.e., inbound operations; see Section 1.2.1). In this setting, we present a scheduling problem for system-users, which is scheduling inbound containers to load MHSs that use sorters as the main element. We find this scheduling problem influential to the operation of sorter systems and thus we dedicate a chapter to it, where we build upon the state-of-the-art algorithm available in

literature and introduce two extensions. In this sense, Chapter 5 provides scheduling tools for the MHSs' users.

Although most of the local traffic decision-making problems are simple and straightforward (e.g., determining a crane's travel trajectory between two pickup and retrieval locations), we identify one local traffic problem that is challenging and requires a sound decision-making algorithm. This is the space allocation problem in conveyor merge configurations. In such configurations, a set of parallel conveyors transport items towards one larger *merge* conveyor. The merge of all incoming items from the parallel conveyors onto the merge conveyor is an operation where several challenges have to be dealt with. For this thesis to provide a comprehensive set of solutions for decision-making problems in MHSs, we dedicate Chapter 6 to analyze this local traffic control problem and to propose an algorithm for it.

In the final chapter of this thesis, i.e., Chapter 7, we present general conclusions and recommendations for practice. Moreover, we highlight directions for future research.

Chapter 2

A Generic Control Architecture[7]

In Chapter 1, we discussed the generic control problem of MHSs as they arise in different industrial sectors and concluded that there is a need for a generic control architecture for such MHSs. Following this conclusion, we have proposed a research agenda of five points. This chapter deals with the first two points, i.e., proposing a concept control architecture (in terms of control levels and generic control units) and detailing it (in terms of concrete decision-making processes).

In this chapter, we propose a variant of the modified hierarchical form of control (see Section 1.3.2) and motivate our design choices. In this concept control architecture, we propose a set of generic control units distributed over three levels of control, i.e., planning, scheduling, and local traffic control. We discuss the elements that represent a crucial part of the control structure in detail, whereas elements that are not crucial to the control structure are treated in general terms and discussed in more detail in subsequent chapters. Section 2.1 builds on the findings of Chapter 1 to develop a concept control architecture that includes a set of generic control units at three levels of control (i.e., planning, scheduling and local traffic control). Thereafter, Section 2.2 details the concept architecture by illustrating the functionality of the control units and the decision-making processes that take place within the control units and among different control units (at various levels of control). Finally, Section 2.3 ends this chapter with concluding remarks.

2.1 A concept for a generic control architecture

In this section, we take a first step towards the development of a generic control architecture by proposing a concept control architecture and detailing it in terms of control units and hierarchies. To propose a concept for generic control, we build on the experience from our industrial cooperation and on the basic forms of control discussed earlier (Section 1.4).

First of all, we comment on the centralized form of control, in which a central control unit performs all planning and control functions for all resources in the system. We find this form of control inappropriate for the generic control of MHS for reasons concerning both the operational environment and the design requirements. Therefore, we exclude this option in view of the following arguments:

[7] *This chapter is based on Haneyah et al. (2013a) and Haneyah et al. (2013b).*

- The centralized form of control is rigid when it comes to: handling the real-time flow of information, dealing with disruptions in material flows, and controlling processes in a dynamic environment.
- The required computation time for a central control unit to process a large amount of data and to make decisions is incompatible with the real-time nature of the MHSs.
- Information on items transported by the MHSs flows in real-time and is revealed gradually with a narrow look-ahead horizon. Therefore, making global decisions that affect every resource in the system based on a narrow scope of information does not make sense, especially because it may easily happen that the centrally proposed decisions change radically when new information becomes available, e.g., about disruptions or new items in transport.
- The software may become complex to build and may not serve the design requirements of being generic, modular, robust, and flexible (see Section 1.3.2).

The centralized approach is one extreme of decision-making; the other extreme is purely decentralized decision-making embodied by the heterarchical approach. According to various authors, the main advantage of the heterarchical approach is that it supports desirable design aspects, i.e., modularity, a generic structure, and robustness. Modularity is embodied in the possibility to build software components separately and to include some intelligence in decision-making activities. The control architecture can be composed by configuring the interfaces between software components.

In our view, a pure heterarchical form of control results in a cooperative approach to global decision-making, where a main concern is the extent of deviation from optimality. In this context, higher level coordination may be necessary for some processes, e.g., planning orders. Moreover, for the generic control problem, decisions made within MHSs are not all at the same level. In particular, when looking at the different industrial sectors that we analyze, we find global decisions that impact the overall performance of the system, while others are local decisions with limited global impact. As a matter of fact, distributed control is beneficial when dealing with complex systems. However, we emphasize that distributed control means having *decisions made at the right level*, and thus it can be realized with other forms of control, e.g., the modified hierarchical form.

Given the aforementioned points and our observations in industry, we propose a control architecture that involves hierarchical control and also a certain degree of intelligence and freedom of controllers at different control levels.

The control architecture is the basic structure on which decision-making processes are mapped. The proposed control architecture builds upon the theoretical framework (Section 1.4.1), while the decision-making framework builds upon established theories in the temporal decomposition of planning, scheduling, and control processes (see Anthony (1965), Hax and Meal (1975), and Zijm (2000)).

From our analysis of the MHSs in the three sectors (distribution, baggage handling, and parcel & postal sorting), and the decision-making aspects in particular, we find it necessary to first have a control level that takes care of the planning activities using an

aggregate view of the MHS. Moreover, this control level should provide the interface with the system-user, e.g., the receipt of flight schedules in a BHS or of order details in a distribution system. Second, the resources of the system have to be controlled but not centrally, as argued earlier. Therefore, resource controllers are needed that schedule and execute work considering their own status and the status of other resources involved in the handling operation. Finally, when all decisions on workload control and material flow are taken, the realization of these decisions by the physical equipment has to be taken care of at a dedicated level, e.g., to store a TSU[8] within a certain storage aisle or to induct a TSU on a conveyor belt at a merge junction. Note that these levels are all within the control software of the MHS, i.e., the equipment, and deal with operational tasks using inputs from higher level user systems such as the WMS in distribution. Following this discussion, we propose three hierarchical levels of control, where each level contains several generic controllers as we will describe later. The three levels of control are as follows (see also Figure 2.1):

- **Planning:** The planning function requires a global view of the system regardless of the system size. This is the control level that interacts with the MHS's user environment, e.g., customer orders or plane schedules. As a result, this level is mainly responsible for the assignment of work to resources/system devices. Planning decisions are made by abstract controllers using aggregate system information.
- **Resource scheduling:** Given a set of assigned tasks, the scheduling function addresses the problem of when and in what sequence to execute these tasks. This level deals with executing the tasks assigned by the planning level. Scheduling decisions are made by resources controllers. In this context, system resources that need their own scheduling controller are either workstations (e.g., pick stations in a warehouse) or key transport and routing resources (e.g., sorting loops and cranes). Routing and task sequencing for each resource are decided upon here. In this sense, this level depends on the system layout and specific system attributes. For example, travel distances within the system and loads in transport may affect scheduling decisions.
- **Local Traffic Control:** This function entails algorithms or routing rules executed within defined boundaries of the physical system. There is minimal interaction with other areas in the system and mostly the aim is local optimization where no global view is needed. Decisions made at this level do not have a major effect on the system. These decisions are implemented at a low level of control or made by resource controllers. Examples include the movement of a crane within its aisle and prioritizing the movement of items on a conveyor junction.

From a theoretical point of view and also based on experiences from practice, local traffic control problems are the easiest to deal with, as they do not affect the overall

---

[8] *Transport Stock Unit (TSU): a generic term to refer to different types of items transported on the automated MHSs, i.e., bag, parcel, or tote.*

control structure or the communications among different controllers. Control methods for local traffic problems can be integrated in a control architecture with minimal difficulty. The higher levels of control, i.e., planning and scheduling, are the challenging levels of which the functionality is highly dependent on the control structure and communication interfaces.

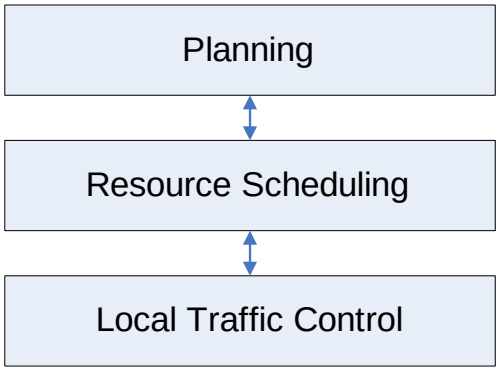


Figure 2.1. Levels of control.

In this concept architecture, planning control units, referred to as *planners*, have an aggregate view of the system and are not directly connected to system resources. On the other hand, scheduling control units, referred to as *schedulers*, are directly connected to system resources, being workstations or transport resources. Planners communicate with each other and assign tasks to subordinate schedulers. Schedulers also communicate with each other to schedule the assigned tasks and report to higher level planners. They are responsible for task sequencing and execution. Schedulers communicate via standardized interfaces to execute the transportation process and fulfill the tasks assigned, e.g., to deliver a bag to its destined lateral. In this thesis, planners and schedulers are pieces of software.

The proposed control architecture (see Figure 2.2) has a certain degree of hierarchy combined with flexible decision-making for subordinates, as in the modified hierarchical form of control. However, we may define several higher level control units (planners) rather than a single higher level control unit. Therefore, the control architecture is a variant of the modified hierarchical form of control (see Section 1.4.1). At this point, we emphasize once more that we take the control perspective of the architecture and not the software implementation perspective (see Section 1.4).

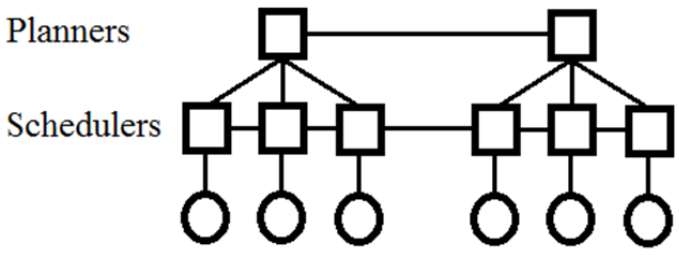


Figure 2.2. Control architecture scheme.

We have stated that schedulers are directly connected to system resources, which implies that each system resource (e.g., a crane) has its own scheduler. Therefore, system resources define the set of schedulers included in the architecture. On the other hand, we define planners as higher level control units that have an aggregate view of

the system and are not directly connected to system resources. There are two main planners that we incorporate in the control architecture:

- *Build planner*: responsible for the *build area*, i.e., workstations. In distribution, this means planning the order picking process, whereas in baggage handling this means planning the *build of flights*, i.e., gathering the baggage belonging to the flight at the right build point(s). This is a planner as it requires a global view on system information, schedules, and the build area. Moreover, it results in assigning work to system resources (see our definition of the planning level in the concept architecture).
- *Storage planner*: this controller is responsible for the storage area, i.e., the ASRS consisting of cranes and storage aisles. The same arguments as for the build planner hold for this controller to be a planner. Here, the global view required is on the ASRS.

Now that we defined the basic control structure, we next have to allocate decision functions to the different levels of control and to the different controllers. In doing so, we follow two main principles:

- Allocate each decision function to the lowest *possible* control level and with the narrowest possible scope. Here, the word *possible* means that no direct deterioration in system performance is expected due to making the decision local and with a narrow scope. However, this principle may be violated due to a required synergy in control among different sectors.
- If an operation with certain characteristics is defined as a scheduling operation, another operation with the same characteristics but on a wider scale due to, e.g., system size, may become a more complex scheduling operation, but does not become a planning operation.

In order to apply generic control methods, we have to treat systems that are at the same level of detail similarly. However, due to the varying nature of MHSs in different industrial sectors, the same level of detail is not always present. Therefore, it is essential for the control architecture to have elements that deal with the differences among systems, to produce a certain level of detail that is then usable by generic control methods. We describe this further in Section 2.2.

2.2 Decision-making processes

In this section, we introduce the main decision-making processes, relevant to the proposed architecture, at each level of control. In this context, we stress that parcel & postal sorting systems are controlled at the scheduling and local traffic levels as described later (Chapters 5 and 6). In parcel & postal sorting, there are no ASRSs (which need the planning level), while the assignment of destinations to chutes is an input parameter to the system that is usually fixed for a longer time. Therefore, the planning processes described in this section regard distribution and baggage handling.

2.2.1 Planning processes

At the planning level, we introduce two main processes: planning the outbound flow from the ASRS and planning the inbound flow to the ASRS. In this section, we

describe these processes and explain their functionality in general terms. Chapter 3 deals with detailing and implementing these processes on a specific system model.

2.2.1.1 Inbound flow to the ASRS

When a TSU requires storage, it is announced to the storage planner, which responds with a destination aisle and crane to perform the storage operation. The decision can be made according to different control rules, which may be based on different criteria. For example, an incoming TSU containing a certain *SKU (Stock Keeping Unit)* can be assigned to the aisle having the minimum level of this SKU. As a matter of fact, the outbound flow is the main contributor to system throughput and assigning the inbound flow of TSUs to aisles may not have a major impact on the outbound flow. Therefore, it may be possible to use simple control rules. The advantage is then simpler software, but we have to test whether such simple rules do not cause any deterioration in the system performance. Chapter 3 presents and analyzes alternative control rules further. Note that TSUs returning from a workstation in a distribution system (no item integrity, see Section 1.3.1) are part of the inbound flow to the ASRS.

For the user of the MHS, smarter control rules or allocation algorithms might be incorporated. There are plenty of studies on detailed allocation decisions, e.g., where to store an incoming tote in terms of the aisle and storage location within the aisle. However, in this thesis we focus on the generic control structure that can accommodate other control algorithms depending on the characteristics of the distribution center or airport where the MHSs is used. Moreover, what happens before a TSU is loaded on the MHS is not within our scope as it is within the responsibility of the system user.

2.2.1.2 Outbound flow from the ASRS

This process (see Figure 2.3) is a planning process as it requires a global view of the ASRS and of the destination workstation(s). Moreover, it results in assigning tasks to resources, e.g., retrieval tasks to cranes. There are two main sub-processes in outbound flow planning:

a. *Stock reservation*: in distribution, a customer order has a set of order lines, each referring to an SKU required in a certain quantity. An order is built on one workstation, but to build the order, stock is retrieved from the ASRS. Since multiple TSUs may hold the same SKU, it is necessary to decide which TSU to reserve for usage of a certain order, i.e., stock reservation. However, in baggage handling, we define an order as a set of bags required for a certain flight. In this sense, bags are uniquely identified, as each bag entering the system via check-in desks or as transfer baggage is already assigned to a specific order (flight). Therefore, we see stock reservation as a process that results in bringing the distribution system to the same level of detail as a baggage handling system, by assigning TSUs to orders. This process is accomplished as the build planner requests stock reservation for certain orders (plans orders) from the storage planner, which in turn looks for TSUs to reserve. Typically, broken TSUs are attempted before breaking a full TSU, as having many broken TSUs in the ASRS means a loss of storage capacity. The build planner makes sure that a

couple of orders are planned and ready for activation on any workstation requesting work.

b. *Order release*: workstations trigger the build planner to activate orders, based on work progress in distribution and according to time schedules in baggage handling. As soon as an order is active on a workstation, stock belonging to this order has to be released from the ASRS. Therefore, the build planner informs the storage planner that a certain order is active. In turn, the storage planner dynamically assigns the reserved TSUs as retrieval tasks to candidate cranes (when a reserved TSU is accessible by more than one active crane). The storage planner may use different control rules to assign a retrieval to a crane (e.g., assign the retrieval to the candidate crane having less workload). From this point on, cranes are responsible for executing and sequencing these tasks at the scheduling level of control. Note that the preceding stock reservation process (a) brings a distribution system (from a control point of view) to the same level of detail as in a baggage handling system.

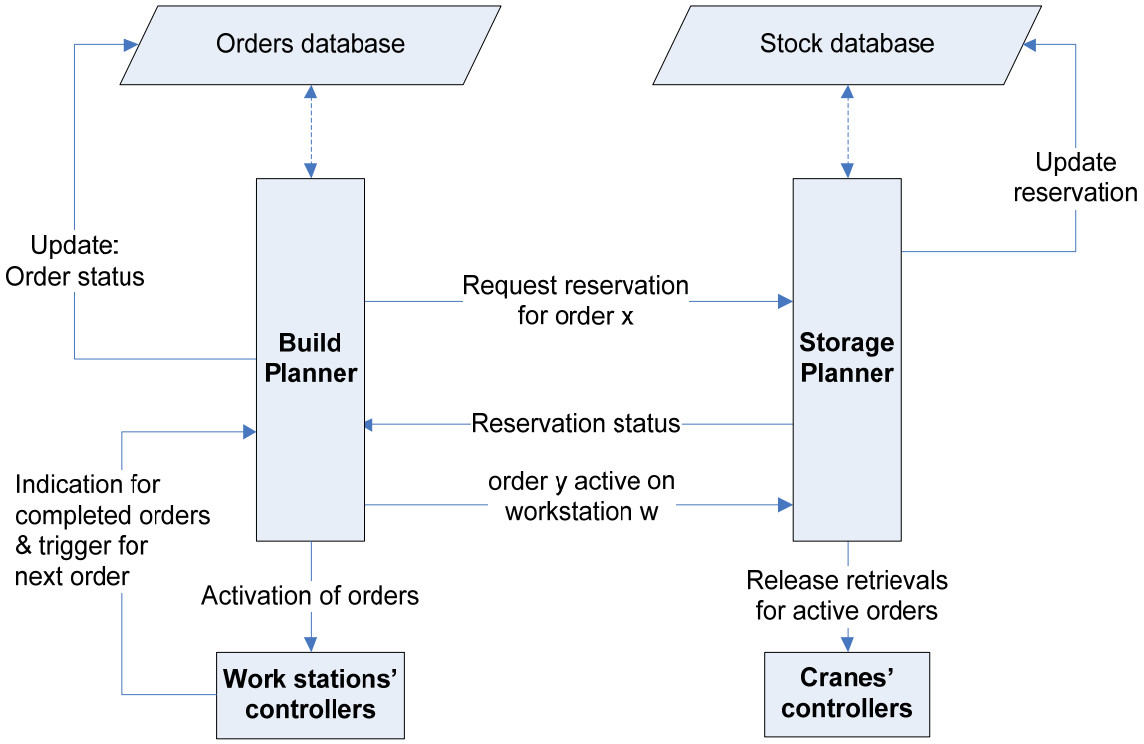


Figure 2.3. Communications at the planning level.

2.2.2 Scheduling processes

At the scheduling level, we introduce the following processes: scheduling crane retrievals, scheduling inbound containers, routing arriving TSUs, and routing TSUs in networks. In this section, we focus on the scheduling processes that represent an integral part of the control architecture. This applies to all of the processes mentioned except for scheduling inbound containers. Although this process influences the operation of MHSs, it is not an integral part of the control architecture that is at Scope 1 of our analysis (see Section 1.2.1). In fact, scheduling inbound containers is at Scope 2 (inbound and outbound operations). Therefore, we do not describe the decision-

making aspect of this process in this section. Likewise, with regard to the implementation of these processes, Chapter 3 deals with detailing and implementing the scheduling processes except for scheduling inbound containers, to which we come back in Chapter 5.

2.2.2.1 Scheduling crane retrievals

Crane controllers have to schedule the released set of TSU retrieval tasks. In order to support functional requirements such as avoiding blockings and deadlocks, the *pipeline* occupation of the TSU destination plays a role in scheduling. TSUs are normally retrieved to be loaded on the main sorting system in baggage handling and distribution MHSs. However, in parcel & postal sorting, the sorter is the main element, where no cranes are involved.

Given a set of retrieval tasks, crane controllers schedule these tasks based on their priorities and the pipelines of destination workstations (see Figure 2.4). However, what defines a priority TSU differs per industrial sector; we explain this further in Chapter 3. The pipeline size of a workstation is the maximum number of TSUs that are allowed to be in transport to this destination at any point in time. Therefore, we can send more TSUs only if the number of TSUs already in the pipeline is less than the pipeline size, in order to prevent overloads and congestion.

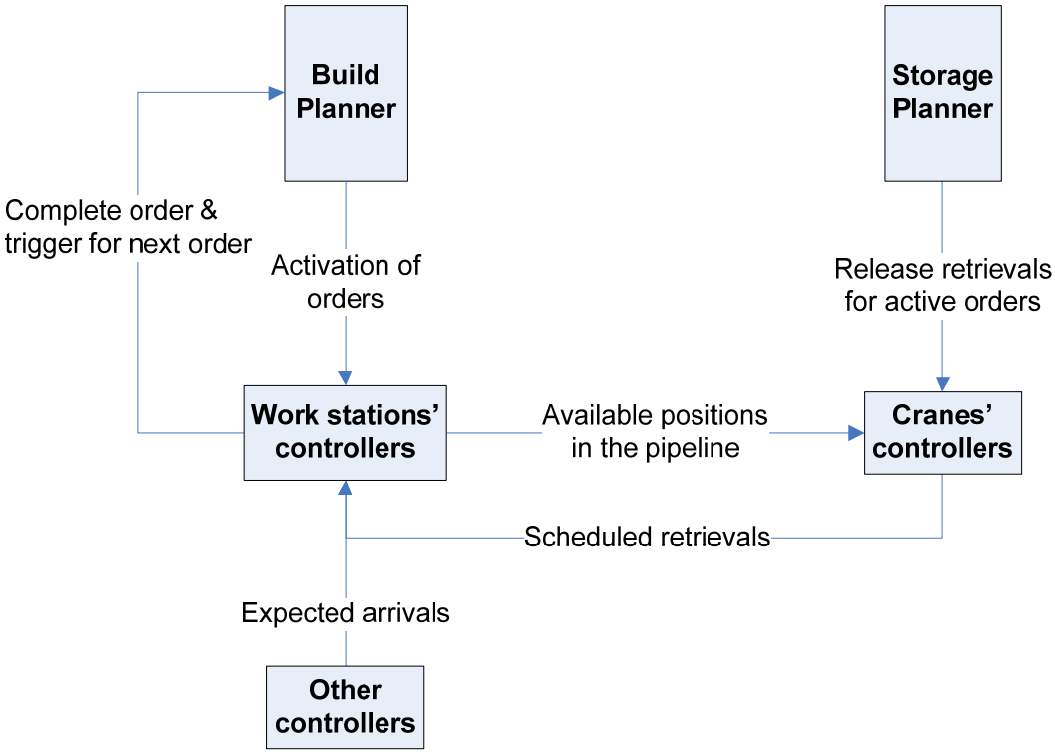


Figure 2.4. Communications at the scheduling level.

We use the pipeline size concept often in the following manner: each workstation receives information about incoming TSUs (from, e.g., cranes or the receiving divert) and in turn updates the information about the number of empty positions remaining in its pipeline. Scheduled retrievals that are not physically in the pipeline yet, are also taken into account as being in the pipeline. Other controllers, e.g., crane controllers,

observe the pipeline capacities and take this information into consideration when scheduling crane operations. In the control architecture, the size of the pipeline is an important parameter to define. In general terms, there are two different approaches to set the pipeline sizes of workstations. These approaches depend on the system layout and the operational environment as follows:

1. If the transport equipment (e.g., sorter system) has limited capacity (in relation to the workload) or if a group of TSUs is planned to arrive at their destined workstation in a predetermined sequence, then the material flow has to be strictly controlled. Therefore, the pipeline size is typically equal to the number of locations in the inbound buffer of the workstation. In this way, if any problem occurs at the workstation or the operator is temporarily absent for some reason, then all TSUs in transport can be accommodated in the inbound buffer (and in a preserved sequence if there is any). No TSU should waste the capacity of a loop sorter by circulating on it due to blocked entry to the workstation. Likewise, circulation may damage the predetermined sequence of TSUs that are moving towards their destination. We provide applications and more details on these points in Chapter 3.
2. If there are long transport routes from the ASRS to the workstations area or if there is a large loop sorter that shows fluctuating occupation levels, e.g., according to baggage arrivals and flight schedules, then it is important to maintain a continuous flow towards workstations. Otherwise, there will be instances of no flow, causing starvation at subsequent processing units (e.g., workstations). Another point is that travel times to workstations may differ due to the larger system. In this sense, the farther workstation needs more TSUs in transport than the nearer one to maintain a continuous and balanced flow. In these settings, we have to propose a pipeline size that is larger than the number of locations in the inbound buffers of workstations. Several methods can be proposed to determine the pipeline size. We propose to define the pipeline size for workstation i based on the capacity of the workstation and the average transport time to the workstation. In more formal terms, we introduce the following notation:

   PS_i = pipeline size of workstation i expressed in the number of items (TSUs).

   TT_i = Average transport time (in minutes) from all source points (e.g., loading points on a sorter) to workstation i assuming a free flow situation.

   C_i = Capacity of workstation i in *ipm* (items per minute).

   The $allowance$ finally denotes a time allowance parameter (in minutes) to account for traffic delays. Now, the pipeline size of workstation i is defined according to the following expression:

$$PS_i = C_i \cdot (TT_i + allowance)$$

Moreover, in order to prevent a deadlock situation in this approach, we also put a limit on the maximum occupation of the loop sorter. We do not allow further retrievals by cranes when the loop occupancy reaches 95%, even if the pipeline limits are not reached yet.

When the pipeline to a certain destination is full, system controllers react by blocking further TSUs from being retrieved or routed to this destination until space becomes available in the pipeline. Therefore, the pipeline size is an important parameter that is used to control material flow in the system and to prevent overflows.

2.2.2.2 Scheduling inbound containers

In Section 2.2.2.1, we mentioned that in parcel & postal sorting, there are no diverts or cranes to load TSUs on sorters, as the MHS (in this sector) is basically the sorter itself. In this case, loading sorters (with TSUs) is a scheduling problem that is already at Scope 2 of our analysis because it is not handled by the automated system itself, but by operators loading TSUs manually (see Figure 1.1 in Section 1.2.1). This loading problem may also apply to some simple BHSs in small airports where bags from incoming ULDs[9] (see Figure 1.5 in Section 1.3.1) are loaded immediately on the sorter with no preceding stages (e.g., routing in networks). In this problem, we want to schedule the inbound containers (the inflow) in order to optimize the outflow of the sorters.

We study sorting systems using conveyors in distribution, where incoming TSUs are not assigned (yet) to a certain order (or destination) and so they are always stored first. Hence, the problem of optimizing the outflow by scheduling the inflow is not clearly defined. In addition, this scheduling problem is not critical (in distribution) for several reasons that relate to the contrast that we make between baggage handling and parcel & postal sorting on the one hand, and distribution on the other hand (see Section 1.3.1). The following arguments apply:

- The time pressure (in baggage handling and in parcel & postal sorting) makes it necessary to have sound scheduling approaches that can handle the strict time schedules.
- Item uniqueness in these two sectors induce additional pressure to sort specific TSUs in time (their destination is generally known, as opposed to the situation in distribution).
- The unpredictable arrivals in these two sectors also make the problem of scheduling inbound containers more challenging.

In Chapter 5, we study scheduling algorithms that consider loads in transport within sorting systems in order to schedule incoming containers in parcel & postal sorting and baggage handling. These scheduling algorithms are not part of a scheduler within the MHS's control architecture, but can be implemented in software tools for the MHSs' users. Scheduling inbound containers should support the functional requirements, e.g.,

9 Unit Load Devices, which are containers used in air transport of baggage.

management of buffers, dealing with urgent items, and starvation avoidance (see Section 1.3.2). Chapter 5 provides a further analysis of this scheduling problem.

There are MHSs in distribution that do share the characteristics of MHSs in baggage handling and in parcel & postal sorting. Although we do not study these systems, we can still model them using the methods developed in Chapter 5 or the generic material flow model of Chapter 3.

2.2.2.3 Routing arriving TSUs

At some point in the transport operation in baggage handling, a decision has to be made on routing arriving bags either to the sorter system or to the ASRS. This choice is made by the arrivals' divert controller (see Chapter 3) using system information and status of destinations.

Obviously, bags are routed to the ASRS when the build for the corresponding flight is not open yet. Moreover, if the build is open and the pipeline(s) of the destined workstation(s) is (are) not full, then the bags are routed to the main sorter. However, if the pipeline(s) is (are) full then, in order to maintain a controlled flow on the main sorter, it may be beneficial to route bags to the ASRS and delegate the scheduling task to crane controllers there. The latter option should not be used for urgent bags as it may cause them to miss their flights. In this case, it makes sense to allow recirculation on the main sorter due to routing bags directly to the laterals, although they are busy.

2.2.2.4 Routing TSUs in networks

In large scale MHSs, there are often service stations, e.g., screening machines (Figure 2.5), which are available at alternative systems. In such configurations, a divert controller has to decide to which system to divert an incoming TSU.

The proposed control logic for these routing problems is dynamic and based on the status of the system. In order to balance loads on parallel systems and yet consider travel times and service rates, we make routing decisions based on the expected throughput of the alternative parallel systems.

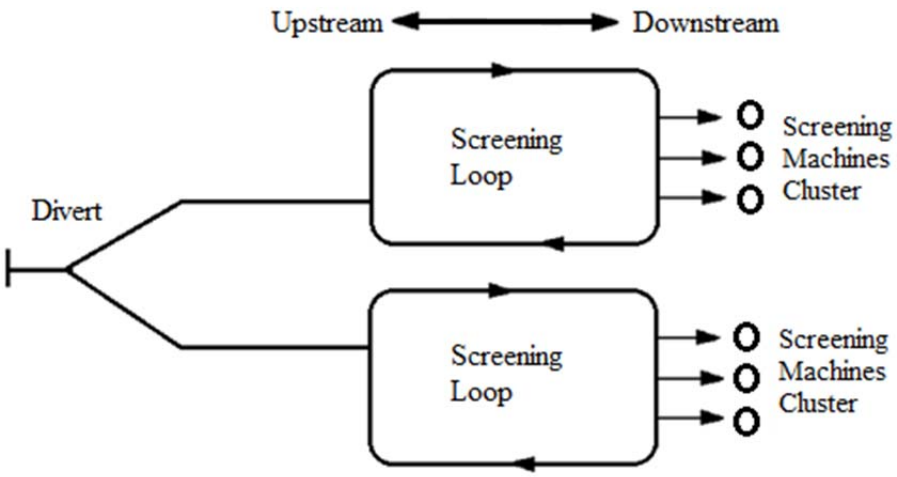


Figure 2.5. Routing in parallel systems.

We develop a *dashboard* logic where downstream controllers post expected throughput times (and possibly other information) on their dashboards. For example, machine cluster controllers (see Figure 2.6) post an expected throughput time for a TSU to be processed (given the number of TSUs already in the pipeline and the service rate of the machines). Upstream controllers use this information to make routing decisions. In this dynamic control, downstream controllers need information about TSUs in the pipeline from upstream controllers, in order to estimate throughput times. This control logic helps in the functional requirements of saturation management, prevention of imbalanced queues and recirculation, management of buffers, and dealing with disruptions. For example, fewer TSUs should go to the system having a lower service rate (e.g., due to a failed component or congestion).

When there is more than one destination to which an incoming TSU can proceed, we calculate the expected throughput time per destination. Upstream flow is always blocked when the system has used all available capacity. System control may plan to occupy less than the available system capacity, in order to always leave an escape and avoid deadlocks or saturation.

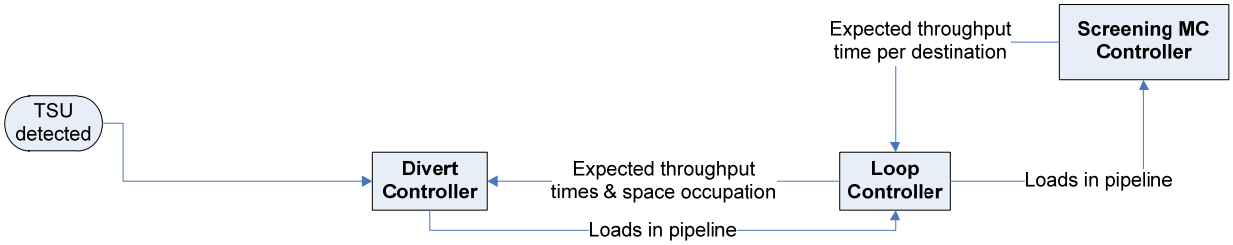


Figure 2.6. Communications for dynamic routing.

We emphasize that the standardized controllers and communication interfaces allow the implementation of the same control logic on different system layouts by merely defining connected controllers upstream and downstream for each component. In this thesis, we examine the applicability of this routing approach in two different settings: Chapter 3 presents an implementation in a distribution system, whereas Chapter 4 deals with an implementation in a large baggage handling system.

2.2.3 Local traffic control

At the local traffic control, we deal with decision-making processes that can be studied independently as they do not affect the overall control structure. At this level, we introduce two main processes: the space allocation in merge configurations and cranes' storage cycles.

2.2.3.1 Space allocation in merge configurations

In merge configurations, we have to allocate free spaces on a merge conveyor (e.g., a sorting loop) to TSUs waiting to enter this conveyor from several infeeds (Figure 2.7). Scheduling crane retrievals, scheduling inbound containers, and routing arrivals are all scheduling processes that result in decisions to load infeeds. Once infeed loading decisions are taken at the scheduling level, we can use a generic local traffic control

algorithm in such a configuration, e.g., by a loop controller. This problem is particularly important for parcel & postal sorting systems, which handle large numbers of TSUs within strict time limits, and so need an efficient merge operation. This local traffic problem has to satisfy several functional requirements, e.g., labor efficiency for operators loading the infeeds. Space allocation is a local traffic problem and so, by definition (see Section 2.1), can be dealt with independently.

This local traffic control problem entails a certain level of complexity and requires a decision-making algorithm. Therefore, we dedicate Chapter 6 to develop a space allocation algorithm that can be incorporated in the generic control architecture for merge configurations.

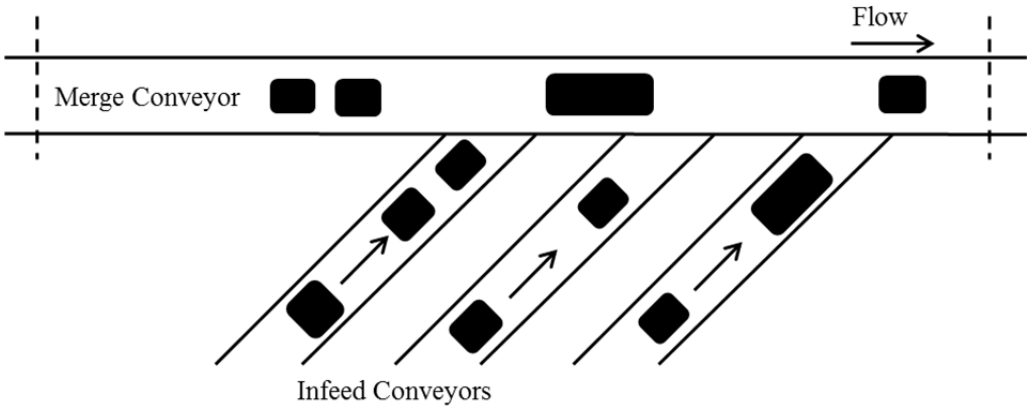


Figure 2.7. Merge configuration.

2.2.3.2 Cranes' storage cycles

Cranes execute storage cycles to store TSUs waiting in their inbound buffers. Higher levels of control assign TSUs to a certain storage rack via a certain crane and route them to the crane. However, regardless of these higher level decisions, once TSUs arrive at a certain crane, the storage operation is similar for all relevant industrial sectors. Since the process boils down to moving TSUs from inbound buffers to empty storage locations in specific racks, we can apply a generic control logic at the local traffic level.

The impact of the decision on where to store a certain TSU within a rack differs according to the user's environment and nature of the stored items. We studied MHSs where selecting an empty location in a certain rack to store an incoming TSU has a minor impact on other system resources or on the overall system performance. Hence, we do not analyze this operation in detail. However, for systems in which the execution of this operation is critical, an algorithm from literature or specific control rules can be implemented in the storage planner or locally in the crane controller.

2.3 Concluding remarks

In the first part of this chapter, we analyzed the forms of control that are the basic structure of any control architecture, while keeping in mind hybrid forms that can result in variants of the basic forms of control. We then evaluated the suitability of these alternative forms of control to our problem and excluded the two extremes

(centralized control and heterarchical control). Next, we built upon the nature of decision-making in MHSs to propose a concept control architecture that entails hierarchical levels of control and generic controllers on different levels. The concept control architecture is a variant of the modified hierarchical form of control, which uses the strong points of heterarchical control architectures (e.g., modular and robust design) and of hierarchical control architectures (e.g., delegation of lower level decisions to lower levels in the hierarchy and higher level coordination). In the second part of this chapter, we presented the main decision-making processes and indicated the potential to model them generically.

In the proposed control architecture, we note that schedulers are the controllers responsible for workload control, because they decide on task execution times, e.g., when to execute retrieval tasks. Moreover, pipeline size limitations reflect a *pull system* for material flow, which we use to avoid congestion, overflow of buffers, saturation, imbalances in loads among buffers or parallel systems, and to support other functional requirements. At the local traffic level, we deal with materials physically moving as a result of scheduling decisions. Therefore, local traffic control has a small impact on the amount of materials in transport.

In the following chapters, we detail the concept control architecture further and test its applicability to MHSs in different settings reflecting the different industrial sectors. Then, we analyze the extent to which we manage to maintain a generic control that conforms to the functional requirements and design requirements as defined in Section 1.3.2. To this end, Chapter 3 provides different applications of the control architecture, whereas Chapter 4 demonstrates an implementation of the generic control architecture in a business case of a large baggage handling system.

Chapter 3

Applications Of The Planning And Scheduling Control Modules[10]

In Chapter 2, we proposed a *concept* of a generic control architecture for MHSs in different industrial sectors. We have presented the control structure and illustrated the decision-making processes at different levels of control.

In this chapter, we develop the concept control architecture into a *concrete* control architecture and validate it. To this end, we apply control modules of the generic control architecture (Chapter 2) to a material flow model with an ASRS and a sorter system with build workstations. These system elements are common to two industrial sectors: baggage handling and distribution, where build workstations can be order picking stations (in distribution) or laterals (in baggage handling). Therefore, we set up this model to be applicable to MHSs in distribution and baggage handling. Moreover, we report on performance and analyze how far we can control the two industrial sectors generically in terms of software implementation. Note that parcel & postal sorting is excluded from this model because MHSs in this sector do not use ASRSs or build workstations.

The system elements described above require the implementation of the decision-making processes we defined at the planning level (see Chapter 2) and a number of decision-making processes at scheduling level. In order to implement the generic routing approach (Chapter 2), which is at the scheduling level, and test its performance, we present another system model with a routing configuration. We also compare the generic routing approach to current practice. In the aforementioned applications, we mainly implement planning and scheduling control modules and include (generic) local traffic control modules in an aggregate manner.

Section 3.1 presents a generic MHS model, which can be tuned to simulate MHSs in different industrial sectors. Section 3.2 presents a modification to the system base modeled in Section 3.1 by changing the system layout and equipment to simulate a different distribution system. The changes result in a routing configuration, which requires an additional control module to be included in the control architecture (which is applied in Section 3.1). This additional control module is routing TSUs in networks (see Section 2.2.2.4). Finally, Section 3.3 ends with concluding remarks.

[10] *This chapter is based on Haneyah et al. (2013b).*

3.1 A generic material flow model

In this section, we introduce a generic material flow model, which uses an ASRS as the main element. The control modules applied reflect the first building blocks of the concept control architecture proposed in Chapter 2. Section 3.1.1 presents the system model and relevant components. Section 3.1.2 presents the implementation of the proposed control architecture on the generic material flow model, discusses experimental results, and analyzes the generality of the software implementation. Finally, Section 3.1.3 presents concluding remarks for the generic material flow model.

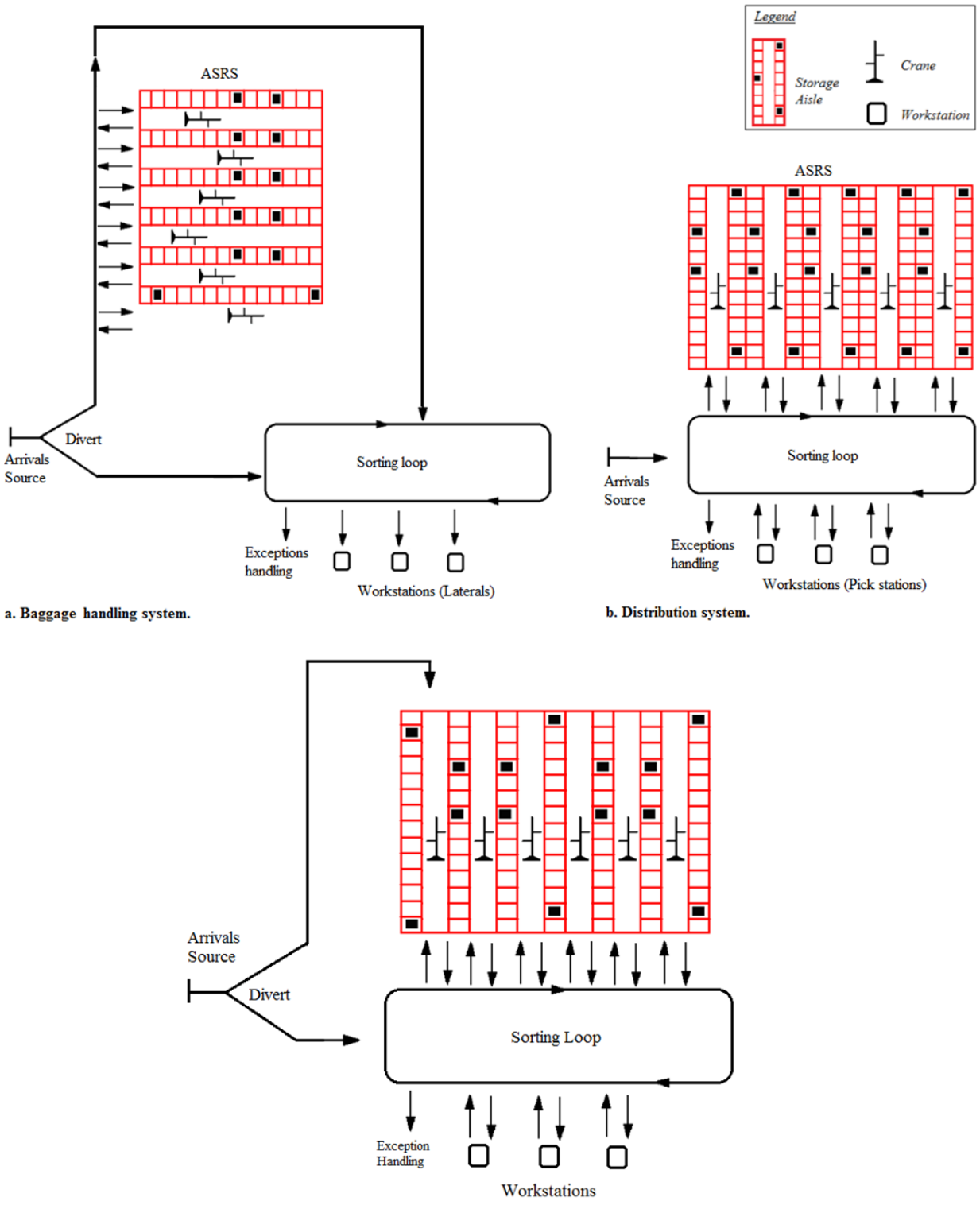
3.1.1 A generic MHS model

Based on the analysis of two MHSs operating in two different sectors (Baggage Handling and Distribution; see Figures 3.1a and 3.1b), we construct a model of a generic MHS (see Figure 3.1c). The generic system entails an ASRS with aisles and cranes, a conveyor in loop configuration as a sorter system, and inbound and outbound conveyors connecting the cranes and workstations to the loop. However, we do not display a comprehensive MHS, i.e., in a typical baggage handling system, there are baggage screening systems entailing other loop conveyors and clusters of screening machines. Chapter 4 addresses these screening systems, which are upstream the ASRS and the main sorter system.

As a general rule in the BHSs that we studied, an arriving TSU goes directly to the sorter system if the build for the corresponding flight is open on one or more workstations; otherwise it is diverted to the ASRS. When the build for a flight is open, relevant bags are released from the ASRS to the workstation(s) assigned to handle the bags of this flight.

In the distribution systems that we studied, arriving TSUs are always stored first, so they never go to workstations directly. TSUs are full when they enter the system. However, when a TSU is used for an order, some SKUs are picked from it at a workstation. If the TSU is not completely consumed, it becomes a broken TSU that has to return to the ASRS. These broken totes also have to return to the ASRS first before being reused for other orders.

We highlight that in distribution arriving TSUs go to the ASRS via the main sorter system (see Figure 3.1b), whereas in baggage handling, arrivals enter the ASRS via another conveyor-based route (see Figure 3.1a). The main sorter system only handles the outflow baggage from the ASRS to the workstations. The reason is that (in baggage handling) the outflow of bags is critical and should not be disrupted by the inflow. Therefore, these two flows are decoupled in terms of the handling equipment. However, this decoupling also exists in some of the large distribution systems, which we study in Section 3.2.



a. Baggage handling system.

b. Distribution system.

c. Generic MHS model.

Figure 3.1. MHS models.

In order to apply generic control methods, we propose the generic MHS model depicted in Figure 3.1c, in which we do not model the transport route leading arriving TSUs (in baggage handling) to the ASRS in detail. We model this route in aggregate terms because it is not critical from a control point of view and it does not exist in the

distribution system that we study for this model. The generic MHS model can still reflect a different distribution system with a different layout, e.g., where inbound totes do not use the conveyor loop. We apply the generic control elements to both industrial sectors by parameterizing the generic MHS model to simulate distinct sectors. For example, the conveyor route from the ASRS to the sorter in Figure 3.1a is incorporated in Figure 3.1c by longer travel times on the sorting loop from the ASRS to workstations, where travel times are configurable parameters in a *travel times matrix* (see Section 3.1.2) in the generic model. The recirculation time on the sorting loop follows the actual travel times (from practice) for each industrial sector. Moreover, when the generic MHS is configured as a distribution system, then arriving TSUs proceed to the ASRS via the sorting loop and do not use the other (baggage handling) route from the divert to the ASRS. In this case, the generic model reduces to the distribution system model (Figure 3.1b).

In this section, we focus on planning the flow of TSUs into the ASRS and out of it to the workstations via the main sorting loop. To this end, we model the following system components:

- *Storage aisles:* we model storage aisles in aggregate terms, i.e., we do not model the exact storage locations within an aisle.
- *Cranes:* cranes are modeled with their inbound and outbound buffers. Travel times for cranes are taken from tested distributions at our industrial partner.
- *Workstations:* workstations are modeled with their inbound and outbound buffers.
- *Divert:* we model the first entry point of TSUs with an *arrivals source* to generate TSUs and a divert that makes decisions on routing TSUs to the sorter or to the ASRS.
- *Sorting loop:* the loop is modeled at some level of aggregation, where we keep track of the loop capacity and travel times on the loop, but we do not keep track of the location of every point on the loop at every moment in time.
- *Exception handling outfeed:* this is a special type of conveyor where TSUs can be diverted in some cases, e.g., when a bag misses its flight.

3.1.2 Implementation and analysis

The scope of the decision-making processes (see Section 2.2) implemented in this generic system model is as follows: the planning level is fully implemented. With regard to the scheduling level, the relevant processes are scheduling crane retrievals and routing arriving TUSs (by the divert; see Figure 3.1c). Finally, local traffic control problems introduced in Chapter 2 are implemented at a certain level of aggregation, because they exhibit identical control procedures for the different industrial sectors.

3.1.2.1 Experimental setup

We use the UGS-Tecnomatix Plant Simulation software to build a simulation model, which is scalable to different system sizes, in terms of the number of aisles, cranes, and workstations. Moreover, our model is flexible to different system settings regarding layout configurations (e.g., accessibility of cranes to storage racks) and

capacities of system's resources (e.g., buffers and cranes). In order to model a certain system, the following modeling parameters have to be configured in the simulation model:

- Inbound/outbound buffer sizes of workstations and cranes.
- Aisle capacities, in number of storage locations per aisle.
- Crane capacity in the number of TSU locations on the crane. In our model, a crane can carry 2 TSUs simultaneously in baggage handling and 4 TSUs simultaneously in distribution. Moreover, we use a cycle time distribution for each crane type depending on its characteristics.
- Workstation capacity in terms of the number of TSUs processed per minute.
- Pipeline sizes of workstations: For the distribution case we follow the first approach of setting the pipeline sizes, i.e., the pipeline size is equal to the number of locations in the inbound buffer of the workstation (see Section 2.2.2.1). For the baggage handling case, we follow the second approach, where we have to tune the pipeline size parameter (see Section 2.2.2.1).
- Sorting loop speed and size in number of TSU locations available.
- Travel times matrix: this matrix provides travel times on the sorter loop for every source and destination pair.
- Aisle-crane accessibility matrix. This matrix shows what cranes are operational in what aisles. In distribution, a storage rack is accessible by one crane only. However, in baggage handling, due to high reliability requirements, a redundant system design is in place, where two cranes can access the same storage rack.

Moreover, the following parameters are control parameters, which are implemented in the simulation model, as well as in the actual software of MHSs:

- Maximum number of orders simultaneously active on a workstation. In distribution, it is common to have multiple orders processed simultaneously. In this distribution system setting, 6 orders are active simultaneously on a workstation.
- Planned orders threshold. This refers to the number of orders ready for activation on any workstation requiring work. These are to be planned in advance by the build planner (in distribution) and are typically equal to the number of active workstations. This is the minimum number of orders needed for activation on any workstation requiring a new order. We do not plan more than this minimum value so that we can dynamically plan new orders based on the status of the system.

For baggage handling, we configure the settings to model a baggage handling scenario according to a major European airport, which was a reference system for our study. This system has 18 workstations and 13 cranes operating on 12 storage racks, so that each storage rack is accessible by two cranes. The service time of workstations depends on the number of operators present, where each operator can handle on average 2 bags/minute. The standard setting is 2 operators per lateral, and we translate this into a uniformly distributed service time between 10 and 20 seconds per bag. We use data sets regarding flight schedules and baggage arrivals from the same system, for a 24 hour day of operation. The flight schedules are known at the beginning of the

operation. An order in baggage handling refers to the baggage required for a certain flight, where each order line is a unique bag that is assigned to the flight.

For distribution, we configure the settings of our model to model an existing automated distribution center in the Netherlands, which has 3 workstations, and 5 cranes in 5 storage aisles. The service time of a workstation is uniformly distributed between 5 and 15 seconds per TSU, assuming one operator per workstation that can handle on average 6 TSUs/minute. For data sets in distribution, we run experiments according to one of the following order structure scenarios, for one day shift:

- Big orders: 15-25 order lines/order.
- Small orders: 1-5 order lines/order.

We generate the orders randomly, where the number of order lines per order is determined using a uniform distribution that uses the order size limits as parameters. Then, for each order line, we randomly select an SKU. All orders to be satisfied for a day of operation are generated at the beginning of the simulation, e.g., before the order picking starts. Orders might have different priorities that result from, e.g., due times. Therefore, we arrange orders to be processed according to decreasing priority. Moreover, at the beginning of the simulation, we generate a replenishment flow of new TSUs to fill the ASRS (which starts empty). During the simulation, although we have multiple TSUs per SKU, we assume that to satisfy an order line, one TSU (holding the required SKU) is always sufficient even if the TSU is broken. In this case, a TSU is never consumed completely and so there is always a returning flow of broken TSUs from workstations to the ASRS. This returning flow compensates for the replenishment flow of new TSUs during daily operation.

We concentrate on the logistics and transport issues of the system. Therefore, issues related to inventory profiles, i.e., inventory position, replenishment times, and the number of different SKUs, are not interesting factors to vary in our model. These issues are the responsibility of the MHS's user, who has to make sure that there is enough inventory in the system. We use a standard value of a 1000 different SKUs in all experiments. These modeling assumptions are based on common practice and modeling cases from our industrial partner.

Experimental control rules for assigning inbound TSUs to aisles are as follows:

1. *SKU distribution:* A rule often used in practice is to select the aisle containing the fewest TSUs of the incoming SKU. In case of ties, the criterion will be to select the aisle with the lowest total number of TSUs. In baggage handling, a control rule at the same level of detail would be to select the aisle containing the fewest bags (TSUs) of the same flight. However, for baggage handling, this level of detail is not sensible for two reasons. First, if we do not deliberately distribute bags of the same flight over aisles, then it is very unlikely that bags of the same flight end up in the same storage rack. This is because (in large airports) baggage arrivals are stochastic and at any point in time a mix of baggage for many different flights arrives. Second, the ASRS in a BHS has sufficient capacity to retrieve all bags of a certain flight in time even if they are in the same storage rack, especially because two cranes can access the same

rack. Therefore, a control rule with this level of detail is not applied in practice for BHSs.

2. *Aggregate TSU distribution:* In distribution, this means to select the aisle having the lowest number of *broken* TSUs in aggregate terms regardless of what SKU they hold (ties are broken as in rule 1). In baggage handling, this rule means to select the aisle having the lowest number of TSUs regardless of the flights to which these TSUs belong.
3. *Round-Robin:* Simply store in aisles sequentially for every incoming TSU regardless of the number of broken TSUs, SKU distribution, or flights.

In distribution, using the aggregate number of broken rather than full TSUs in the aisle makes more sense (control rule 2), because stock reservation always looks for broken TSUs first, and then distributing these over aisles contributes to workload balancing among cranes. In all control rules, there should be storage locations available in the selected aisle and at least one crane active on the aisle to perform the storage operation. Otherwise, the aisle is not a candidate.

Note that in this simulation study, we model a daily operation both in baggage handling and in distribution, and not the behavior in the long run. Therefore, the dynamics at the beginning and at the end of the day are part of the study, so the data collection starts from the beginning of the simulation.

3.1.2.2 General results on performance

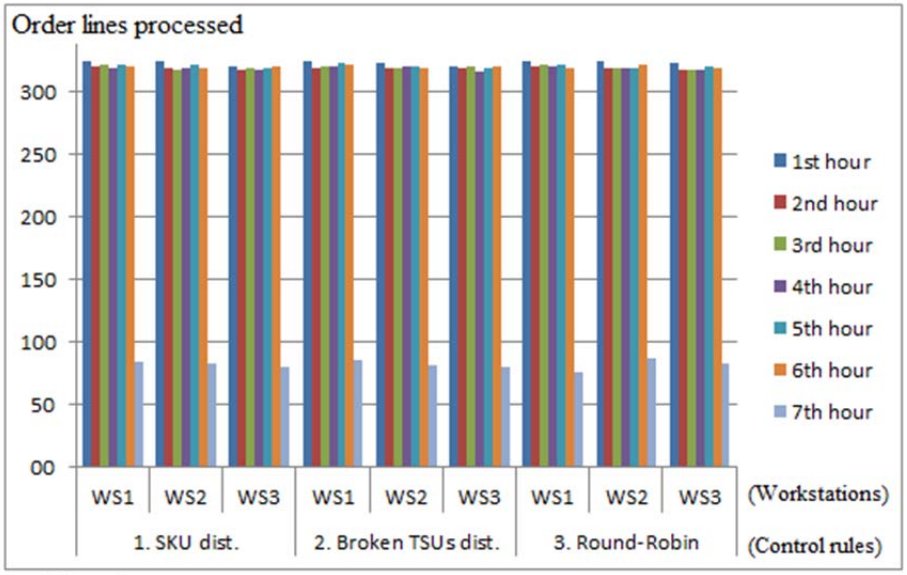
Distribution

Figure 3.2 shows the number of order lines processed in every hour of operation (each column represents the throughput of an hour), for each workstation, order size, and inbound control rule. We observe that there are no significant differences in terms of throughput, when considering various control rules for inbound flows. Naturally, the last hour has fewer order lines processed due to the end of a shift and is therefore excluded from performance measures. The average number of order lines processed per hour was 314 and 320, with 87% and 88% utilization of workstations, for small and big orders respectively.

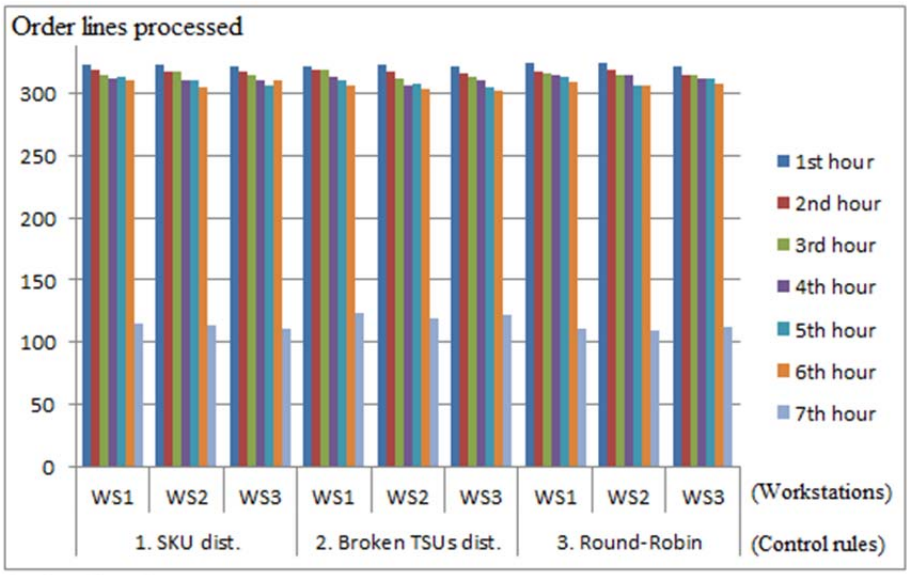
Note that smaller orders cause throughput levels to decline. To analyze this trend further, consider orders that are extremely small (1 order line per order), then 6 orders active on a workstation lead to only 6 TSUs in transport. In this case, as an order is picked and a new order is to be activated, more delays are expected until the next TSU is scheduled, retrieved, and transported. When the order is big, each order requires several TSUs that are in transport and keep the workstation busy. We experimented with orders of 1 order line each and found that the number of picks per hour drops to 271, and utilization of workstations drops to 75%.

Moreover, we notice that in the small orders scenario the number of picks tends to decrease as time goes by (see Figure 3.2). The interpretation is that when orders are small, the number of orders that have to be processed is higher, in order to generate a similar number of total order lines to simulate. Therefore, the probability that several orders simultaneously need the same SKU becomes higher. In this case, as more TSUs are simultaneously needed by several orders, delays may occur until reserved TSUs

are available for a specific order. Note that there is a limited number of TSUs per SKU in the system.



a. Big orders



b. Small orders

Figure 3.2. Throughput levels per workstation under different control rules.

With regard to inbound control rules, rules 2 and 3 do not show any deterioration in throughput measures over a whole day of operation compared to rule 1, which is common practice. Hence, we recommend using one of these rules as they require simpler software implementation and result in synergy in control with baggage handling. At this point, we recommend abandoning rule 1, but recommending rule 2 or 3 depends on the results from the baggage handling scenario.

Baggage Handling

We found that setting the time allowance parameter to zero in the second approach for determining the pipeline size (see Section 2.2.2.1) results in an underestimated pipeline size. This causes insufficient material flow in the system. Many bags were

missed (irregularity rate = 13.2) and the utilization of system resources was low. The explanation is that the workstations were often idle, because there was an insufficient flow of TSUs to the workstations. In fact, a time allowance of zero assumes a free flow situation, where in reality delays occur, e.g. due to waiting times to merge on the loop which in turn may cause blocking entrance of new bags due to the maximally allowed pipeline size. Moreover, TSUs that are scheduled and not physically in transport contribute to pipeline occupation.

Therefore, we have to propose a positive value for the time allowance parameter in order to account for retrieval time of cranes and traffic delays. We tested the system performance given different time allowance values and under different control rules (see Figure 3.3). Based on these simulation results, we find that a time allowance of 170 seconds leads to a good performance, with an irregularity rate of 0.06 (bags lost in every 1000 bags). This irregularity rate is achievable by both control rules 2 and 3. However, we emphasize the time allowance is a layout-specific configurable parameter.

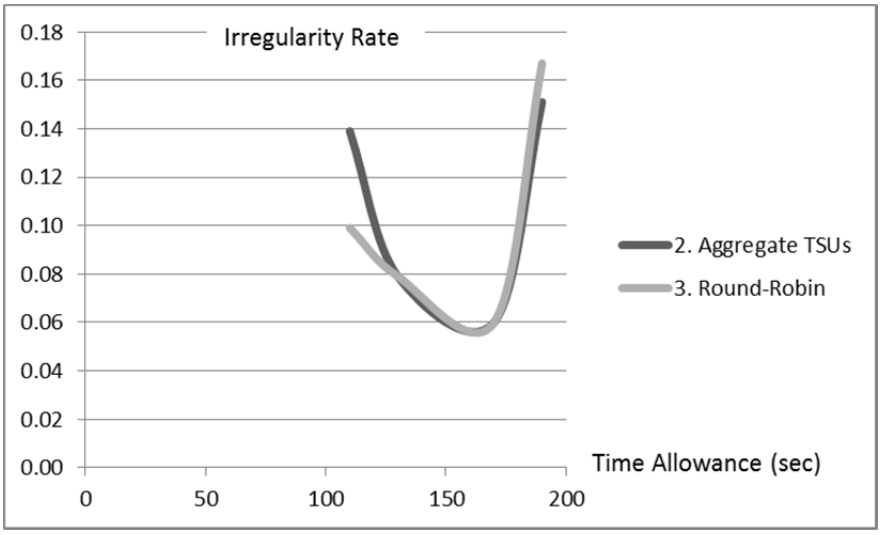


Figure 3.3. Irregularity rate given different allowance values and different control rules.

3.1.2.3 Analysis of generality

We have implemented the proposed control architecture in the material flow model to analyze to what extent generic software code can be maintained for the two different industrial sectors: baggage handling and distribution. We focus on the analysis of our software code that deals with the logistic planning and control of the system, which is often different per industrial sector (i.e., decision-making functions). Moreover, we do not analyze Application Programming Interfaces (APIs) or Graphical User Interfaces (GUIs) that need to be provided to the hardware (see Section 1.4). The latter applications are standard interfaces that are available at our industrial partner and at other software developers. These applications are used to drive the hardware, but are not part of the decision-making functions. However, we provide standard interfaces between the decision-making procedures and decision-making units (controllers). In other words, we focus on the decision-making functions in detail (the control architecture), but not on the implementation of the control architecture in a real-life

software system, and on its connection to a real-world installation (the software architecture).

Our perspective on software is therefore on an abstract level and deals with the code we used to implement these decision-making functions. This perspective supports our generic approach to decision-making in providing a generic control architecture, where connecting this control architecture to the hardware of a certain MHS is a different issue.

Our target is to keep the control generic, but at some point we had to deviate to satisfy sector-specific requirements. As a matter of fact, for decision-making processes that exist in both industrial sectors, we implemented a software code of which 84% (in terms of non-redundant lines of code) is used identically by both sectors, while the remaining 16% vary. We present these percentages to give insight into the potential synergy between the different industrial sectors. These percentages are dependent on the implementation in our simulation and so they may vary in other implementations or other simulation models or packages. However, the generic design of our decision-making processes leads to a high degree of synergy even with different implementations.

We note that our analysis is focused on the software related to decision-making functions, which addresses the differences in these sectors. As we claimed earlier in Section 1.4, standardization in software that is not related to decision-making should be the rule rather than the exception, and so including it in our analysis has to bring even more synergy.

We claim that our control architecture is generic in its applicability to different systems, in the sense that implementers need to understand and customize only 16 % of the code, where sector-specific elements need to be handled. Implementers should understand the majority of the code (roughly 84%) at a high level using standard procedures and reusable modules. To explain this claim, we use Figure 3.4 in which we provide a representation of the relevant decision-making procedures, where we include the main decision-making procedures for the main controllers we have in the model (the build planner, the storage planner, an object from the crane controller class, an object from the workstation controller class, the loop controller, and the arrivals divert controller). To keep Figure 3.4 readable, we do not show many of the standard procedures, databases and their connections, or the communication links on assigning orders to workstations and retrieval tasks to cranes (shown in Figures 2.3 and 2.4).

At this point, we need to further analyze decision-making procedures where the software is not the same and explain why is it inevitable to deviate from the generic code. Therefore, the remainder of this section is dedicated to the analysis of the decision-making procedures that are not standardized for both industrial sectors (16% of the software code). Moreover, we analyze the procedures that can be incorporated in the control architecture or omitted depending on the requirements of certain industrial sectors (in the form of a plug-and-work mechanism).

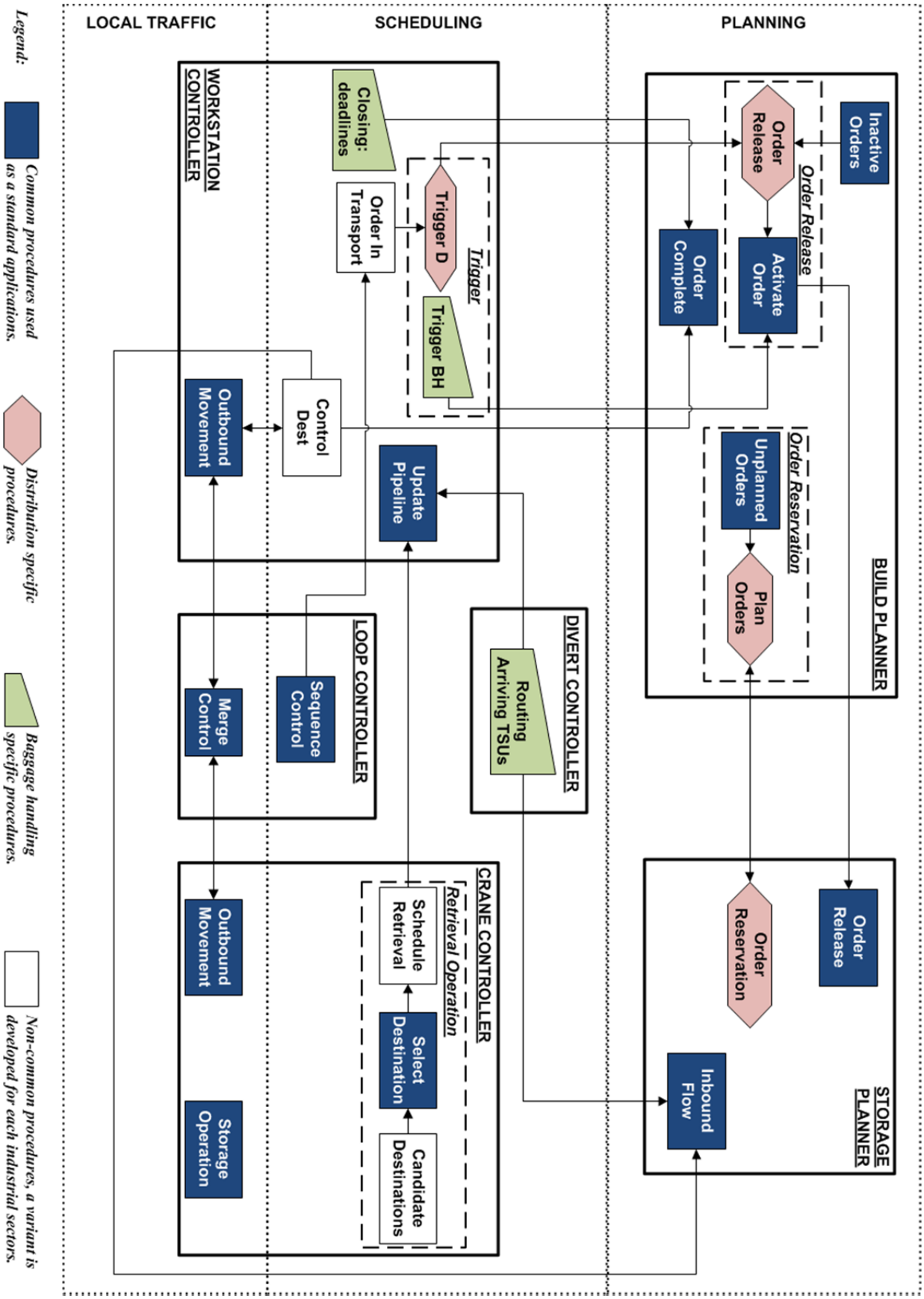


Figure 3.4. Main decision-making procedures in the control architecture.

The order reservation process

At the planning level, Figure 3.4 shows the 'order reservation' process as a non-common process in both the build planner and the storage planner. As mentioned earlier, this process brings distribution at the same level of detail as baggage handling.

Figure 3.4 shows the components of this process in the build planner, i.e., the 'Unplanned Orders' and the 'Plan Orders' procedures. The 'Unplanned Orders' procedure checks (in connection with the orders database) whether there are still unplanned orders and triggers the 'Plan Orders' procedure to plan new orders given the threshold conditions (see Section 2.2.1). This procedure is active in our model for the two industrial sectors. However, in baggage handling it always finds that all orders in the database (in this case flights) are planned, and so it never triggers the 'Plan Orders' procedure and in turn the 'Order Reservation' procedure in the storage planner is never used. These procedures are plug-and-work like procedures that can be simply removed from the architecture when implemented in baggage handling.

Triggering and releasing orders

In this point, we discuss two elements of Figure 3.4 together, because they are closely connected. First of all, the trigger process in the workstation controller differs per industrial sector, because this process is directly related to the different operational environment, which we described in Section 2.2.1. This is summarized as follows:

- In distribution: the trigger to activate a new order and the announcement of a complete order is based on work execution (late TSUs are waited for).
- In baggage handling: orders are triggered to start and are declared completed based on time schedules (late TSUs are missed).

In some other distribution systems, orders might also have strict due times. However, in the system we studied, orders are arranged beforehand according to their priority, but meeting a certain due time is not as strict. Therefore, in Figure 3.4 there are two variants of the 'Trigger' procedures, one per sector. This is an unavoidable sector-specific application. In distribution, the 'Sequence Control' procedure in the loop controller sends update messages about TSUs in transport to the destination workstation. Specifically, the 'Order In Transport' procedure receives these messages. In distribution, the 'Order In Transport' procedure checks whether all of the TSUs of an order are in transport and sends a message to the 'Trigger D' procedure. In turn, the 'Trigger D' procedure checks whether a new order should be activated and, if so, sends a message to the higher level build planner asking for a new order. This message is received by the 'Order Release' procedure. The latter procedure checks whether there are still any planned but inactivated orders. If so, then it selects an order to activate and sends a message to the 'Activate Order' procedure to activate the selected order on the triggering workstation.

In baggage handling, the 'Order In Transport' procedure does not send any outgoing messages and, therefore, part of the code in this procedure is unused for this system model. As a matter of fact, in baggage handling, the orders (in this case flights) are planned beforehand according to certain time schedules. Therefore, at certain moments in time the 'Trigger BH' procedure sends a message to the build planner to activate an

order (in this case the planned flight). Since it is already known what order is to be activated, this message is received directly by the 'Activate Order' procedure in the build planner.

We observe that in different industrial sectors we use the relevant variant of the 'Trigger' procedure in the workstation controller. Moreover, we find that the 'Order Release' procedure in the build planner is used only in distribution to search for an order to activate, if any, on the triggering workstation. The 'Activate Order' procedure is standard and is used in both sectors.

Controlling TSUs that leave the workstation

As TSUs are processed at workstations, they have different options to proceed for each of the industrial sectors. The procedure 'Control Dest' (Figure 3.4) is responsible for guiding these processed TSUs. In baggage handling, TSUs always leave the system and so the 'Control Dest' procedure instructs low level controllers to move the bag. However, in distribution, processed TSUs are often broken TSUs, which need to be returned to the ASRS. Therefore, they need to be announced and re-routed via the 'Inbound' procedure. Moreover, if a TSU is the last TSU of a certain order, then 'Control Dest' sends a message that the order is complete to the build planner.

The 'Control Dest' procedure needs to have a variant per industrial sector. It reflects an unavoidable system-specific application due to the different operational environments.

Order complete in baggage handling

In the previous point, we showed how the order is announced complete in a distribution system. In baggage handling, the procedure 'Closing' (Figure 3.4) is responsible for sending the order complete messages at certain moments in time, based on planned end times for flight loading operations.

Routing arriving TSUs

A sector-specific procedure 'Routing Arriving TSUs' is added in baggage handling to route arriving bags (see Section 2.2.2.3), because in baggage handling a scheduling decision has to be made on routing bags either directly to workstations or to the Early Bags Storage.

Scheduling crane retrievals

Scheduling crane retrievals is a vital decision-making process that has to adapt to sector-specific requirements, although it has the same basic structure in the crane controller. To understand this process better, we highlight an important operational difference in the systems we studied: in distribution multiple orders are simultaneously active on a single workstation, whereas in baggage handling a single order is active on one or more workstations simultaneously.

Figure 3.4 shows that the retrieval process consists of three main procedures, executed in each retrieval decision:

1. *Defining candidate destinations (for which a retrieval can be scheduled):* in distribution, the destinations of all active order line retrievals define the

candidate destinations. In baggage handling, we have to serve the most urgent flight first, and so those workstations on which the urgent flight is active are the candidate destinations.

2. *Selecting a candidate destination:* the selection among candidate destinations is identical for both industrial sectors and depends on least occupied pipeline.
3. *Scheduling a retrieval:* in distribution, we schedule a retrieval to the selected destination, based on the oldest active order first. In baggage handling, we schedule a bag for the selected workstation and the selected (urgent) flight.

In this process, the second procedure is applied identically in both industrial sectors. However, for each of the first and third procedures above we have a variant customized for baggage handling and a variant customized for distribution.

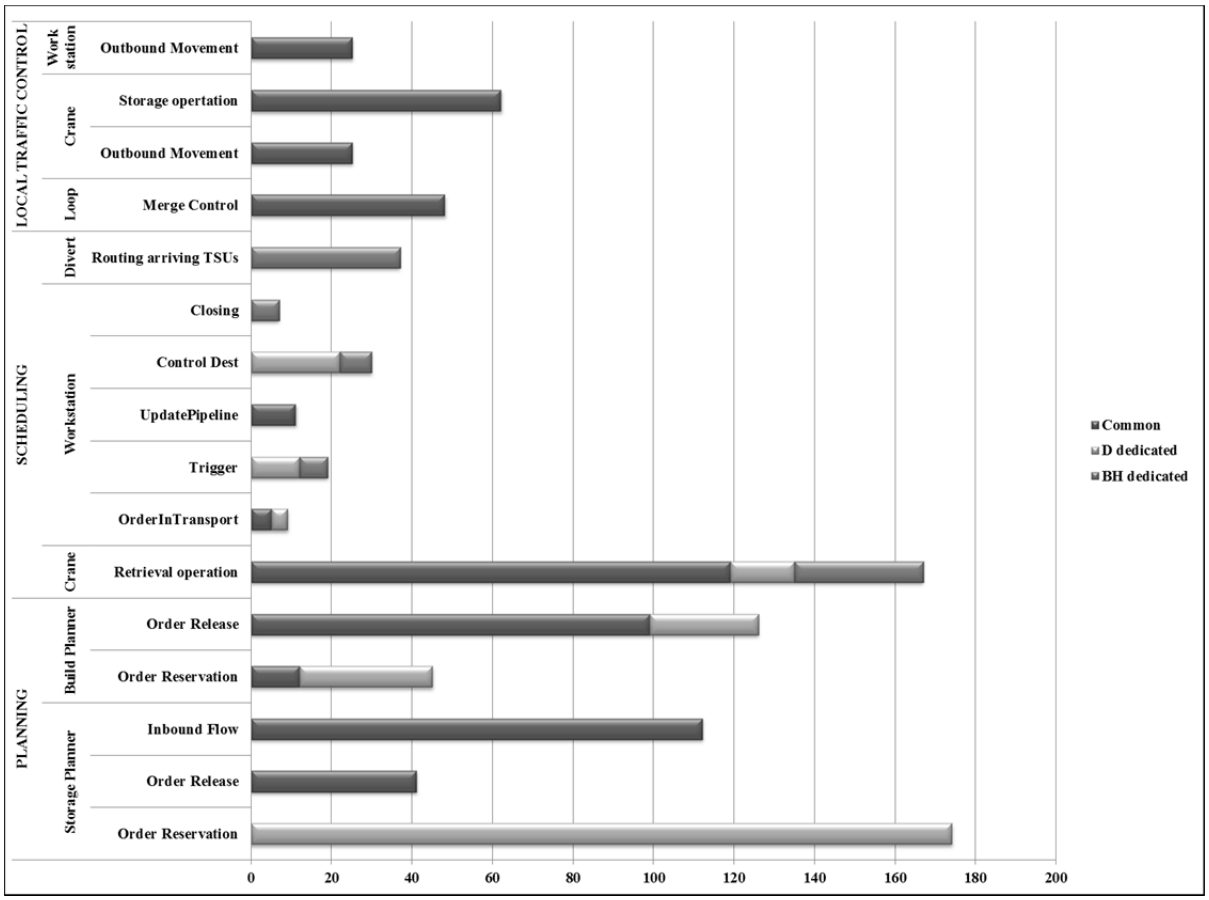


Figure 3.5. Commonality of code.

Overview

Having discussed the decision-making procedures in detail, we find it useful to come back to the decision-making processes in generic terms (see Section 2.2) and provide an overview of the commonality in software for these processes according to the implementation in the simulation model. Figure 3.5 presents the software coverage, i.e., commonality in the lines of code. In this sense, Figure 3.5 presents the number of lines of code in common and the numbers of lines dedicated per sector, for every main decision-making process. We note that decision-making processes at the local traffic control are common for both sectors, which should be the case for such lower level decisions that deal mainly with the movement of TSUs on the equipment. Variations

occur at the planning and scheduling levels, where the processes that are divided between the baggage handling or the distribution process with no common segments indicate that these processes require a variant per sector as described in the aforementioned analysis. On the other hand, processes that have dedicated segments per sector but also common segments, imply that certain procedures within these processes are common while others require a variant per sector. Finally, processes that are entirely dedicated refer to plug-and-work procedures.

3.1.3 Concluding Remarks

In this section, our main target was to provide a proof-of-concept for the applicability of generic control for MHSs in different industrial sectors. To this end, we presented a material flow model that is applicable in two different industrial sectors and applied generic control procedures on it building upon Chapter 2, which proposes a generic control architecture.

We provided a generic system model and explained the implementation of our control for this model. Next, we reported on general results and analyzed the generality of our control approach in detail. As a matter of fact, we managed to achieve a high level of synergy in control over common decision-making processes. This is demonstrated by a software code that is 84% identical for both the baggage handling and the distribution sectors. The generic structure of our decision-making processes facilitates a minimal deviation from the generic code. This is achieved because we do not hamper the overall structure of the decision-making processes when adapting to sector-specific rules. For example, the different decision-making criteria in scheduling crane retrievals (flight due time or oldest active order first) are easily integrated in specific procedures within the overall decision-making process. However, such differences are due to sector-specific parameters, e.g., plane departure schedules, which we have to adapt to.

In addition, we found that inbound flow control in distribution can be implemented in synergy with baggage handling. Inbound flow control can be implemented using aggregate information about TSUs in storage or using a simple control rule, i.e., Round-Robin. Uncommon decision-making processes that are specific to one sector, e.g., order reservation, are inevitable differences. These processes are built as functional add-ons to the generic control architecture, which do not hamper the generic structure and do not need special interfaces as we have standardized communications between controllers.

3.2 An MHS with a routing configuration

In this section, we develop the generic model of Section 3.1 further in order to (i) test the applicability of the generic control architecture (Chapter 2) on different configurations and (ii) to make an application of routing TSUs in networks, which is the scheduling process discussed in Section 2.2.2.4. This section is structured as follows: Section 3.2.1 presents the modified model. Next, Section 3.2.2 illustrates the implementation of the generic control architecture on the modified model and analyzes the performance. Finally, Section 3.2.3 ends with concluding remarks.

3.2.1 An MHS model with a routing configuration

We use an MHS from the distribution sector in the United Kingdom to modify the generic MHS model of Section 3.1.1 (see Figure 3.1). Figure 3.6 presents the modified MHS model, where the original sorter downstream the ASRS is removed and a large conveyors transport network is introduced. Moreover, there are two sorters, each with 2 workstations. The outbound flow of TSUs from the ASRS now proceeds to the first divert, i.e., Divert 1, which routes an incoming TSU downstream to one of the secondary diverts, i.e., Divert 2a or Divert 2b. Each of these secondary diverts then routes an incoming TSU to one of the sorters via the connected conveyors downstream. In this MHS, the transport operation from the ASRS to the sorters is done by conveyors.

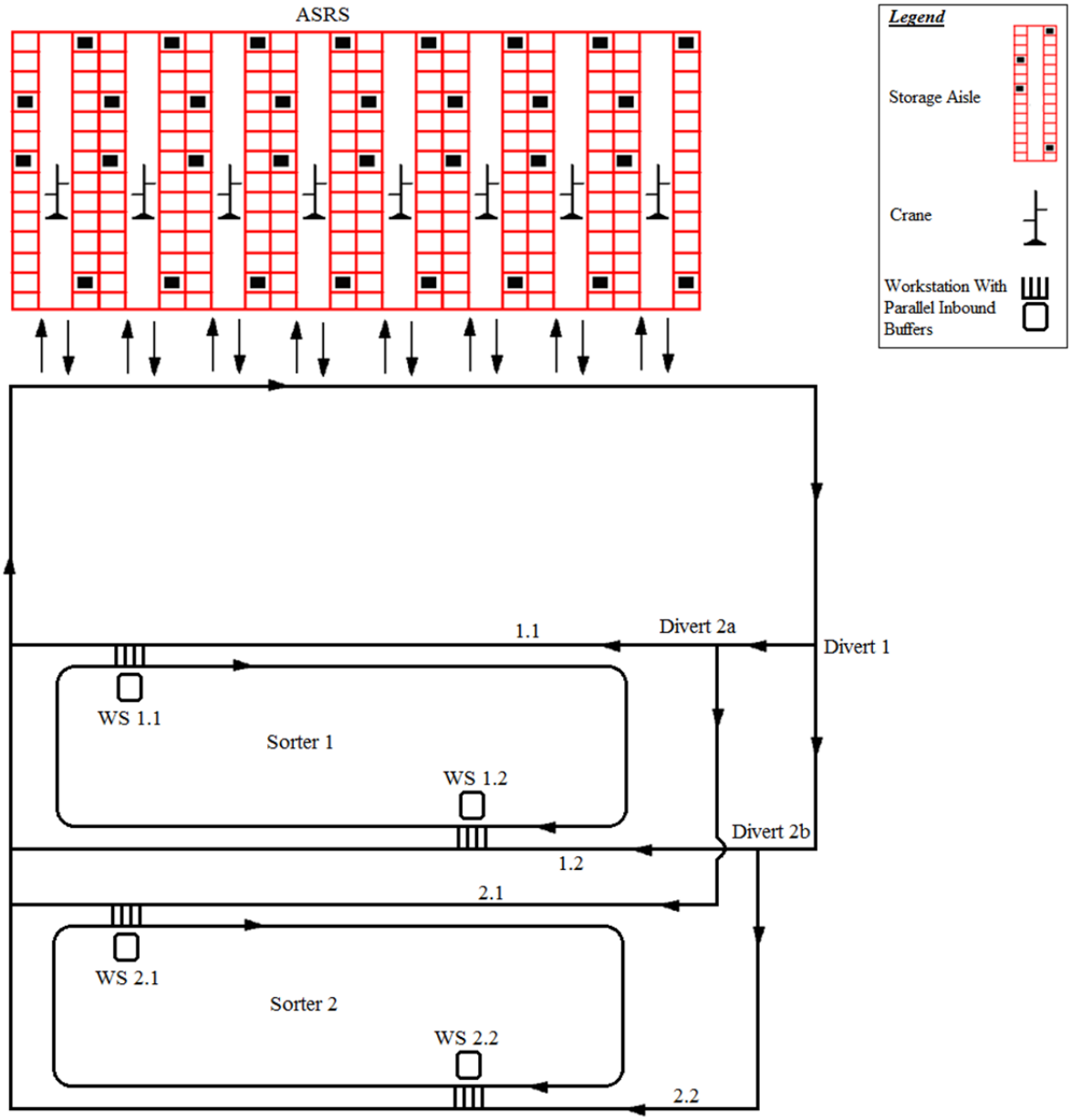


Figure 3.6. An MHS model with a routing configuration.

We note that in this large MHS, compared to the generic MHS model (Section 3.1), there is a larger transport network and there are more workstations. Therefore, a higher retrieval capacity from the ASRS is needed and so we extend the ASRS from 5 cranes in 5 storage aisles (for the distribution system in Section 3.1) to 8 cranes in 8 storage aisles. In this MHS, the outflow of TSUs from the ASRS (the product totes; see Section 1.3.1) do not use the sorters for transport. Actually, these outgoing TSUs use the conveyors network to eventually enter the inbound buffers of the workstations, where operators pick from them. The sorters are used to transport and consolidate customer/order totes (see Section 1.3.1) using additional equipment that we do not include in this model. In this setting, we do not model the sorter loops in detail as they are not the critical elements to the operation of this MHS. On the other hand, we focus on the conveyor lines upstream the sorters and on the control of the diverts, because these are the critical elements.

As mentioned before, this model reflects an MHS from the distribution sector, which has the following business rules:

- Picking for an order is done exclusively at one of the two sorters. On a certain sorter, it does not matter which of the two workstations processes a TSU for an order, since all TSUs are consolidated via the sorter.
- If a TSU cannot proceed to its destination workstation due to blocking or congestion in the system, then it has to be re-routed back to the ASRS. We explain this point further in Section 3.2.2.

3.2.2 Implementation and analysis

Current practice uses a centralized control strategy for routing TSUs. In this strategy, the planning level of control gives a destination for an outgoing TSU as a specific workstation on the sorter where the corresponding order is being processed. The decision to which workstation to assign is made based on the status of the system at the time of decision-making, e.g., assign the least busy route on the destined sorter. In current practice, the diverts are at a low level of control. They merely guide TSUs according to the centrally planned route.

This central control strategy gave unsatisfactory results in practice and caused imbalances in the material flow and a high rate of unprocessed TSUs returning to the ASRS. The reason is basically the incapability of the central decision-making approach to deal with the dynamically changing status of this large MHS. It is best to describe the resulting behavior of this strategy using an example. Assume that a TSU is planned for Sorter 1 via Workstation 1.1 (Figure 3.6), then the planned route is from the ASRS to Divert 1 and from Divert 1 to Divert 2a. In this case, as the TSU approaches Divert 1, assume that other TSUs have accumulated on Conveyor 1.1 due to a short absence of the operator on Workstation 1.1. In this case, the TSU proceeds to Divert 2a via Divert 1 according to its centrally planned route. However, at Divert 2a the Conveyor 1.1 is blocked and, therefore, Divert 2a routes the TSU to Conveyor 2.1. The TSU ends up at Sorter 2, which is a wrong destination and so the TSU returns back the ASRS unprocessed.

Section 2.2.2.4 proposes a generic, distributed, and dynamic routing strategy for similar types of configurations. In this section, we illustrate how this strategy not only brings more benefits in terms of simpler and more robust design, but improves the system performance as well. Using the decision-making processes of Chapter 2, we propose to make routing decisions for the MHS at hand in the following manner:

- The build planner at the *planning* level of control assigns TSUs to a destination sorter as a general destination.
- Divert controllers at the *scheduling* level of control dynamically route to workstations.

The planning level of control is implemented identically as in the generic MHS model (Section 3.1) with one small modification: an order is assigned to a sorter (with two workstations) instead of assigning an order to a single workstation. Therefore, an outgoing TSU from the ASRS has a general destination of Sorter 1 or Sorter 2, and not a specific workstation destination.

To more clearly describe the dynamic routing strategy at the scheduling level, we explain how the elements of the process in Section 2.2.2.4 are applied to this MHS. First of all, we add the dashboard logic to the controllers involved in the routing process, i.e., workstation controllers and divert controllers. Controllers use dashboards in posting information to upstream controllers to help them make decisions that balance loads on parallel routes while considering travel times and service rates. Therefore, besides posting information whether a certain destination (downstream) is accessible or not, the expected throughput times for each route downstream are posted. In more concrete terms, for this MHS as a TSU can go either to Sorter 1 or to Sorter 2, we calculate the expected throughput time per destination, where throughput time is the travel time to the relevant workstation at the destination sorter plus the processing time.

Workstation controllers on a certain sorter always post (on their dashboards) that this sorter is accessible and the other sorter is not. However, when the workstation is not in an operational mode, then both sorters are not accessible. Throughput times are calculated based on the service rate of the workstation and the number of TSUs in the pipeline. To this end, downstream controllers need information about TSUs in the pipeline from upstream controllers, in order to estimate throughput times (see Figure 3.7).

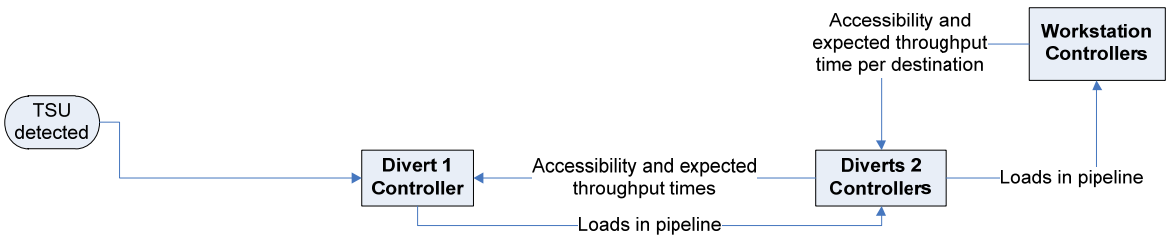


Figure 3.7. Routing decision-making.

In this context, when a disruption occurs as in the aforementioned example at the beginning of this section, e.g., route 1.1 blocked, then Divert 1 can observe which of

the downstream diverts gives access to the general sorter destination. Given no accessibility via Divert 2a and accessibility via Divert 2b, Divert 1 decides to route the incoming TSU to Divert 2b. In turn, Divert 2b routes the TSU via route 1.2, which delivers the TSU to the assigned sorter where it can be processed.

In order to compare the generic routing strategy to current practice, we develop the generic simulation model (Section 3.1.2) by adding the required modules, e.g., diverts (with their controllers), and modify the model settings according to the MHS at hand.

In this MHS, the route from the ASRS to Divert 1 accommodates 100 TSUs, while the routes from Divert 1 to Divert 2a and to Divert 2b accommodate 26 and 36 TSUs respectively, and finally the routes from secondary diverts to workstations routes accommodate from 30 to 47 TSUs depending on the source divert and destination workstation.

In such a large MHS, the first approach for determining the pipeline size (see Section 2.2.2.1) by setting it equal to the number of locations in the workstations' inbound buffer (20 TSUs each) is inadequate. A pipeline of this limited size results in an insufficient flow of TSUs in the system. Therefore, the pipeline sizes of workstations are parameters that have to be tuned (see the second approach; Section 2.2.2.1). For this MHS, our industrial partner has conducted simulation experiments to parameterize the pipeline size and found the best results when the pipeline size is set to 100 for all workstations. We also use this value in our simulation model in order to compare the performance of generic control with current practice. To this end, we implement both the generic control architecture and current practice in our simulation model. Moreover, we introduce operational failures on workstations to impose disruption in the material flow in order to test which control approach reacts better to disruptions. In this setting, the centralized control approach results in 11% of all TSUs being rerouted back to the ASRS unprocessed. However, with the generic routing approach, this figure reduces to 7%.

3.2.3 Concluding remarks

In this section, we presented an MHS with a complex routing configuration in order to apply the generic routing methodology (Section 2.2.2.4). We showed how to incorporate this routing approach in an MHS that uses other modules of the generic control architecture (Chapter 2) at the planning and scheduling levels of control. Compared to the generic MHS (Section 3.1), the MHS with a routing configuration uses the generic divert module with the methods for routing in parallel systems, as outlined in Chapter 2. This generic divert module connects to different types of resources upstream and downstream. In this section, Divert 1 connected upstream to the ASRS and downstream to other diverts, whereas the secondary diverts connected upstream to Divert 1 and downstream to workstations.

A key element in the generic routing approach is the application of dashboards as a mechanism of transmitting information to accomplish the routing operation. We illustrated the implementation of dashboards in this specific MHS. For other MHSs, the structure of the routing processes and elements used remain generic, but the information posted on dashboards or the calculations used may be adapted. As a matter

of fact, Chapter 4 presents another routing configuration in another industrial sector, i.e., baggage handling, where we implement the generic routing approach with minimal adaptation efforts.

3.3 Chapter conclusion

In this chapter, we presented applications of the planning and scheduling control modules of the generic control architecture proposed in Chapter 2. In Section 3.1, we introduced a generic MHS model of which the settings can be changed to enable it to simulate a BHS or a distribution system. In this context, the generality of control was the main aim of our analysis, but we were also interested in the implication of generic control methods on the overall system performance. In Section 3.2, we paid attention to an important scheduling process that is not covered in the generic MHS model, i.e., routing in parallel systems. To this end, we presented an MHS with a routing configuration and illustrated how the routing elements and methods are easily incorporated in the control architecture applied in Section 3.1.

We have developed the concept control architecture into a concrete architecture and validated the architecture by the implementation of all modules at the planning and scheduling levels of control. We found interesting results with regard to the extent to which we were able to maintain the generality of control. On the one hand, we were able to exploit the synergy in system elements in order to apply generic decision-making processes. On the other hand, we showed that the generic control architecture results in a good system performance.

However, the models we presented in this chapter were distinct applications that are not integrated to represent a larger MHS with a variety of processes. Therefore, in order to prove the adequacy of the control architecture further, we select a business case to make an implementation of the generic control architecture on a larger scale. Moreover, we note that our data structures should be flexible enough to adapt to unforeseen future applications. Therefore, in Chapter 4, we make a comprehensive application of the generic control architecture to a large BHS, which includes a more complex routing configuration than in Section 3.2. In addition, we introduce robots as a new type of workstations that are different from the laterals as the main workstations in BHSs and analyze how these new resources can be modeled generically.

Chapter 4

A Baggage Handling Business Case[11]

In Chapter 3, we developed the concept control architecture (Chapter 2) into a concrete control architecture and validated it. To this end, we presented applications of the control modules of the generic control architecture in different system models.

In this chapter, we provide a business case as a proof-of-concept for the applicability of the generic control architecture to a specific MHS. To this end, we build upon previous chapters to present a comprehensive application of the generic control architecture to the baggage handling system (BHS) of a terminal at a major European hub. The generic control architecture should be adaptable to the particularities of this BHS. Failing to do so means that the generic control architecture is not appropriate for the system under study and that its ability to serve as a generic architecture is questionable. This also applies if we have to make major changes to the architecture to make it applicable to the system we study.

In the BHS that we study in this chapter, there are two new elements that we did not study in earlier chapters. The first element is a screening area, which requires the implementation of the generic routing mechanism. Second, in addition to laterals as the main workstations, this BHS contains a new type of workstations, i.e., loading robots. We show how the control of robots is incorporated in the generic control architecture and how standardization can be achieved over different workstation types.

In this chapter, we also compare the generic control architecture and the current practice of this BHS. We highlight performance indicators, discuss the behavior of each of these control approaches, and comment on the architectural design and software complexity.

This chapter is organized as follows: Section 4.1 provides a more detailed description of the baggage handling process than the one provided in Chapter 1. Thereafter, Section 4.2 describes the BHS (of our business case) in detail in terms of its core task, components, challenges, and objectives. Section 4.3 presents the control architecture as applied to the BHS at hand, and briefly presents the current practice control approach of this BHS. Then, Section 4.4 presents implementation aspects. Finally, Section 4.5 ends with concluding remarks.

4.1 The baggage handling process

Most people who travel with airplanes perceive a limited part of the baggage handling process. They hand over their baggage at the check-in desks and receive it when they

[11] *This chapter is based on Haneyah et al. (2013c).*

arrive at their destinations. The background process, however, is rather complex, particularly in major hubs.

Figure 4.1 presents a high level overview of the baggage handling process. We choose to center the process around the main sorter system, where bags are sorted to corresponding flights. Incoming ULDs (unit load devices) at an airport contain either *transfer baggage* or *reclaim baggage*. All the reclaim baggage is transported to and unloaded on dedicated unloading conveyors (flow (1) in Figure 4.1). There are hardly any penalties for delayed reclaim baggage and reclaim baggage does not use complex and expensive material handling equipment. Therefore, we do not investigate this baggage flow in detail, as it is not challenging from a control or optimization point of view. The interesting flow is the transfer baggage, which has to make its way through the BHS onto another airplane. As these bags have to go through the sorter system, they mix with other bags. Therefore, delays may occur and cause bags to be late for their connecting flight(s), even if their inbound flight was on time. Normally, the details of a transfer bag become available only after the bag has been transported (2) to one of the transfer areas.

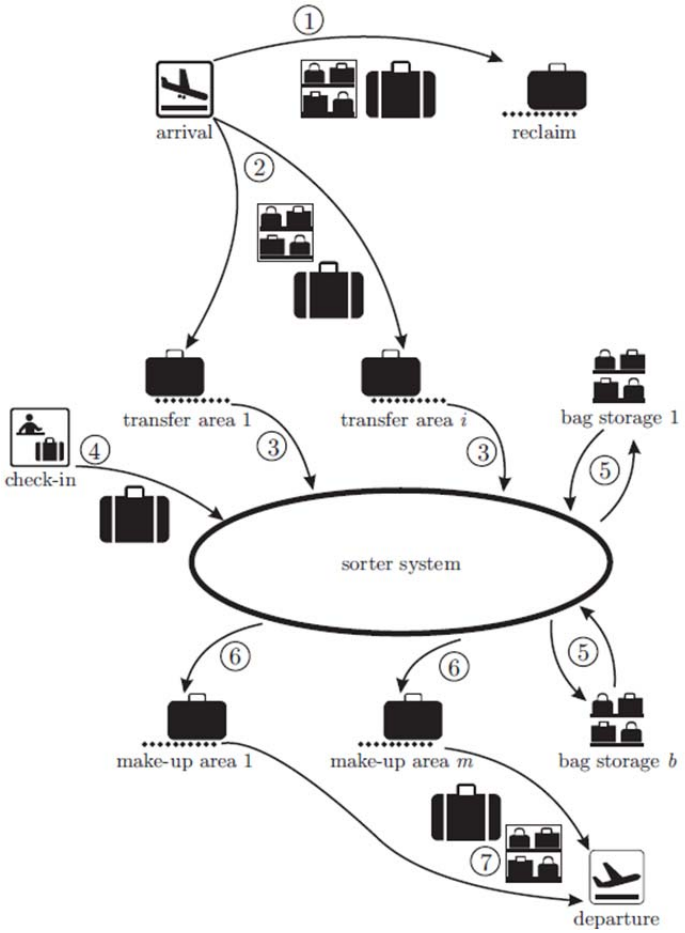


Figure 4.1. The baggage handling process.

A transfer area is defined as an area that contains multiple baggage loading conveyors, which can transport the transfer baggage onto the sorter system (3). At this point the transfer baggage merges with the baggage that has been dropped off by passengers at the check-in desks (4). The arrival process of checked-in baggage is stochastic and

difficult to influence. Early baggage is directed to one of the automatic baggage storing facilities (5), from which it is released back onto the sorter system when the build for the corresponding flight starts.

When the build for a flight starts, one or more laterals are opened to handle the baggage for this flight. The baggage arrives from the sorter system at one of the *make-up* (i.e., build) areas (6). There, workers take the baggage off the lateral and transport it towards the airplane (7), either on a ramp cart or inside a ULD. In addition to the processes described above, there are also processes for handling exceptions. A typical sorter system, for instance, has one or more laterals for bags that are too late to make it to their flights, for bags that can no longer be identified (e.g., because the label has been smudged or torn), and for bags that are extremely urgent and need manual transport to the destined airplane.

Based upon the time remaining until departure of the corresponding flight, a bag is classified as being 'cold', 'warm', or 'hot'. 'Hot' baggage is baggage that is supposed to be on an airplane that departs very soon. The exact definition differs per airport, but generally 'hot' baggage has between 45 and 20 minutes until departure. 'Warm' baggage can be transported directly towards the build area, because a lateral has already been opened to its flight. Finally, 'cold' baggage is baggage that is too early to be transported towards the build area, because no lateral has been opened to its flight yet.

4.2 The baggage handling system

In this section, we first describe the core task of a BHS in general (Section 4.2.1). Next, we focus on the BHS under study and discuss its main components (Section 4.2.2). Finally, we describe the objectives and challenges of the BHS (Section 4.2.3) which are valid for the BHS under study and for BHSs in general.

4.2.1 Core task

As mentioned in Chapter 1, the purpose of a BHS is to deliver each bag from some point A (related to its source) to some point B (related to its destination), within a specific time limit. However, the airport environment of a BHS is very dynamic and stochastic, which complicates the delivery job and generates additional challenges. Moreover, the BHS can be a complex transport network that is challenging to control. Finally, every stakeholder has his own requirements, which affect his criteria for assessing the system, e.g., the compulsory security screening operation.

4.2.2 System components

In this section, we explain the components of the BHS of a terminal in a major European airport that we take as our business case. Figure 4.2 presents a simplified material flow diagram that may help in understanding the main components of the system, which are as follows:

- **Baggage:** Baggage items that can be transported on the BHS are referred to as *conveyables*, which represent the main material flow. On the other hand, *non-conveyables* represent a very small portion of the total number of bags that

cannot be loaded on the BHS because they are too small, too large, too light, too heavy, too unstable (e.g., a ball), etc. Non-conveyables are handled with dedicated equipment. Moreover, there is often another flow, which we do not consider. This is the flow of empty totes that are used to carry bags. In general, empty totes are not bottleneck resources for the performance of BHSs. In this BHS, empty totes are used only in the storage area. The other parts of the BHS consist of classical conveyors without the use of totes.

- **Diverts and Basic Switches:** In general terms, *diverts* make a routing decision on directing the bag to one of two possible routes. The selected route leads to the bag's next process step, e.g., the main sorter or the ASRS. If the routing decision is based on the operational conditions of the BHS, e.g., congestion, then diverts make scheduling decisions. On the other hand, if the routing decision is based on the bag's attributes, e.g., the assigned destination, then diverts make local traffic control decisions. In the latter case, we call them basic switches or simply *switches*. The physical device composing a divert or a switch may be identical, but the control involved in it determines its role either as a scheduling device or merely as a local traffic control device. In this BHS, streams of bags coming from check-in desks or transfer belts mix and proceed to one of a group of 8 diverts (depending on the input point on the BHS). In 6 out of the 8 diverts, each divert is connected to 2 screening loops downstream. For the remaining 2 diverts, however, each divert is connected to 2 other diverts downstream (see Figure 4.2). Moreover, there are 2 switches connected downstream 2 of the 6 screening machines (see Figure 4.2).
- **Screening Area:** Security control occurs in this area. In this BHS, there are 4 screening loops and 2 clusters of screening machines. Each cluster consists of 3 screening machines and is accessible via two screening loops upstream. The design of the screening area with the upstream diverts is redundant in the sense that a bag has two alternative route options (decided upon by the divert) to go through one of the parallel screening loops. Next, the bag may access one of the parallel screening machines within a cluster. After having passed a screening machine, the bag proceeds on a network of conveyors to one of the downstream system areas, e.g., storage and laterals build areas. However, in this specific BHS, the screening machines do not have the same connections to downstream resources. More specifically, each cluster has only one machine giving access to both the main sorter and the storage area, a second machine giving access only to the storage area, and a third machine giving access only to the main sorter.
- **ASRS:** The main function of the ASRS is to store early bags until laterals of the corresponding flights open. In this BHS, there is a racking system to store totes with bags or without (empty locations), cranes with a double load unit device, and pickup and delivery stations at the end of each storage aisle. The main function of this system is to store cold bags until the laterals of the corresponding flights open, but it is also used for other purposes such as storing bags that missed their flights. The ASRS has 12 storage racks and 13 cranes, where each rack is accessible by two cranes.

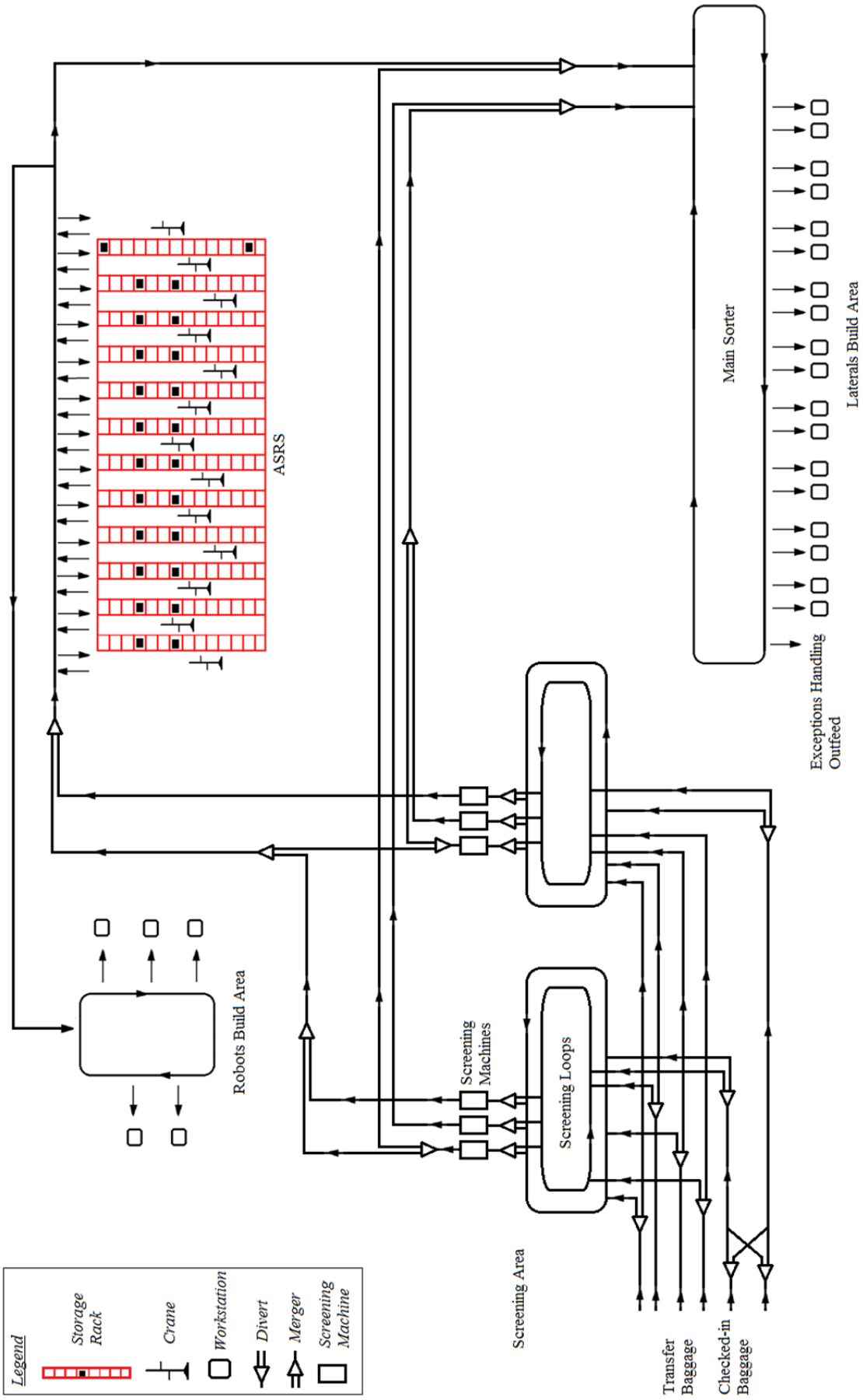


Figure 4.2. A simplified material flow diagram of the BHS.

- **Main sorter**: This is a conveyor-based sorting system, which sorts bags to the laterals that are open for the corresponding flights.
- **Laterals build area:** This is the main build area, including the sorter(s) and the laterals, where workers unload bags from the BHS to prepare them for loading on destined airplanes, often using ULDs. One or more laterals are assigned to a single flight, and will be opened a couple of hours before the scheduled time of departure. In this BHS, there is one sorter with 18 build laterals and one exception handling lateral that collects bags that missed their flights.
- **Robots build area**: This is not a common resource in conventional BHSs, but exists in the BHS under study. Robots can be used to build ULDs for different flights, where each flight may use robots starting an hour before lateral open time (for the flight) and closing 15 minutes before lateral open time. Section 4.3 discusses the build operation in more detail and differentiates between the robots build area and the laterals build area. In this BHS, there are 5 robots, 2 of which are fully automated, while the other ones are semi-automated. The semi-automated robots can handle bags with a higher rate, because a number of operators work on them.
- **Mergers:** Mergers are devices that combine two incoming flows of baggage from two different sources into one outgoing flow.
- **Conveyors**: System areas and resources are connected by a complex network of conveyors, which we take into account in our control methods, but do not model explicitly as they are at a low level of control that mainly executes decisions that are taken at higher levels of control.

4.2.3 Objectives and challenges

As stated in Chapter 1, the main KPI for BHSs is the irregularity rate. In large BHSs, such as the BHS at hand, it is common to have several resources that can execute the same job, e.g., parallel screening machines and redundant transport systems. Hence, there are different possible routes to realize the transport operation from the entry point of the BHS until the destined exit point. The logistic control function should use the resources of the BHS as efficiently as possible to minimize the bag's flow time.

In this thesis, our objective is to achieve generic control of MHSs in different industrial sectors. Moreover, we are interested in generic control within each specific sector. In the latter context, the control of the BHS needs to use generic methods, in contrast with current approaches where the control of BHSs is customized per project and not standardized even within a single project. For example, in the BHS there are workstations that work on the build operation, being robots or manned-laterals. However, in current practice, there is a different control approach for each workstation type. Section 4.3 discusses the generic control of these workstations. Moreover, current approaches often use the static shortest path in routing decision-making, i.e., the shortest path in-time from A to B given an empty system. However, in reality the static shortest path may not result in the smallest travel time for several reasons, e.g., congestion and failing workstations. In Section 4.3, we show how to dynamically determine the best route according to the generic routing approach proposed in Section 2.2.2.4.

4.3 The control architecture applied to the BHS

Chapter 3 presented an application of the generic control architecture to a generic MHS model (with an ASRS, a main sorter, and a laterals build area). In this chapter, we develop a more comprehensive application of the control architecture to the BHS. In this application, we extend the BHS of the generic MHS model (Section 3.1) to include two new areas, i.e., the screening area and the robots build area.

In this section, we present the components of the control architecture (Section 4.3.1) and the decision-making processes involved (Section 4.3.2) in this BHS. Figure 4.3 shows the control structure with the main control units and decision-making processes.

4.3.1 The control architecture of the BHS

In this section, we present the controllers involved in this BHS at the higher levels of control (planning and scheduling) and their main tasks.

- *Build planner:* This controller is responsible for the build areas, i.e., it coordinates the build workstations being manned-laterals or robots. In baggage handling, this means the controller plans build operations for flights, i.e., it activates the build of certain baggage groups on workstations and communicates with the storage planner to request the release of bags from the ASRS to the right build point(s).
- *Storage planner:* This controller is responsible for the storage area, i.e., the ASRS consisting of cranes and storage racks. The storage planner assigns retrieval tasks to subordinate crane controllers based on information from the build planner. Moreover, upon request by the build planner, the storage planner investigates the possibility of releasing baggage groups to build ULDs for certain flights (see Section 4.3.2).
- *Workstation controller:* In the modeled BHS, we have two types of workstations, i.e., laterals and robots. Flight build times are planned beforehand on laterals and so laterals are reserved for certain flights during some time frame. On the contrary, robots are flexible workstations that can be used to fill a ULD for any candidate flight. A candidate flight at a certain moment in time is a flight that is allowed to use robots at this moment in time and that has a sufficient number of bags in the ASRS to fill a ULD. Both workstation types trigger the build planner to release work. However, a lateral workstation triggers the build planner whenever the planned time to build a flight commences, while a robot triggers the build planner whenever it is about to finish the build of a certain ULD and can start receiving bags to build another one.
- *Crane controller:* At the scheduling level of control, the main task of the crane controller is to schedule the retrieval tasks (timing and sequencing). This scheduling process considers the urgency of retrieval tasks and the pipeline of destined workstation(s).

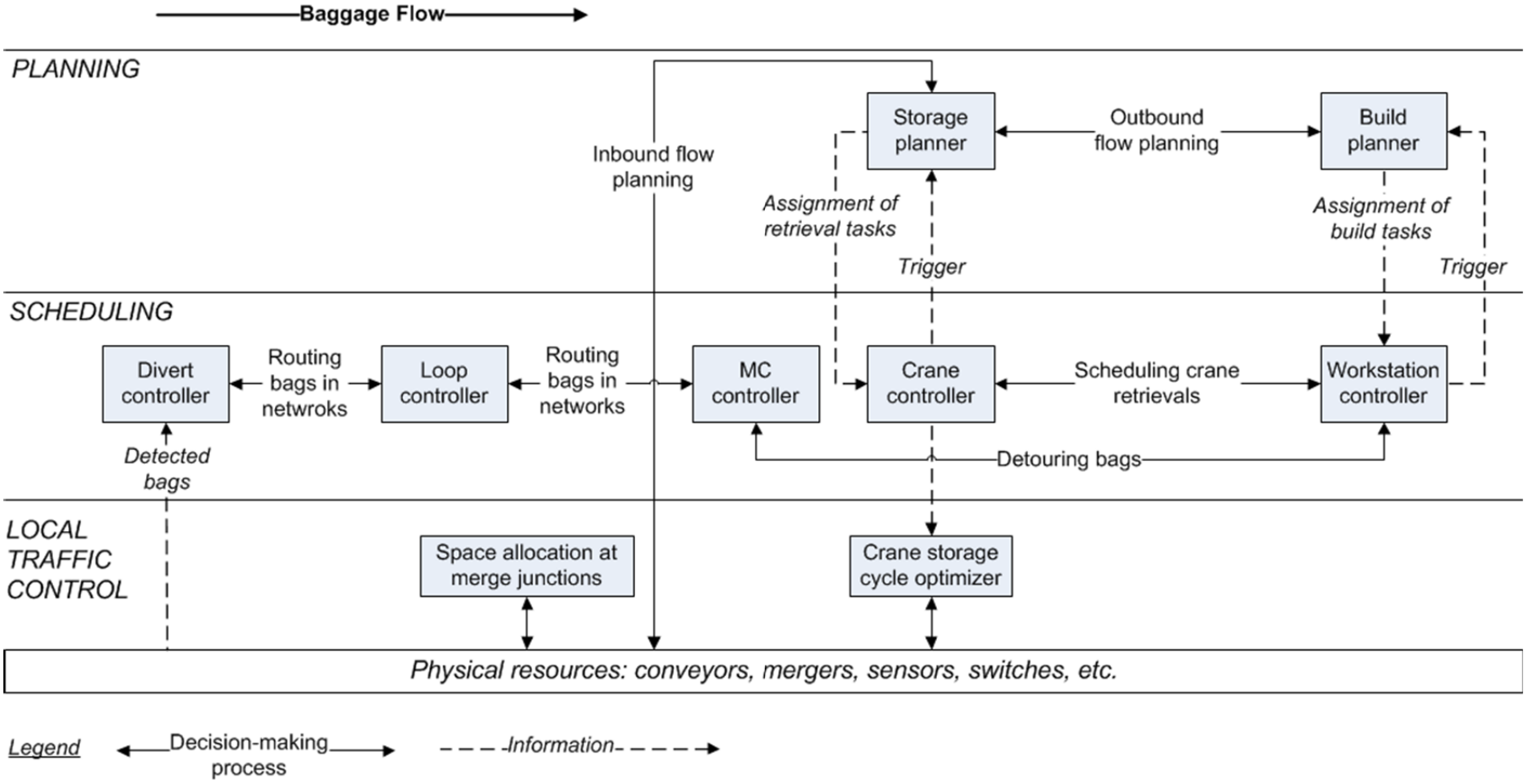


Figure 4.3. Control structure.

- *Machine cluster (MC) controller:* A machine cluster controller is responsible for monitoring a group of parallel screening machines that are connected to the same resources upstream, and for posting information to upstream controllers about estimated throughput times. To this end, information about bags in the pipeline from upstream controllers is required.
- *Loop controller:* Loop controllers participate in the routing process, which is on the scheduling level of control, by transmitting information from downstream machine clusters to upstream diverts and the other way around. In addition, loop controllers post information about space utilization of the loop (see Section 4.3.2).
- *Divert controller:* This controller makes scheduling decisions on diverting bags to one of several downstream systems in the screening area. To make this decision, it uses information transmitted from downstream controllers.

We stress that the planning controllers are unique and aggregate, whereas scheduling controllers are duplicated for every resource. Section 4.3.2 explains the interfaces between the controllers mentioned in this section and how they communicate to perform the main decision-making processes in the control architecture as implemented in this BHS.

4.3.2 Decision-making processes and communication

In this section, we list the decisions taken at each level of control and communication that takes place between different controllers. In the context of the BHS, we emphasize the key descriptions and applications presented in previous chapters and illustrate the new elements or the adaptations that are required to control this BHS. Moreover, we briefly indicate how decision-making processes are controlled in current practice.

4.3.2.1 Planning level

The two main problems at the planning level of control in a BHS are to plan the inflow of bags to the ASRS and to plan the outflow of bags from the ASRS towards build areas (see Chapter 2). In this section, we describe these planning processes in the context of the generic control architecture as applied to the BHS.

Inbound flow planning to the ASRS

In current practice, an incoming bag (which requires storage) triggers a higher level of control. The higher level of control responds with a destination rack and a crane to perform the storage operation.

In the generic control architecture, the bag is announced to the storage planner, which responds with a destination rack and crane to perform the storage operation. In this BHS, for both control approaches, the storage rack with the smallest number of bags is selected, provided that there is at least one active crane on the rack.

Outbound flow planning from the ASRS

In current practice, cranes trigger a higher level of control that they are ready to perform a retrieval cycle. At the higher level of control, two different approaches are employed to release baggage from the ASRS:

- Retrievals for robots are released on an individual basis considering the pipeline of the destined robot and the sequence of bags required in the ULD. Retrievals for robots have priority over retrievals for laterals.
- Retrievals for laterals are released in baggage groups (e.g., bags of a certain flight) considering a limit for the ASRS on the rate of retrievals for a certain baggage group. The baggage groups are classified at a high level of control in different priority classes according to the number of bags in the ASRS and to the planned retrievals' finish times of these bags. Priority classes of the baggage groups and the limits on release rates are dynamic and updated continuously as they depend on the elapsed time and on the number of bags in the ASRS. The high level of control assigns the bags that the triggering crane has to retrieve in its next cycle. In these assignments, prioritization rules within a certain priority class are also considered.

The generic control architecture provides a generic release approach that is based on standardizing the two types of workstations (laterals and robots). To achieve this standardization, we propose: first, setting pipeline size limits for all workstations and second, imposing due times on all crane retrievals. Figure 4.4 provides an overview of the outbound planning process as applied to the BHS.

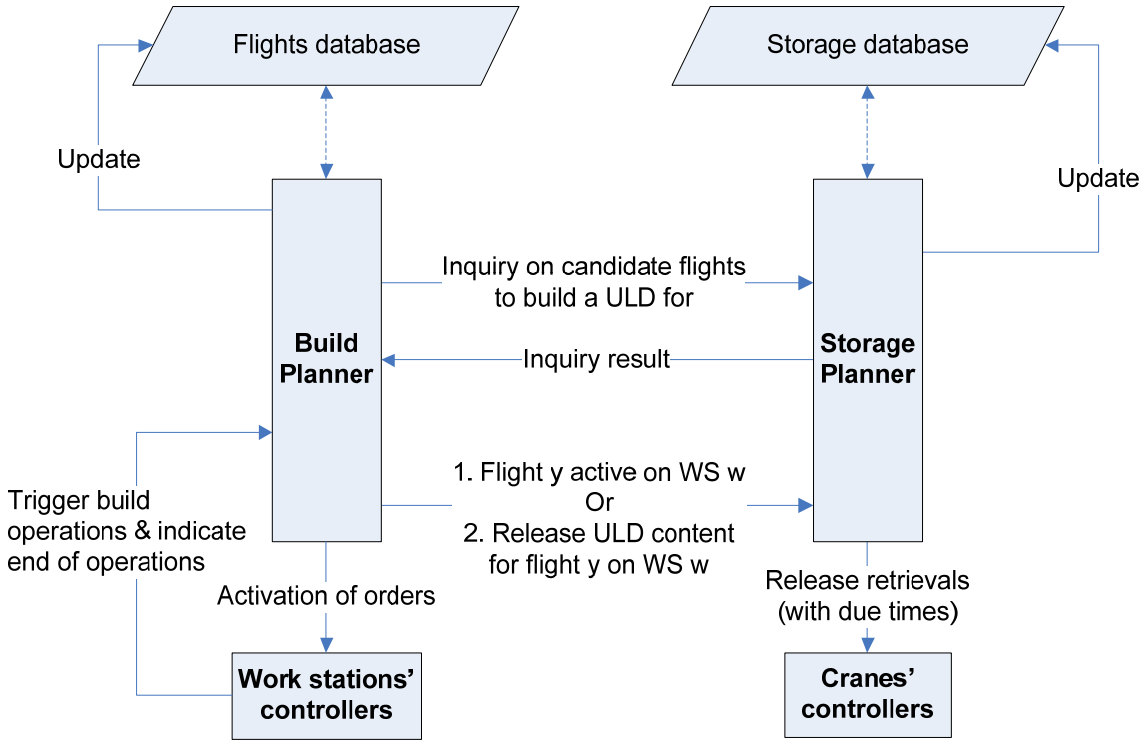


Figure 4.4. Communications for outbound planning.

We now describe the outbound planning process in more detail and show how the control is standardized over the workstations of the system.

Baggage is released from the ASRS in groups, where a baggage group can be as large as all bags belonging to a certain flight or a subset of these bags defined by the storage planner. There are two main sub-processes in outbound flow planning:

- *Stock reservation*: this is a generic planning process that we apply in Chapter 3 for the distribution sector to assign product totes to orders. In baggage handling, this process is often not needed because each bag entering the system via check-in desks or as transfer baggage is already assigned to a specific flight. However, the extension to the robots build area requires the use of some functionality of this process, because the storage planner has to make a selection of bags from a possibly larger set to be assigned to a certain ULD. In other words, when a robot-workstation announces its availability to build a ULD, the build planner inquires the storage planner about candidate flights to build a ULD for. Candidate flights should have enough content of bags in the ASRS to fill a ULD and should be within the time allowed to build ULDs on robots (see Section 4.2). The storage planner responds to the build planner with available options. The build planner may then request to release a baggage group (ULD content of bags to be selected by the storage planner) for a certain candidate flight (e.g., the flight with minimum time remaining until departure) and assign this baggage group to the triggering robot.
- *Order release:* workstations trigger the build planner to activate the build of baggage groups, based on work progress for robot-workstations (in this case a baggage group consists of the content of a ULD) or according to planned build times for lateral-workstations (in this case a baggage group consists of all bags for the flight concerned). As soon as a baggage group is active on a workstation, bags belonging to this baggage group have to be released from the ASRS. Therefore, the build planner informs the storage planner that a certain baggage group is active. In turn, the storage planner dynamically assigns relevant bags to candidate cranes as retrieval tasks, since each storage location is accessible by two cranes. If both cranes are active, then the storage planner assigns the retrieval to the crane having the smallest workload. Moreover, the storage planner sets due times for retrieval tasks. The due time for bags going to robot-workstations is a parameter that we use (see Section 4.4) to indicate the end of the time interval allowed to build a ULD, whereas the due time for bags going to a lateral-workstation is the planned end time of the flight build. From this point on, cranes are responsible for executing and sequencing these tasks at the scheduling level of control.

4.3.2.2 Scheduling level

At the scheduling level, we apply the following generic scheduling processes (see Chapter 2): scheduling crane retrievals, routing TSUs in networks, and routing arriving TSUs. We use the latter to make decisions on detouring bags as described later in this section.

Scheduling crane retrievals

Given a set of retrieval tasks, crane controllers schedule these tasks based on their due times and the pipelines of destination workstations (see Figure 4.5), as described in Chapter 2 and applied in Chapter 3. However, in Chapter 3 we only had lateral-workstations in baggage handling, but in this BHS there are also robot-workstations. For robot-workstations, the flow has to be strictly controlled, because bags are to be

handled according to a predetermined sequence. Therefore, recirculation of bags due to blocked entry to a robot is prohibited. In this context, we apply the first approach for determining the pipeline size (Section 2.2.2.1), in which the pipeline size is equal to the number of locations in the inbound buffer of the robot-workstation. In this way, if any problem occurs in the workstation, then all bags in transport can be accommodated in the inbound buffer with a preserved sequence. On the other hand, we determine the pipeline sizes of lateral-workstations according to the second approach as applied in Section 3.1.

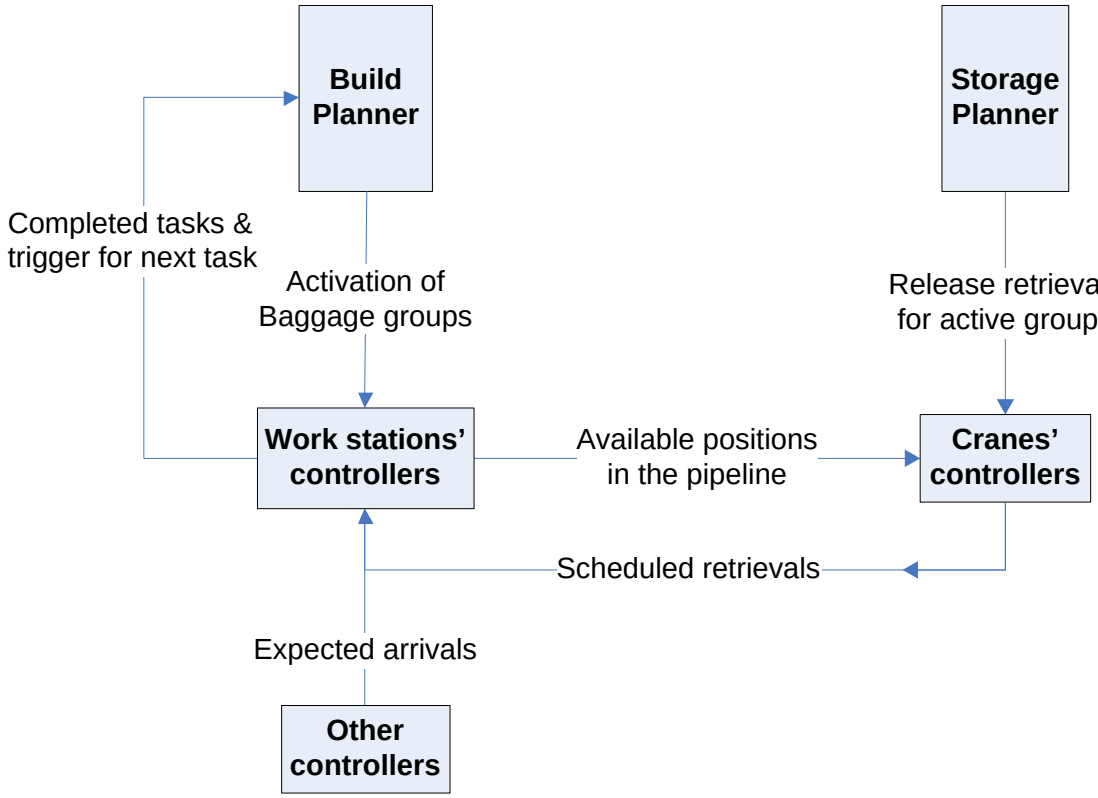


Figure 4.5. Communications for scheduling crane retrievals.

We highlight that for the robot-workstations, we use the control procedures as applied to the workstations in the distribution system of Section 3.1 in view of the high synergy between these two workstations. This reuse of control procedures emphasizes the generality of the control architecture. The following are the main points of similarity between the robot-workstations and the distribution-workstations:

- Workstations receive a group of TSUs in a predetermined sequence and TSUs of different groups are not allowed to mix in transport, at least when destined for the same workstation. These operational conditions govern the pipeline size determination approach.
- Workstations trigger the build planner for new task assignments according to the progress of work and not according to time schedules (see triggering and releasing orders: Section 3.1.2.3).
- Workstations use the parameter 'maximum number of orders simultaneously active on a workstation' (see Section 3.1.2.1). However, this parameter is 1 for

robot-workstations, which handle 1 ULD at a time (note that this parameter was 6 for the distribution-workstations considered in Section 3.1).

Routing bags in networks

In Chapter 2, we outlined an approach for routing TSUs in networks, where divert controllers make the routing decision based on the state of the system downstream. In Section 3.2, we discussed an application of this routing approach to a routing configuration in the distribution sector. In this section, we show another application, namely to the screening area of the BHS at hand. In the screening area, screening machines (see Figure 4.2) are available at alternative systems. In such configurations, a divert controller has to decide to which system to divert an incoming bag. Contrary to current practice where static shortest path algorithms are often implemented, we apply the generic and dynamic routing approach. We note that we model 8 diverts, each connected to two resources downstream. More precisely, for 6 diverts, each is connected to 2 screening loops downstream, while for the remaining 2 diverts, each is connected to 2 other diverts downstream. Each screening loop is connected to one cluster of screening machines downstream (see Figure 4.2).

We aim to balance the load on parallel systems and react to machine failures, i.e., fewer bags should go to the system having a lower capacity (due to, e.g., failed machines). For the distribution system modeled in Section 3.2, we focus on expected throughput times of the systems. In the BHS however, as the bag can go either to the laterals build area or to the storage area (from the screening area) as general destinations, we calculate the expected throughput time per destination. Note that in Section 3.2 the general destinations were Sorter 1 or Sorter 2. To this end, machine cluster controllers post expected throughput times. In turn, upstream controllers use this information to make routing decisions. Upstream controllers also have to provide information about bags in the pipeline for downstream controllers to estimate throughput times.

In this large BHS, the decision to divert a bag to system A or to system B is impractical to take for each bag separately due to the high rate of bags passing the divert at a high speed. It may cause excessive switching of diverts (which is undesirable for the equipment). Therefore, the divert is positioned to one of the downstream systems until the difference in throughput times between downstream systems exceeds a certain threshold. Then, we need to react to the imbalance and so the divert switches position. As long as throughput times are balanced, we check whether space occupation on downstream loops is balanced in the same manner. We use the dashboard logic to post status updates to upstream controllers. Each component (machine cluster, loop, or divert) has a dashboard that posts accessible destinations downstream, expected throughput times, and space occupation on loop(s) downstream (see Figure 4.6 for an example). Upstream flow is always blocked when the system has completely absorbed the allowed capacity.

With regard to the generality of the routing approach, we note that if the threshold values are set to 0, then the control mechanism is reduced to the mechanism that we use in the distribution system of Section 3.2. Therefore, in this section we actually provide a more general routing approach that can be applied to the distribution system

we modeled in Section 3.2. Moreover, we again show that the standardized controllers and communication interfaces allow the application of the same control logic to different system layouts by merely defining connected controllers upstream and downstream for each component.

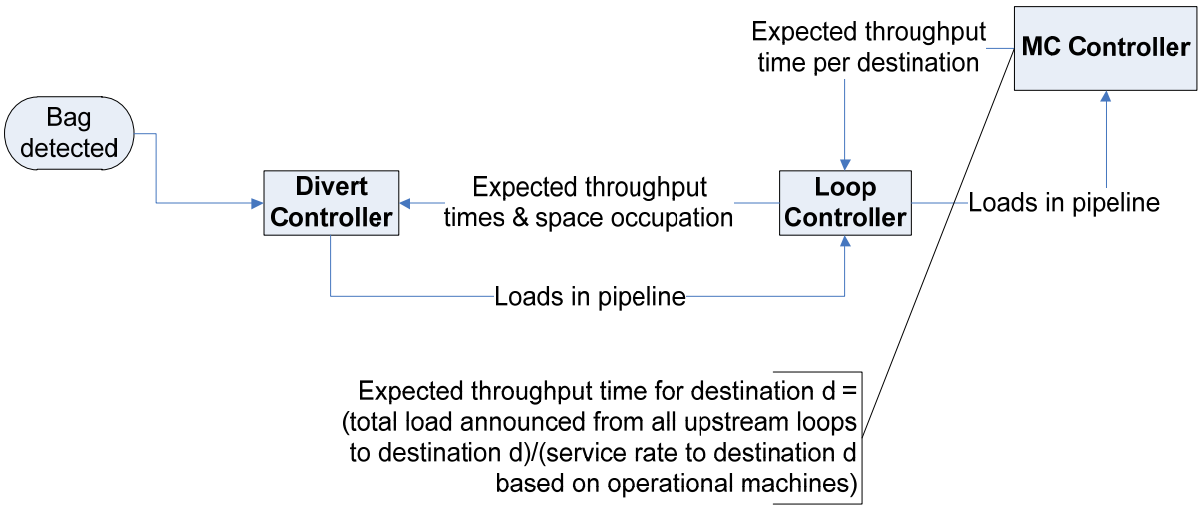


Figure 4.6. Communications for dynamic routing.

Detouring bags

As mentioned in Section 4.2.2, one screening machine per cluster has connections to both the main sorter and to the ASRS. Therefore, outgoing bags can be routed either to the sorter system or to the ASRS via the switches available downstream these screening machines (see Section 4.2.2).

In current practice, switches route the bag based on the status of the corresponding flight. In more concrete terms, if the corresponding flight is building then the switch directs the bag to the main sorter. Otherwise, the switch directs the bag to the ASRS. In this approach, there are no scheduling tasks involved in the switch controller.

However, in the generic control architecture, we analyze the option of upgrading these switch controllers to divert controllers by incorporating decision-making rules at the scheduling level (e.g., based on working conditions). In this approach, bags are routed to the ASRS when the build for the corresponding flight is not open yet. Moreover, if the build is open *and* the pipeline(s) of the destined workstation(s) is (are) not full then the bags are routed to the main sorter (we route to the least occupied pipeline when more than one lateral is available). However, if the pipeline(s) is (are) full, then, in order to maintain a controlled flow on the main sorter, it may be beneficial to route bags to the ASRS and delegate the scheduling task to crane controllers there. We refer to the latter option as the *detour* option since it causes longer handling times by routing bags through additional system areas. The detour option should not be used for urgent bags as it may cause them to miss their flight. In this BHS, we can detour bags only if they have at least 30 minutes until departure. Otherwise, we route them directly to the laterals although they are busy. In this case, recirculation on the sorter is a safer option.

4.3.2.3 Local traffic control

In addition to the planning and scheduling decision-making processes, we apply the local traffic control processes (see Chapter 2) in an aggregate manner as they do not affect the overall architecture and merely execute scheduling decisions. The main local traffic control processes are: first, space allocation at merge junctions, e.g., allocating free spaces on the main conveyor in front of the ASRS to bags waiting in the outbound buffers of cranes. Second, the crane storage cycle and in-aisle travel optimizer, which concerns the determination of travel sequences for a crane within an aisle, e.g., to execute a storage cycle to store incoming bags.

4.4 Implementation

In order to test the control architecture, we extend the simulation model used in Chapter 3, which includes the main building blocks of the BHS under study and of the control architecture. In this section, we describe the experimental setup (Section 4.4.1), the model parameterization (Section 4.4.2), and finally we present general results (Section 4.4.3).

4.4.1 Experimental setup

We configure the settings of the simulation model to represent the BHS at hand. In addition, there are control parameters, which need to be tuned for this BHS. These parameters are:

- The threshold values we use in parallel screening systems to determine the allowable difference in expected throughput times (or space occupation on screening loops). When these threshold values are exceeded, the divert switches the baggage flow to the other system downstream.
- The time allowance in the pipeline size expression according to the second approach (Section 2.2.2.1).
- Due times on retrievals to robot-workstations.

We use data sets regarding real-life flight schedules and baggage arrivals for the BHS of which the physical components are explained in Section 4.2.2. The operational scenario covers a complete day of operation in each simulation run. We include common screening machine failures occurring in practice during normal operation with exponential distribution for the time to failure (mean = 6 hours) and an exponential distribution for repair time (mean = 10 minutes). Each simulation run includes: 61 flights where each flight is scheduled to be built on two laterals for 75 minutes, ending 15 minutes before the scheduled time of departure. We set the number of operators per lateral to 2, with a handling capacity of 120 bags per hour per operator. Moreover, each flight may use robots for 45 minutes, starting 1 hour before lateral build time starts. Automated robots and semi-automated robots handle bags at a capacity of 200 and 350 bags per hour, respectively. Finally, the number of bags modeled per simulation run is 25,198, which consist of 7983 transit bags and 17215 checked-in bags.

4.4.2 Model Parameterization

In this section, we propose suitable values for the main control parameters in the model.

4.4.2.1 Pipeline allowance parameter

We tested the generic routing logic in parallel screening systems and the detour option (Section 4.3.2.2). To this end, we use a BHS of a limited scope, which includes the screening area and the areas in direct connection downstream: the main sorter and the ASRS. We do not include the robots build area, because it is not in direct connection with the screening area where routing takes place. We test the detour option (see Section 4.3.2.2) versus always sending bags to laterals when the build time of the flight is open (even when the pipelines are full).

Routing parameters (see Section 4.3.2.2) are selected according to desirable values in practice. We use a throughput time threshold of 1 minute, which means that a divert switches position only if switching leads to at least 1 minute savings in throughput time downstream. Moreover, we keep track of space utilization on (screening) loops. Once the space utilization on one of the loops exceeds 90%, and as long as throughput times are balanced, we start balancing for space utilization. We set the space utilization threshold to 20%, which means that a divert switches position only if switching leads an incoming bag to the loop which is at least 20% less occupied than the other accessible loop downstream.

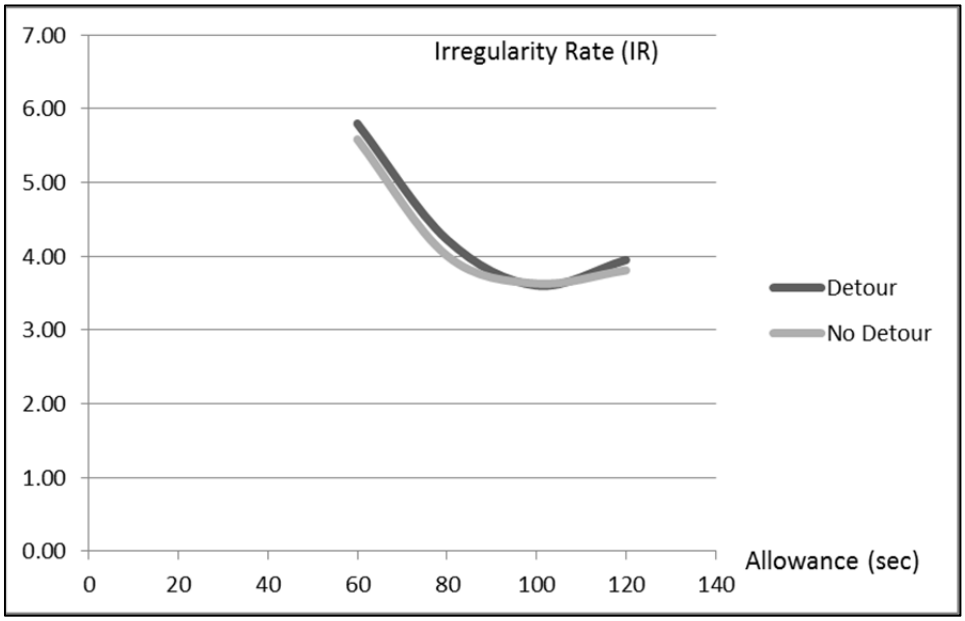


Figure 4.7. Time allowance versus irregularity rate with robots disabled.

In order to determine the pipeline size, we conduct several experiments to configure the time allowance parameter (see Section 4.4.1). Figure 4.7 shows the effect of different time allowance values on the irregularity rate, which is the main KPI. We conduct the experiments without the detour option and with the detour option, in which a bag may be detoured when pipelines are full provided that the bag is not urgent, i.e., it has at least 30 minutes until departure for the BHS at hand. The best

value for the time allowance is found to be 100 seconds for both options (with detour and without detour). We observe that, in this BHS, the detour option does not have a big effect on the irregularity rate, since we are able to get comparable results without this option. Moreover, the detour option is seldom used; on average 0.4% of all bags were detoured. There are two causes for this result: first, the possibility of the detour option is limited to only 2 out of the 6 screening machines in the screening area due to the design of this particular BHS (Section 4.2.2). Second, the BHS under study is a large system where a bag is allowed to be detoured only if it has at least 30 minutes until departure (see Section 4.3.2.2). Therefore, the number of bags that can be detoured is limited.

4.4.2.2 Setting due times

In Section 4.3.2, we stated that in order to achieve standardization among different types of workstations, it is necessary to impose due times on all retrieval tasks assigned to cranes. In this way, cranes can have a standard approach for retrievals and do not have to distinguish retrievals going to robots from those going to laterals as in current practice.

Due times on retrievals going to laterals are straightforward (Section 4.3.2) because they are based on planned build times on laterals. Moreover, current practice implements an approach that is based on due times as well. On the other hand, we have to find the right parameters for due times on retrievals going to robots. For the robots case, due times is a new concept that is not applied in current practice (see Section 4.3.2).

To this end, we use the simulation model to find the right parameters (using a fixed pipeline allowance time for retrievals for lateral-workstations). As we include the robots build area, the system capacity becomes high enough to handle all bags in time (i.e., to have an irregularity rate equal to 0). No capacity problems occur because we do not model major disruptions such as plane delays or severe equipment failures. However, we need to have irregularity rates in order to make comparisons between different due time approaches. Therefore, we impose a restriction that all bags have to arrive at their destined lateral before the lateral close time. Otherwise, we consider them missed. However, in reality there is still 15 minutes between lateral close time and plane departure within which a bag can receive special handling to make it to its flight.

We experiment with several options to set due times on retrievals to robots and examine the effect of each option on the irregularity rate (IR) as shown in Table 4.1. We also measure the number of ULDs built by all robots during a complete day of operation. The first option in Table 4.1 is based on the fact that each flight has 45 minutes allowed to use robots, starting 1 hour before laterals open (ending 15 minutes before laterals open). Therefore, we set due times for retrievals of each flight to be the end time of this 45-minutes interval. However, this option leads to an unacceptable performance. Following this option, a retrieval to a certain flight may have 45 minutes until its due time, when it is released early. On the other hand, another retrieval to the same flight may have few minutes until its due time, when it is released towards the end of the time allowed for the flight to use robots. A resulting behavior is that crane

controllers delay the retrievals to robots as they had initially long time until they are due. As a result, robots do not trigger for more work because the early assigned ULD builds were not processed and robots are still waiting retrievals from cranes. The 5 robots build only 78 ULDs over the whole day of operation, which is a severe underutilization. Moreover, when retrievals for robots are due soon, cranes start working on them and cause retrievals to other flights building on laterals to be late. The aforementioned system behavior causes the irregularity rate to be too high.

Consequently, we have to impose a ULD-specific due time on the bags of each ULD instead of the general 'robots build close time' as the due time for all ULDs of a certain flight. Therefore, we propose a time interval allowed to build a ULD, which is based on dividing the number of bags planned for the ULD by the handling capacity of the assigned robot. In this option, we plan for lower than 100% capacity on robots in order to take into account issues such as transport times, waiting on junctions, blockings, and retrieval times by cranes. Therefore, we actually divide the number of bags planned for the ULD by 60% of the robot capacity (see Option 2 in Table 4.1). Compared to Option 1, the resulting performance improves dramatically, where on average less than 1 bag is not properly delivered per 10,000 bags and the number of ULDs built is 475. A similar option to set a due time for a ULD, is to define the ULD due time based on 100% utilization of robots and then add a time allowance to this value (see Option 3 in Table 4.1). In this option, we divide the number of bags planned for a ULD by the robot capacity and then add a time allowance. With this option, the irregularity rate is the same as in Option 2, and the number of ULDs built is 479.

Option	Due time criteria	Description	IR	ULDs built
1	Robots build close time *Flight-based*	The end of the time interval where usage of robots is permitted per flight.	27.33	78
2	Capacity-based, with slack *ULD-based*	(NrBagsPlanned/[slack*RobotsCap]) *best at slack=0,6*	0.095	475
3	Capacity-based with allowance *ULD-based*	(NrBagsPlanned/RobotsCap+allowance) *best at allowance = 5 min*	0.095	479
4	Experimental *ULD-based*	Different values for ULD build time of semi-auto and auto robots *(best at 15 min)*	0.071	481

Table 4.1. System performance under different due time settings for retrievals to robots.

As a matter of fact, Option 4, in which we use experimental build times for ULDs, gives the best results, with 15 minutes allowed to build a ULD for any robot. So we use this as a parameter in our further experiments as follows: when a retrieval to a robot is assigned to a crane, the due time is always the time of release plus 15 minutes. It may be surprising that we achieve best results when both semi-automated robots and automated robots are given the same time to build a ULD, although semi-automated robots have higher capacity in handling bags (Section 4.2.2). However, giving both the same time to build a ULD at some point in time means that the due times for all retrievals released at this point in time are the same. Therefore, cranes work on retrieving bags for all 5 robots in the system, but if the allowed due time intervals for

semi-automated robots were even a few minutes smaller, the cranes would serve these robots and delay serving the automated robots. Consequently, at the system level, bags are handled at a lower rate causing the overall performance to decline.

4.4.3 General results

We test our control architecture on the large scale BHS, using the parameters as tuned in Section 4.4.2. In this BHS, we include the robots build area and implement a common practice of 2 operators working on each lateral. Moreover, we allow special handling for bags that miss lateral close time, but can still be delivered to the plane before departure by an operator for example. These bags should arrive at the plane in the 15 minutes time interval between lateral close time and the scheduled time of departure for the corresponding flight.

In addition, we compare the performance of the generic control architecture to current practice (see Section 4.3.2) for the same input data. Given the large BHS with full capacity, it is possible to properly deliver all bags under generic control as well as under current practice approaches. In this case, we analyze other performance indicators (PIs) that are of interest. These PIs are as follows (see Table 4.2):

- *The average and maximum measures of traffic delay:* these measures concern the traffic delay (waiting time) before an outgoing bag from the ASRS enters the main sorter.
- *Percent recirculations:* this measures concerns bags that arrive at full inbound buffers of their destined laterals and thus have to recirculate on the main sorter for a second delivery attempt.
- *Percent detours:* this measure reflects the proportion of bags that were detoured (see Section 4.3.2.2).
- *Number of ULDs built:* this is a measure of the total number of ULDs built during the complete day of operation.
- *Percent special handling bags:* this is a measure of the proportion of bags which arrive after the lateral has closed but before the departure time of the plane, and thus could still be loaded on the plane.

We observe that, due to the pull concept, generic control performs better in minimizing re-circulations on the main sorter, which affects traffic delays as well. However, current practice compensates for less output on the sorter by better utilization of robots (more ULDs are built). This may be justified because retrievals for robots always get priority in current practice, while in generic control, there is no strict distinction between retrievals for robots and retrievals for laterals. With regard to the percentage of bags receiving special handling due to missing the lateral close time, the performance is comparable to current practice.

To test the effect of the number of operators per lateral, we find that having 1 operator per lateral instead of 2 does not increase the irregularity rate, so the BHS would still perform well even with a lower number of operators than usual, which is desirable in practice. On the other hand, if the robots are disabled, all other settings being the same, then the irregularity rate increases to 3.60, which is unacceptable. However, we can compensate for disabling robots by increasing the number of operators per lateral to 3.

In this case, the irregularity rate is the same as with 2 operators per lateral and robots being enabled.

Performance Indicators	Generic control (without detours)	Generic control (with detours)	Current Practice
Average Traffic Delay	1.05 sec	0.90 sec	2.20 sec
Maximum Traffic Delay	79.58 sec	64.06 sec	175.43 sec
Percent Recirculations (of bags that entered the sorter)	15.93%	15.44%	53.94%
Percent Detours (of all bags)	0.0%	0.12%	0.0%
Number of ULDs built	480	481	490
Percent special handling bags (of all bags)	0.017%	0.016%	0.012%

Table 4.2. Performance indicators for generic control versus current practice.

4.5 Chapter conclusion

In this chapter, we provided a proof-of-concept for the applicability of the generic control architecture for MHSs in different sectors. To this end, we have extended the applications we made in Chapter 3 to present a more comprehensive application to a business case in the baggage handling industrial sector. In this business case, the BHS consists of several areas: a laterals build area, a robots build area, a storage area, and a screening area. Moreover, a variety of decision-making processes, at different levels of control, are implemented in this BHS.

One of the advantages proposed by this study is to model workstations, being laterals or robots, in a generic manner. This resulted in a simpler control software for the order release and retrieval processes. In current practice, one approach is implemented in the storage area to retrieve bags for robots, while another approach is implemented if the destination is one of the laterals. Moreover, we implemented a dynamic routing strategy that uses the dashboard logic to make routing decisions and to react to breakdowns and congestion. These control methods have a modular and generic structure, which allows them to be implemented in different BHSs and different MHSs in other industrial sectors.

In our application of the generic control architecture to the BHS at hand, we highlight some points that support the concept of generic control: first, we used the same planning level as in the generic MHS model (Section 3.1), but on an extended system base. Similarly, we used the same storage system with identical controllers, but introduced new connections to the upstream screening area. Moreover, we introduced a new type of workstations (robots) to the basic material flow model and found them analogous to the workstations in the distribution sector. Finally, to model the layout of the BHS at hand, we could easily modify transfer times, capacities, systems

connections, etc., since they are adjustable parameters in both the simulation model and the control architecture.

Current practice approaches at the planning level are customized, complex, and computationally intensive. Alternatively, the generic control architecture identifies the decision-making processes at the right level of control, and handles layout-specific details by configurable parameters. As a result, the architecture is scalable and tunable to different system layouts and designs. Moreover, the architecture allows for a much faster implementation and is both flexible and more robust, still without compromising the overall performance. Finally, we stress that the comparisons we made are based on normal operational conditions. When more severe and unexpected disturbances in the material flow occur, we expect generic control to outperform current practice as generic control reacts directly to problems in material flow and takes actions to avoid possible congestions and imbalances.

As the generic control architecture is designed, implemented, and confronted with more challenges in a business case, we extend our analysis to a scheduling problem, which is influential for the operation of MHSs but that is not part of the control architecture (as a software component). This problem is scheduling inbound containers to load MHSs that use sorters as the main element. Chapter 5 analyzes this scheduling problem.

Chapter 5

Improving The Performance Of Sorter Systems By Scheduling Inbound Containers[12]

In Chapter 4, we applied the generic control architecture to a BHS (baggage handling system) and showed how generic decision-making processes are applied at the different levels of control. With regard to the material flow, we modeled the arriving bags (so far) as they appear at diverts. However, what system-users do before a TSU is placed on an infeed or after a TSU has been retrieved from an outfeed was not within our scope of analysis (Scope 1; see Section 1.2.1).

In this chapter, we study scheduling algorithms that lead to better use of sorter systems. Such algorithms allow system-users to have better systems' performance without installing additional equipment. More specifically, this chapter investigates the inbound containers scheduling problem (see Section 2.2.2.2) for automated sorter systems in two different industrial sectors: parcel & postal sorting and baggage handling. For the distribution sector, this scheduling problem is not relevant due to reasons discussed in Section 2.2.2.2.

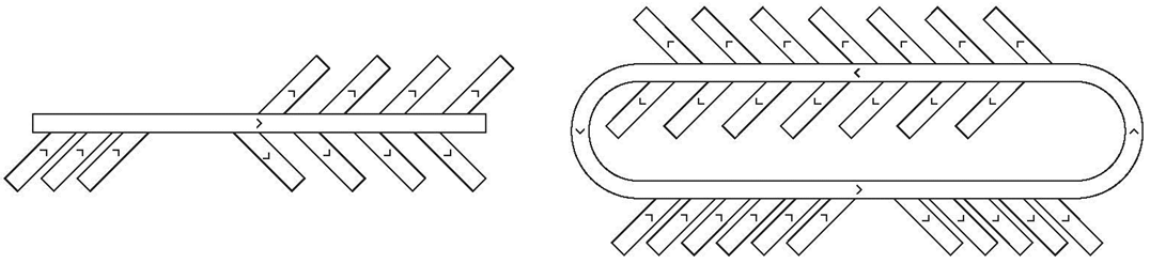
The aim of this chapter is to investigate which scheduling algorithm to use for each industrial sector, operational scenario, and system model. To this end, we build upon existing literature, particularly on the state-of-the-art scheduling algorithm designed for parcel hubs. We present an adapted version of this algorithm that allows for non-zero internal travel times on sorters that in addition may differ per infeed/outfeed combination. Moreover, we show how to apply the scheduling algorithms in baggage handling as a new application area. We also propose extensions to the algorithms in order to adjust to the operational environment in baggage handling. To analyze different scheduling algorithms, we conduct computational studies on different system models and for different operational scenarios.

This chapter is organized as follows: Section 5.1 describes the problem of scheduling inbound containers and presents a generic process model for sorter systems in the two industrial sectors. Then, Section 5.2 provides a literature review on containers scheduling. Next, Section 5.3 discusses scheduling inbound containers in parcel & postal sorting. Afterwards, Section 5.4 discusses scheduling inbound containers in baggage handling. Section 5.5 presents the experimental setup and the results of computational experiments. Finally, Section 5.6 ends with concluding remarks.

[12] *This chapter is based on Haneyah et al. (2013d) and Haneyah et al. (3013e).*

5.1 A generic process model for sorter systems

In this chapter, we focus on sorter systems in baggage handling and parcel & postal sorting. We exploit the commonalities between these industrial sectors to describe the systems in a generic way. However, we first distinguish the basic physical layouts of sorters with a *line configuration* (Figure 5.1a) from more complex sorters with a *loop configuration* (Figure 5.1b). In this chapter, we focus on sorter systems with a loop configuration, but the analysis and the results are applicable to sorters with a line configuration as well.



a) Line configuration b) Loop configuration

Figure 5.1. Basic configurations of sorter systems.

Although baggage handling and parcel & postal sorting are two different industrial sectors, a common operation is scheduling inbound trailers, ramp carts, and ULDs (unit load devices) to unload at the infeeds of the sorter system.

In practice, a lot of information about the contents of specific containers or ULDs is available in the network (Scope 3; see Section 1.2.1). For example, when loading a ULD with bags at an airport of origin, the information about the number of bags in the ULD and their destinations is registered. However, this information is not used at the next airport where this ULD arrives. System-users typically apply a first-come-first-served (FCFS) policy when unloading inbound containers. As a result, uncontrolled peak flows for a particular outfeed may arise. Peak flows for outfeeds may cause them to be overloaded, which may reduce the capacity (measured in sorted items per hour) or at least increase material handling costs.

In sorter systems, outfeeds are generally coupled to specific destinations or regions of destinations. When an outfeed coupled to a particular destination is full, a sorter in a line configuration transports the corresponding items to the outfeed for unsorted items, which leads to an area (downstream the sorter system) where unsorted items are collected. The capacity of the sorter system is indirectly reduced, because the unsorted items have to be re-loaded onto the sorter system for a second delivery attempt. The other solution is that a worker manually delivers the item to the right outfeed, which may increase material handling costs significantly. In a sorter system with a loop configuration, a full outfeed results in recirculation. This reduces the sorter capacity directly, since a recirculating item claims space that otherwise could have been used by another item. In this context, balancing the workload across outfeeds may help reducing the overload incidents and thereby reduce recirculation. This in turn might increase the operational capacity of existing sorter systems or reduce the required

design capacity of new systems. Therefore, the main problem we tackle is how to schedule the unloading operations of inbound containers using the knowledge about their contents, in order to keep the workload on sorters well-balanced. We assume that operators assignment to chutes and outbound destinations planning are tasks that are already done by the system user's process, which provides inputs to the operation of the automated MHS. Moreover, we assume highly loaded sorters. In this context, the main objective is to balance the load in order to maximize throughput.

In baggage handling, incoming ULDs contain either transfer baggage or reclaim baggage that is transported on dedicated unloading conveyors. We do not consider the flow of reclaim baggage further as it is not critical from a scheduling point of view and not part of the flow on the main sorter system. In addition to ULDs, there are bags arriving from check-in desks. Since these arrivals are random and unpredictable, we model them as an uncontrollable inflow.

Contrary to the situation in BHSs, temporary storage facilities are generally not used in parcel & postal sorting, where an outfeed is usually assigned to a single destination during the entire shift. As a result, arriving parcels can be routed to their corresponding outfeeds at any time. This contrasts with baggage handling where an outfeed is usually assigned to multiple flights during the day, and so it is not always possible to route an arriving bag to a destined outfeed. Note that routing may not result directly in successful delivery due to, e.g., congestion, overloaded outfeeds, etc.

Figure 5.2 presents a generic process model for sorter systems in both industrial sectors. Note that there can be several infeed areas (1..*J*), storage areas (1..*S*), and outfeed areas (1..*K*). In this model, we can set the uncontrollable flow equal to zero to model a parcel & postal sorter, where no uncontrollable flow of check-in items exists. Likewise, a zero capacity temporary storage system models a parcel & postal sorter.

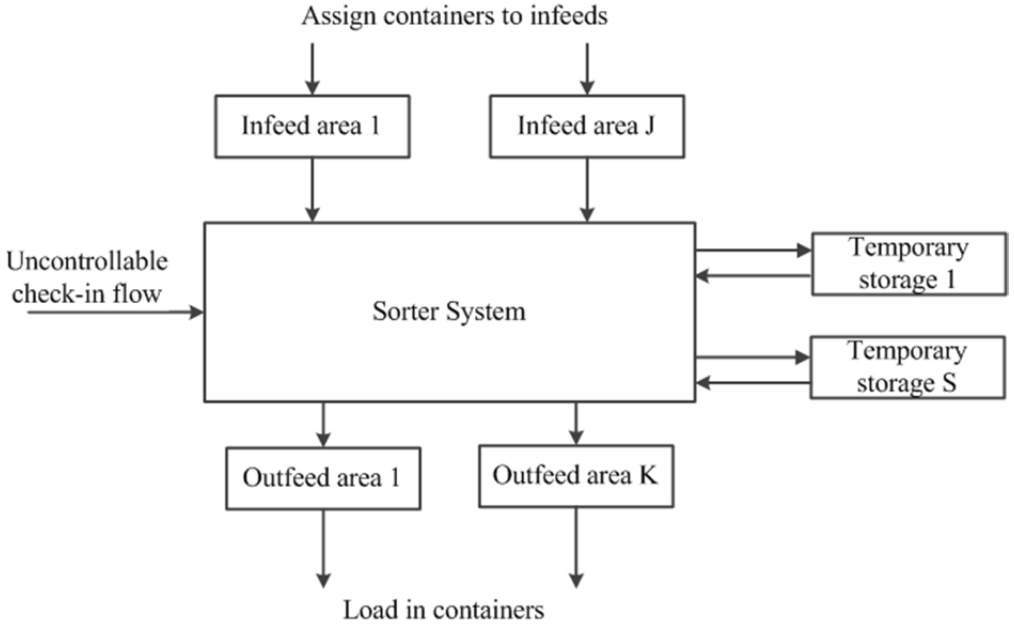


Figure 5.2. A generic process model for sorter systems.

5.2 Literature on container scheduling

Parcel & postal sorting

The *parcel hub scheduling problem (PHSP)*, introduced by McWilliams et al. (2005), is one of the first studies that focus solely on scheduling inbound containers at parcels sorting hubs. The authors consider a parcels sorting hub with three unloading docks and nine loading docks. McWilliams et al. (2005) use a sorter system in line configuration, where they try to minimize the *makespan* of the sorting process. They use a simulation-based scheduling algorithm (SBSA), which is based on a genetic algorithm, to solve the problem and show that their approach is superior to the arbitrary scheduling (ARB) approach, which randomly assigns available containers to available infeeds. McWilliams (2005) shows that similar results can be achieved using iterative local search or simulated annealing techniques. McWilliams (2009a) aims at an approach to balance the workload on the loading docks. He solves small problems to optimality using a binary minimax programming model. For big problem instances, he uses a genetic algorithm that outperforms the SBSA and ARB approaches used in McWilliams et al. (2005). A drawback of this approach is that due to the minimax problem, there may exist many optimal solutions in a large non-convex solution space. In further research, McWilliams (2010) shows that iterative approaches, such as simulated annealing, provide solutions that are on average 6% better than the solutions provided by the genetic algorithm, although large problems require more time to solve.

McWilliams (2009b) develops a relatively simple dynamic load balancing algorithm (DLBA). While the other algorithms require information on all trailers in a particular shift, this algorithm only requires knowledge of the trailers that are waiting to be assigned to an unloading dock. He finds that the DLBA performs much better than random assignments (makespan reduction of 15%). Furthermore, the DLBA is generally better (makespan reduction of 8%) in large complex problems than the approach of McWilliams (2010).

A relevant problem is the cross-docking problem, for which Cohen and Keren (2009) develop an algorithm given forklifts as the mean of freight transport. However, this algorithm does not suit conveyor-based sorter systems. Boysen and Fliedner (2010) present a literature review of cross dock scheduling and propose a research agenda in this field. Although a cross-dock is defined as a no-inventory sorting facility, many studies explicitly use temporary storage. Li et al. (2009) consider the situation in which the floor space in the center of the facility is used to temporarily store products. They study a problem where each inbound trailer is also an outbound trailer that has to be loaded directly after it has been unloaded (unlike the scheduling problem we address). They use a heuristic based on the parallel uniform machine scheduling problem. Yu and Egbelu (2008) focus on coping with the possibilities of limited intermediate storage when scheduling the inbound and outbound operations of a cross-dock to minimize the makespan of the operation. They provide both a mathematical model to solve the scheduling problem to optimality and a heuristic algorithm. However, their approach entails a number of restrictive and unrealistic assumptions, e.g., that all trailers are available at the start of the operation and the unloading sequence of products from an inbound trailer can be determined.

McAree et al. (2002) test the bin and rack assignment model (BRAM) using a realistic case from a large package sort facility. This algorithm was specifically designed for air terminals where inbound ULDs are assigned to bins to be broken into individual pallets. Their main goal is to minimize the operational cost. Because the BRAM is too complex to solve, they develop a new algorithm that finds a solution by iteratively solving the Bin Assignment Model (BAM) and Rack Assignment Model (RAM), both of which are formulated as mixed integer programs (MIPs). McAree et al. (2006) find solutions for different layouts with running times ranging from a few minutes to a few hours, which is fast enough for large scale investment decisions, but too slow for online scheduling decisions.

Gue (1999) determines which docks to use for unloading and which for loading in a cross-dock facility. The author uses a simple algorithm based on scheduling rules and logic similar to that in the approaches of McWilliams (2009b) and Yu and Egbelu (2008).

Werners and Wülfing (2010) consider a more complicated sorter system. In their model of a *Deutsche Post* parcels sorting center, each parcel is unloaded at an unloading dock, sorted into a chute and then assigned to a loading dock. The authors aim at minimizing the total transport effort, i.e., reducing the total distance travelled on the sorters. In order to solve this complex problem, they hierarchically decompose the problem into two sub-problems. They show that their approach ensures a balanced workload over the different areas in the sorting center, whilst providing robust solutions. However, they do not discuss the inbound unloading process, they solely focus on scheduling the outbound process.

Baggage handling

Robusté and Daganzo (1992) provide an extensive overview of the possible pre-sorting strategies, whilst aiming at minimizing baggage handling costs. They model the baggage handling process in detail, by specifying for each strategy the number of moves (for each bag, staff member, container, etc.) and determining the resulting costs of the strategy. They conclude that airlines could achieve significant cost reductions if they segregate the baggage for the larger destinations at the origin airport.

Abdelghany et al. (2006) address the outbound assignment problem, i.e., assigning outfeeds to specific flights. Frey et al. (2010) apply a mathematical approach for a BHS scheduling problem. They consider a baggage handling facility with an EBS (Early Bags Storage) system and assign flights to workstations and carousels. They solve a decomposed problem to determine when to retrieve bags from the EBS. This problem could be converted into a scheduling problem for inbound containers, but there are two main limitations: the assumption that full knowledge is available is questionable, and the runtime of the algorithm is too long.

Although not entirely related to the aforementioned studies, Hallenborg (2007b) presents an approach to determine the *urgency* of a bag. Even though he focusses on agents-based scheduling in BHSs using DCVs (destination coded vehicles), the urgency function of a bag may be useful for us to determine the urgency of a *container* of bags. Hallenborg (2007b) proposes an approach where a bag j becomes urgent

when the time allowance remaining before the destined lateral closes (*cutoff time*) is below a threshold U_{start}. Let C_j denote the cutoff time for the lateral handling bag j and let U_{max} denote the maximum time allowance a bag can have before cutoff, hence the lateral handling bag j opens at time $C_j - U_{max}$ (note that any lateral's open and closure time is of course the same for all bags that are handled there, for one flight, the sub-index j just serves to distinguish the urgencies of bags destined to distinct laterals. Now if we plot the urgency function on a time scale, then the urgency $u_j(t)$ of the bag is determined by the following function:

$$u_j(t) = \begin{cases} -\left(\frac{C_j - t - U_{start}}{U_{max} - U_{start}}\right)^2 & , \; C_j - U_{max} \leq t \leq C_j - U_{start} \\ \frac{1}{(C_j - t)^2} & , \quad C_j - U_{start} < t < C_j \end{cases}$$

Figure 5.3 shows the urgency function when U_{max} is equal to 120 minutes, U_{start} is equal to 30 minutes, and C_j is equal to 150 minutes. As the time t approaches the cutoff time, urgency increases at a decreasing rate until it is zero when the bag has U_{start} time allowance remaining. From that moment on, the bag becomes urgent, its urgency increases at an increasing rate until it tends to infinity when the build area closes, i.e., when $t = C_j$. This approach may provide a good solution to determine which containers need to be unloaded first to ensure that their contents are sorted in time. However, in this approach, we find it disadvantageous that the increase in the urgency is relatively late and that the urgency suddenly becomes very steep.

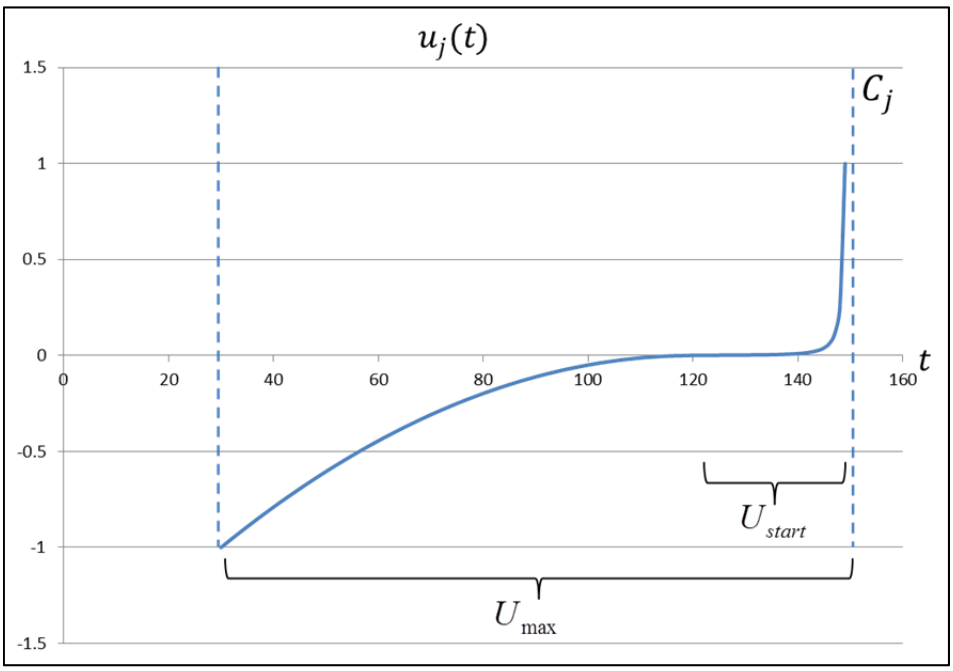


Figure 5.3. Hallenborg's urgency function with $U_{start} = 30$, $U_{max} = 120$, and $C_j = 150$ minutes.

Conclusion

From our review, we find the DLBA (McWilliams, 2009b) of the PHSP (McWilliams et al., 2005) to be the most relevant study from different points of view. First, the DLBA is an online algorithm that does not require full knowledge about incoming containers but uses existing knowledge about containers that are already at the sorting

hub. Second, it is a relatively simple and fast approach, which can be implemented easily in practice. Finally, McWilliams (2009b) reports impressive reductions in the makespan of the sorting operation. Section 5.3 builds further upon this conclusion.

5.3 Scheduling inbound containers in parcel & postal sorting

In this section, we first describe the state-of-the-art load balancing algorithm, which is the DLBA by McWilliams (2009b) as developed for the PHSP (parcel hub scheduling problem). Note that the DLBA does not consider internal travel times on sorters (Section 5.3.1). However, in particular in BHS but also in more complicated sorting systems, sorter travel times may be significant and in addition differ for distinct infeed-outfeed combinations. Therefore, we formulate the load balancing problem with non-zero and variable internal travel times (Section 5.3.2) and present an adapted DLBA to incorporate these features (Section 5.3.3).

5.3.1 The dynamic load balancing algorithm (DLBA)

The DLBA (McWilliams, 2009b) constructs unload schedules for inbound containers using an online scheduling approach. The aim is to dynamically balance the workloads of the outfeeds in order to minimize the probability of an outfeed being overloaded.

The DLBA assumes that the infeeds of the sorter are parallel identical resources. Given this assumption, whenever an infeed becomes idle, a *dispatcher* decides (online) which of the available inbound containers to assign to the idle infeed. The dispatcher's objective is to balance the flow of parcels over the sorter system and to avoid congestion. Thus, monitoring the state of the system is essential. In other words, the dispatcher has to be informed once an infeed becomes idle. Moreover, the dispatcher has to know (at each moment in time) the number of parcels going to a certain destination in the system. This includes the parcels flowing across the sorter and the parcels in the inbound containers being processed at the other infeeds. This information is known because parcels' tags are read when they are unloaded from the containers to the infeeds and when they exit the sorter system at the outfeeds.

The DLBA is an online algorithm that is triggered when an infeed becomes idle. Then the DLBA evaluates all containers available and selects the container that (when assigned to the idle infeed) minimizes the *overflow* on outfeeds (number of excess parcels).

The DLBA assumes zero internal travel times on the sorter. As a result, a parcel loaded on an infeed is immediately unloaded at an outfeed. Note that the assumption of zero internal travel times is not a stronger assumption than the assumption that internal travel times are fixed and equal for any infeed-outfeed pair. As a result of the assumption of zero internal travel times, all infeeds are implicitly assumed to be identical. Therefore, if a container arrives and there are multiple infeeds idle, then the DLBA assigns the container to an arbitrary infeed. The restriction of zero internal travel times might be a valid simplification for small single-sorter systems where internal travel times for any infeed-outfeed pair are comparable, or when unloading a container requires much more time than the internal travel time of parcels in the system. However, this may not hold for larger systems or when the time to unload a

container is short compared to the travel time on a large sorting system with multiple loop sorters, multiple infeed areas, and routing complexities. Therefore, in Section 5.3.2, we propose an adapted version of the DLBA, which takes internal travel times into account.

5.3.2 Problem formulation with internal travel times

In this section, we formulate the workload balancing problem of parcels sorters, which takes (unequal) travel times on the sorter into account. For the PHSP, the DLBA only keeps track of the total number of parcels in the system destined for a specific outfeed. McWilliams (2009b) argues that as long as the total number of parcels in the sorting process for each of the outfeeds was more or less equal, the resulting workload is balanced.

Incorporating travel times means that the workload should not only be balanced over the different outfeeds, but also over time. Determining the expected *outflow* (the number of parcels that arrive at the chute) for each outfeed at each moment in time indicates the excess in capacity (if any) of the outfeed at some moment in time. However, not only the number of excess parcels is relevant, but also the rate at which these excess parcels arrive. We choose to use the squared value of excess flows as an optimization criterion to heavily penalize large excess flows. Another possible goal function would be a *minimax* goal function that minimizes the maximum excess amount. A drawback of this approach is that it may not properly distinguish different solutions. For example, a solution in which only *one* outfeed exceeds its capacity by n parcels is considered the same as a solution in which *all* outfeeds exceed their capacity by n parcels. In fact, in the latter case many more parcels may arrive at a full outfeed and thus are forced to recirculate on the sorter.

Determining the squared excess outflow on a continuous time scale is impractical. A computationally less challenging approach is to use *time buckets*. In the time bucket approach, we determine for each parcel in which time bucket it is likely to arrive at the outfeed. The size of the time buckets is an important model parameter since it affects the level of detail that can be achieved. In order to achieve sufficient detail, we use a time bucket size of 1 minute. This is approximately a quarter of the time required to unload a single ULD and roughly equal to the smallest travel time between infeed-outfeed pairs in the sorter systems that we study. Using time buckets of 1 minute provides sufficient detail but also results in valid and meaningful outflows. Furthermore, we use a concept related to time buckets, namely *container segments*, where we divide the load of each container into fictitious segments of equal size, each needing exactly one time bucket to unload all parcels in the segment.

By including the internal travel times, the problem increases in complexity from the *integer-linear program (ILP)* for the PHSP that McWilliams (2010) provides. Unlike the PHSP, the infeeds are not identical in our problem. Therefore, it is important to know at which infeed a container is docked since travel times to outfeeds may differ amongst infeeds. We now present a mathematical formulation (with linear constraints and a non-linear objective function) for parcel sorting hubs with (possibly unequal) internal travel times.

Parameters

I: set of infeeds ($i \in I$).

O: set of outfeeds ($o \in O$).

T: set of time buckets ($t \in T = \{1,2, \dots\}$).

C: set of inbound containers ($c \in C$).

S_c: number of time buckets (segments) needed to unload container c.

tb: length of one time bucket in seconds.

F_o: outflow capacity of outfeed o (in parcels per hour). Note that it is common practice to define the capacities of sorter systems in parcels per hour.

$f_{c,o}$: number of parcels in container c destined for outfeed o.

$t_{i,o}$: internal travel time from infeed i to outfeed o (in time buckets) excluding possible traffic delays.

Decision variables

$$x_{c,s,i,t} = \begin{cases} 1 & \text{if container } c, \text{segment } s, \text{is assigned to infeed } i, \text{in time bucket } t \\ 0 & otherwise \end{cases}$$

Auxiliary variables

$E_{o,t}$: excess outflow at outfeed o, in time bucket t.

E_{tot}^2: total squared excess flow for all outfeeds.

Constraints

1. *Overlap prevention constraints:* this set of constraints ensures that each infeed is used by at most one container segment per time bucket, where a container is divided into S_c segments of equal size such that in one time bucket the parcels of exactly one container segment can be unloaded at an infeed.

$$\sum_{c \in C} \sum_{s=1}^{S_c} x_{c,s,i,t} \leq 1 \quad \forall t, i$$

2. *Assignment constraints:* these sets of constraints are similar to the ones proposed by McWilliams (2010), except for the addition of the index i for the infeeds. The combination of the following two sets of constraints ensures that each container segment is assigned exactly once and that a container is emptied in successive time buckets.

$$\sum_{i \in I} \sum_{t \in T} x_{c,s,i,t} = 1 \quad \forall c, s$$

$$S_c \cdot x_{c,1,i,t} - \sum_{s=1}^{S_c} x_{c,s,i,(t+s-1)} = 0 \quad \forall c, i, t$$

3. *Parcels flow constraints:* the first term in this set of constraints incorporates the internal travel times $t_{i,o}$. In order to measure the outflow at outfeed o at time t, the flows that were generated by the infeeds $i \in I$ at time $t - t_{i,o}$ have to be considered. Here, we use an approximate outflow measure since we assume that the parcels from a certain container that are destined to a certain outfeed are uniformly distributed over the segments of this container. The second term of

this set of constraints ensures that only outflows that exceed the capacity F_o force the value of $E_{o,t}$ to be positive.

$$\sum_{c\in C}\sum_{s=1}^{S_c}\sum_{i\in I}\left(\frac{f_{c,o}}{S_c}\cdot x_{c,s,i,(t-t_{i,o})}\right)-\frac{F_o\cdot 3600}{tb}\leq E_{o,t}\quad \forall o,t>t_{i,o}$$

Objective function

The objective function minimizes the sum of the squared values of the excess outflows for all outfeeds and all time buckets in the planning horizon.

$$min\quad E_{tot}^2=\sum_{t\in T}\sum_{o\in O}(E_{o,t})^2$$

In this formulation, we assume that the total sorter capacity is sufficient to accommodate incoming parcels, whereas the chute capacity is the bottleneck which we control. However, for systems with limited sorter capacity, further inflow would be blocked when the capacity on the loop conveyor is reached. Another options is to add a constraint to stop the inflow when the occupation of the sorter reaches a predefined limit.

This formulation merely describes the static form of the problem. In order to solve it, we need full knowledge about incoming containers, which is unrealistic and would lead to an intractable problem. In Section 5.3.3, we present a dynamic online approach.

5.3.3 The Adapted-DLBA

In this section, we propose the *Adapted-DLBA (ADLBA)*, which is an online algorithm that modifies the DLBA to incorporate (unequal) travel times on the sorter (without possible traffic delays). The main idea of the ADLBA is to make a selection on which containers to unload at idle infeeds in order to minimize the excess outflow over outfeeds *and* over time. Moreover, we show how to deal with an arriving container if multiple infeeds are available once the container arrives.

Given a certain idle infeed, we examine the containers available in the queue at the sorting hub. For each container c, we examine the $f_{c,o}$ values for every outfeed o. These values represent the number of parcels destined to outfeed o (see Section 5.3.2). Next, we find the time bucket in which the first of these $f_{c,o}$ parcels are expected to arrive at the destination outfeed o and the time bucket in which the last of these parcels are expected to arrive at this outfeed. Then, we evenly spread the $f_{c,o}$ parcels over the time buckets from the first time bucket until the last time bucket. In this way, we determine the expected outflow of parcels to outfeeds if a container c is selected to unload. Before explaining the procedures of the ADLBA, we present additional notations to those presented in Section 5.3.2.

$f_{c,total}$: total number of parcels in container c.

F_i: capacity of infeed i (in parcels per hour).

a^c: the time when the first parcel from container c is announced at an infeed (in seconds).

$TB_{c,o,start}^{container}$: number of the first time bucket in which the parcels from container c which have the destination outfeed o are expected to arrive at outfeed o.

$TB_{c,o,end}^{container}$: number of the last time bucket in which the parcels from container c which have the destination outfeed o are expected to arrive at outfeed o.

$FL_{c,o,start,end}^{container}$: number of parcels from container c that are expected to arrive at outfeed o in any time bucket from $TB_{c,o,start}^{container}$ until $TB_{c,o,end}^{container}$. Note that these are auxiliary variables that depend on alternative assignment decisions.

$Flow_{t,o}$: actual total expected outflow of parcels at outfeed o in time bucket t based on all assignment decisions made so far.

$Flow_{i,c,t,o}$: expected outflow of parcels at outfeed o in time bucket t if we decide to assign container c to infeed i at time a^c.

Now, given a container c assigned to infeed i, then for each destination outfeed o (to which parcels exist in the container), we determine the values of $TB_{c,o,start}^{container}$ and $TB_{c,o,end}^{container}$ using the following equations:

$$TB_{c,o,start}^{container} = \left\lceil \frac{a^c + \frac{3600}{F_i} + t_{i,o}}{tb} \right\rceil$$

$$TB_{c,o,end}^{container} = \left\lceil \frac{a^c + \frac{3600}{F_i} \cdot f_{c,total} + t_{i,o}}{tb} \right\rceil$$

Then, we calculate the expected outflow from container c at outfeed o as follows:

$$FL_{c,o,start,end}^{container} = \frac{f_{c,o}}{TB_{c,o,end}^{container} - TB_{c,o,start}^{container} + 1}$$

Note that the time required to unload a container ($\frac{3600}{F_i} \cdot f_{c,total}$ seconds) depends on all parcels inside the container, while the expected outflow arriving at a specific outfeed depends only on the parcels destined to this outfeed.

The variable $Flow_{t,o}$ keeps track of the total outflow of parcels at outfeed o in time bucket t based on all assignment decisions made so far. Therefore, we increase the value of $Flow_{t,o}$ (for $t \in \{TB_{c,o,start}^{container}, .., TB_{c,o,end}^{container}\}$) by $FL_{c,o,start,end}^{container}$ when the assignment decision of a container is fixed. Figure 5.4 summarizes the procedure to update the outflow values when a container is assigned to an infeed.

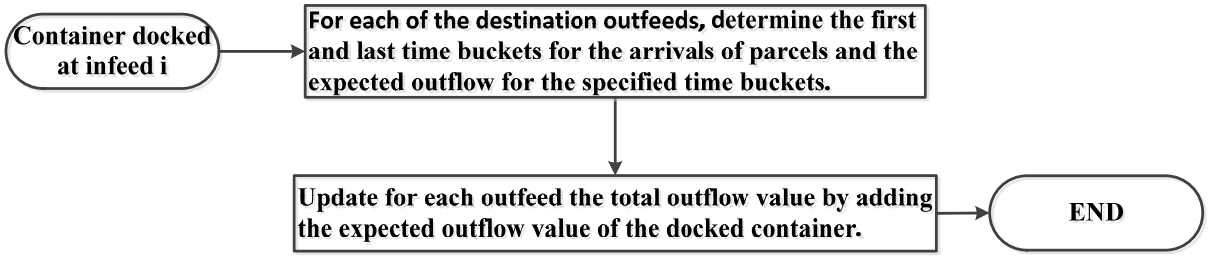


Figure 5.4. Updating the outflow values when assigning a container to an infeed.

In order to make an assignment decision, we have to select one of the containers in the queue to unload at an idle infeed. Therefore, we have to determine the objective value for an assignment decision of a specific container to a specific infeed. In this regard, time buckets in the past are irrelevant, and information about time buckets that are

relatively far in the future is not reliable, because recirculation and merging difficulties may alter these predictions. Therefore, we focus on the total expected outflow in the next 15 minutes. We introduce the set H, which defines all time buckets that are part of the planning horizon. Then, for every alternative decision of assigning a container c to an infeed i at some time a^c, we calculate the total expected outflow for each outfeed and each time bucket $t \in H$ and store these values in the variables $Flow_{i,c,t,o}$.

Let $EF_{i,c}$ be the total expected overflow of parcels at all outfeeds, summed over all time buckets in the planning horizon if container c is assigned to infeed i at time a^c. Then, we determine the objective value for each assignment decision as follows:

$$EF_{i,c}^2 = \sum_{t \in H} \sum_{o \in O} \left(max \left\{ 0, Flow_{i,c,t,o} - \frac{F_o \cdot tb}{3600} \right\} \right)^2$$

The best assignment is the assignment with the lowest $EF_{i,c}^2$ value, which represents the cumulative squares of the overflows over all time buckets and outfeeds if container c is assigned to infeed i. The general procedure to implement the ADLBA is as follows: given an infeed i we calculate the expected overflow for each possible container selection and then select the container that minimizes the total expected squared overflow. However, if a containers arrives and there are multiple infeeds available, then we assign this arriving container to the infeed which minimizes the total expected squared overflow (see Figure 5.5 for an overview of the ADLBA).

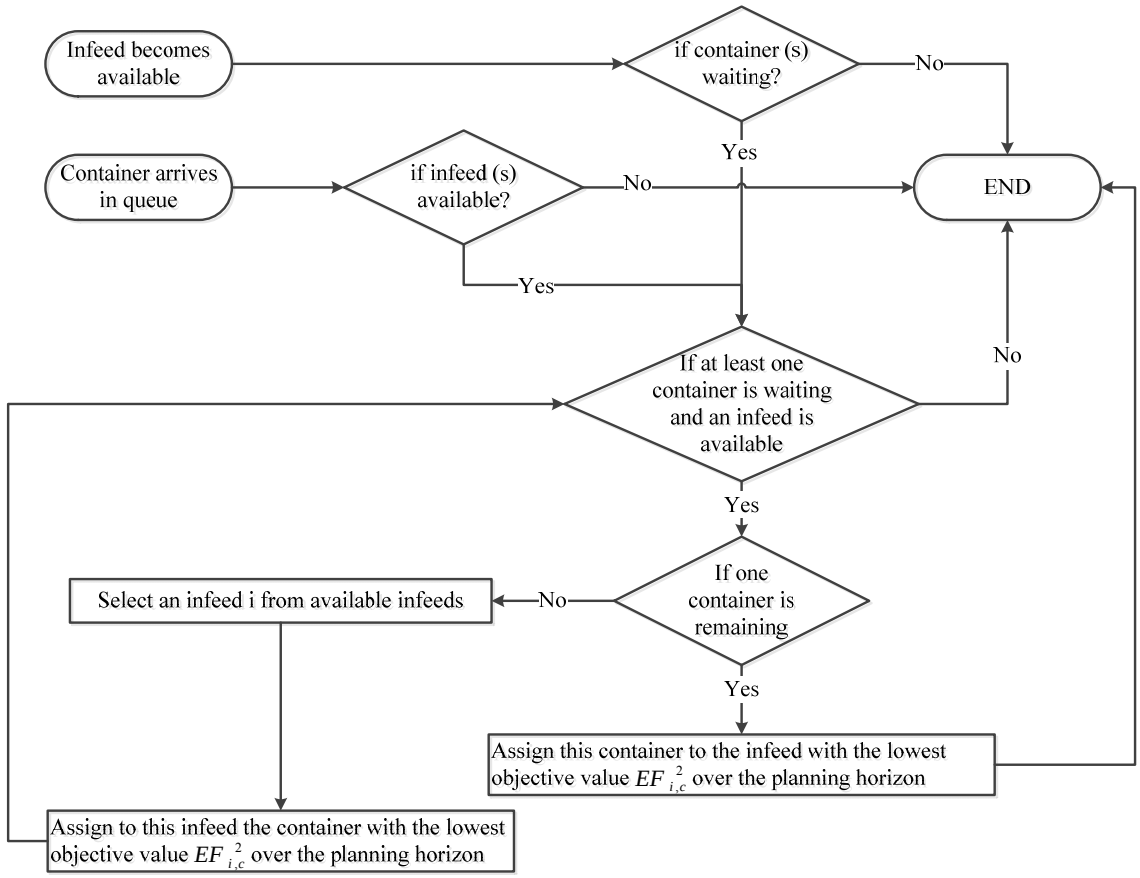


Figure 5.5. Main logic of the ADLBA.

5.4 Scheduling inbound containers in baggage handling

In Section 5.3, we discussed scheduling inbound containers at parcels sorting hubs. We described the state-of-the-art approach and adapted this approach to cover operational scenarios where internal travel times on sorters are important. In this section, we move to the baggage handling sector and adapt the scheduling approaches of parcel & postal sorting to baggage handling systems.

Since we study scheduling approaches for inbound containers in two industrial sectors, we use the same system models in our experiments (see Section 5.5). Therefore, to keep the implementation of scheduling approaches generic to both sectors, we assume that in the BHSs we study, only one lateral is assigned for each destination. However, to model BHSs where more than one lateral is assigned for the same destination, we may aggregate the multiple laterals into one lateral but with a higher capacity, a longer build time, and a larger inbound buffer.

In parcel & postal sorting, inbound containers are the (only) source of parcels' inflow to the sorter. In baggage handling, the inbound containers that we consider are ULDs that contain transfer bags that have to be unloaded at the sorter and to be sorted to destination outfeeds (see Section 5.1). In this context, we note that at the airport of origin (where these containers were loaded), there is normally a segregation of baggage where transfer bags are not mixed in the same container with bags arriving at their last destination. Moreover, as in parcel & postal sorting, the information about the contents of inbound containers in terms of the number of bags and their destination is available from the moment the container is loaded at the airport of origin. However, this information is often not (fully) used in practice.

In addition to inbound containers, there are two sources of inflow that exist in baggage handling but not in parcel & postal sorting. These are the EBS and the check-in flow of bags (see Figure 5.2). In this section, we first study how to incorporate these two elements in the scheduling algorithms (Section 5.4.1). Then, we analyze the baggage handling environment further and propose additional scheduling tools for baggage handling (Section 5.4.2).

5.4.1 The EBS and the check-in baggage

In this section, we present additional procedures that need to be added to the scheduling approaches of Section 5.3 in order to make them applicable in baggage handling. To this end, we show how the EBS and the check-in baggage are incorporated in the DLBA and in the ADLBA. Note that the DLBA is an algorithm that is designed for parcel & postal sorting. Likewise, the proposed ADLBA is for parcel & postal sorting. In this chapter, we test the applicability of both the DLBA and the ADLBA in baggage handling as well. Therefore, in this section, we show how to make both algorithms applicable in baggage handling.

We model the EBS as a resource with a special type of outfeed from the sorter to receive bags. Consequently, when a container is assigned to an infeed and there are bags inside it for which the flights are not open yet for loading, then we set their destination as the EBS outfeed. Then, we handle these bags similar to other bags with

other destination outfeeds. However, when the destinations of bags in the EBS are open on the sorter, then the bags have to be retrieved from the EBS to deliver them to their destination outfeeds. The EBS has a special infeed for the flow of bags to the sorter.

In the remainder of this section, we show how to incorporate the inflow from the EBS and check-in desks to the sorter. To this end, we merely show how the baggage flows generated by these two elements are incorporated in the outflow figures at outfeeds. We use these outflow figures when scheduling inbound containers (see Section 5.3). In this chapter, we do not propose decision-making algorithms for the EBS. On the other hand, the check-in flow is uncontrollable by nature and we model it as an uncontrollable and stochastic flow.

The DLBA in baggage handling

In order to implement the DLBA in baggage handling, we have to incorporate the inflows from the EBS and from check-in desks in the outflow numbers to outfeeds. For the bags from the EBS, we release the bags for destination d open at outfeed o as soon as this destination is open and then increase the outflow to the assigned outfeed o by the number of bags in the EBS with destination d. Moreover, to process a checked-in bag using the DLBA, we increase the outflow for the bag's destination outfeed by one as soon as the bag is announced in the system.

Incorporating the baggage flow from the EBS in the ADLBA

For this process, we assume that when a destination is open at a certain outfeed o, then the bags destined to this outfeed are retrieved from the EBS and arrive homogeneously at outfeed o over a set of time buckets that we define beforehand. We determine the time buckets in a way similar to determining the time buckets when a container is assigned to an infeed (see Section 5.3.3). Before showing this procedure, we present additional notations to model the flow from the EBS.

a_d^{EBS}: the time when the first bag with destination d (i.e., flight) is announced at the infeed from the EBS to the sorter (in seconds).

EBS_d: total number of bags in the EBS with destination d.

F_{EBS}: retrieval rate from the EBS (in bags per hour).

$t_{EBS,o}$: travel time from the EBS to outfeed o (in seconds).

$TB_{o,start}^{EBS}$: number of the first time bucket in which the bags with destination outfeed o are expected to arrive at the outfeed o, when announced at a_d^{EBS}.

$TB_{o,end}^{EBS}$: number of the last time bucket in which the bags with destination outfeed o are expected to arrive at the outfeed o, when announced at a_d^{EBS}.

$FL_{o,start,end}^{EBS}$: number of bags from the EBS that are expected to arrive at outfeed o in any time bucket from $TB_{o,start}^{EBS}$ until $TB_{o,end}^{EBS}$.

Note that in the BHS case, it is not always sufficient to define outfeeds as destinations. We distinguish some parameters with destinations (e.g., flights) because a single

outfeed is assigned to multiple destinations during the day. Given the aforementioned parameters, we determine $TB^{EBS}_{o,start}$, $TB^{EBS}_{o,end}$, and $FL^{EBS}_{o,start,end}$ as follows:

$$TB^{EBS}_{o,start} = \left\lceil \frac{a^{EBS}_d + \frac{3600}{F_{EBS}} + t_{EBS,o}}{tb} \right\rceil$$

$$TB^{EBS}_{o,end} = \left\lceil \frac{a^{EBS}_d + \frac{3600}{F_{EBS}} \cdot EBS_d + t_{EBS,o}}{tb} \right\rceil$$

$$FL^{EBS}_{o,start,end} = \frac{EBS_d}{TB^{EBS}_{o,end} - TB^{EBS}_{o,start} + 1}$$

In Section 5.3.3, we used the variable $Flow_{t,o}$, which keeps track of the total outflow of parcels at outfeed o in time bucket t from all assigned containers. In baggage handling, we include the flow from the EBS by increasing the value of $Flow_{t,o}$ by $FL^{EBS}_{o,start,end}$ for $t \in \{TB^{EBS}_{o,start}, .., TB^{EBS}_{o,end}\}$.

Incorporating the check-in baggage arrivals in the ADLBA

Let a^{ci} be the time when a checked-in bag is announced in the system (in seconds). Then, we determine $TB^{ci}_{o,start}$, which is the number of the time bucket in which the checked-in bag is expected to arrive at its destined outfeed o, as follows:

$$TB^{ci}_{o,start} = \left\lceil \frac{a^{ci} + t^{ci} + t_{i,o}}{tb} \right\rceil$$

Here t^{ci} is a time delay that we add because bags are announced before they actually arrive at the check-in infeed of the main sorter. What remains is to increase, by one, the variable $Flow_{t,o}$ for the destination outfeed o and $t = TB^{ci}_{o,start}$.

Conclusion

We discussed that for BHSs, inbound containers are not the only source of inflow. There are two other sources of inflow, which are the EBS and the check-in baggage. In this section, we showed how these additional sources of inflow can be incorporated in order to use the containers scheduling algorithms of parcel & postal sorting (see Section 5.3). However, keeping track of the inflow of bags from these two resources may not be sufficient to implement scheduling approaches of parcel & postal sorting. In Section 5.4.2, we analyze the characteristics of the operational environment in baggage handling, which differs from parcel & postal sorting. Then, we examine whether we need additional scheduling tools in baggage handling.

5.4.2 Extensions to the scheduling approaches

In the DLBA and the ADLBA, we schedule the inbound containers with the aim to balance workload in the system. In doing so, we implicitly assume that the inbound containers have equal priority. This assumption does not hold for the baggage handling sector where flights, and as a result bags, have different deadlines. Moreover, in baggage handling, a single outfeed is usually assigned to multiple flights during the day. As a result, for the early arriving bags, the destination outfeeds may not be open yet. We solved this problem in Section 5.4.1 by assigning such early bags to the EBS

outfeed and then dealing with the EBS outfeed as any other outfeed in the system. However, we propose another method to deal with containers carrying early bags.

In this section, we propose two extensions to the scheduling approaches that we study. These extensions concern the topics of *urgency* and *delayability*. Both extensions deal with inbound containers that carry transfer baggage.

Urgency

Hallenborg (2007b) provides an approach to determine a bag's urgency (see Section 5.2). We build on this approach to calculate the urgency of a *container* of bags. Note that from a certain point in time onwards, it becomes physically impossible to transport bags through the sorter system to arrive at the assigned outfeeds before they close. For our problem, bags in a container for destination d become non-urgent if they have less than a duration of time U_{end} remaining before cutoff time. In the systems that we model, we set this time duration U_{end} equal to the internal travel time $t_{i,o}$, where i is the infeed under consideration, and o is the outfeed to which destination d is assigned. Thus, a bag that cannot be delivered on time, even if it was the first to be unloaded from a container, is non-urgent. Unfortunately, this extension does not suit scheduling approaches that do not keep track of internal travel times (e.g., DLBA). For such approaches, we use a fixed value of 5 minutes that is an estimation of the average internal travel times in our experimented system layouts. Another relevant time threshold is U_{start}, where bags become urgent if they have less than this threshold remaining before cutoff time. Let C_j denote the cutoff time for a bag j.

We use a simple rule from practice, where the bags to a certain destination are urgent for 30 minutes. Thus, U_{start} is equal to $U_{end} + 30$ minutes. We model the urgency of a bag j to start at zero (the minimum value) when a bag has U_{start} time allowance remaining, and to increase at an increasing rate to one (the maximum value), when the bag has U_{end} time allowance remaining. We define the urgency function of a bag j as:

$$u_j(t) = \left(\frac{t - (C_j - U_{start})}{U_{start} - U_{end}} \right)^2, \qquad C_j - U_{start} \leq t \leq C_j - U_{end}$$

Figure 5.6 shows the urgency function of a bag j, assuming C_j is equal to 40 minutes.

To determine the urgency of a container, we propose one of three simple approaches. A container is assigned either the *maximum* of all individual bags' urgencies, the *average* of all individual urgencies, or the *sum* of all individual urgencies. The maximum measure might often fail to correctly differentiate between the urgencies of containers. For instance, if the bags for one destination are urgent, then a container that holds one bag for this destination is as urgent as a container that holds 10 bags for this destination. The average measure tackles this issue, as the latter container would have an urgency that is 10 times higher than that of the former, assuming that they contain the same number of bags. However, this assumption is the drawback of this approach, as a container holding only 13 bags, provided they are all urgent, may receive priority over a container in which 14 out of 15 bags are urgent. The sum approach solves this issue, and so it is used in this research.

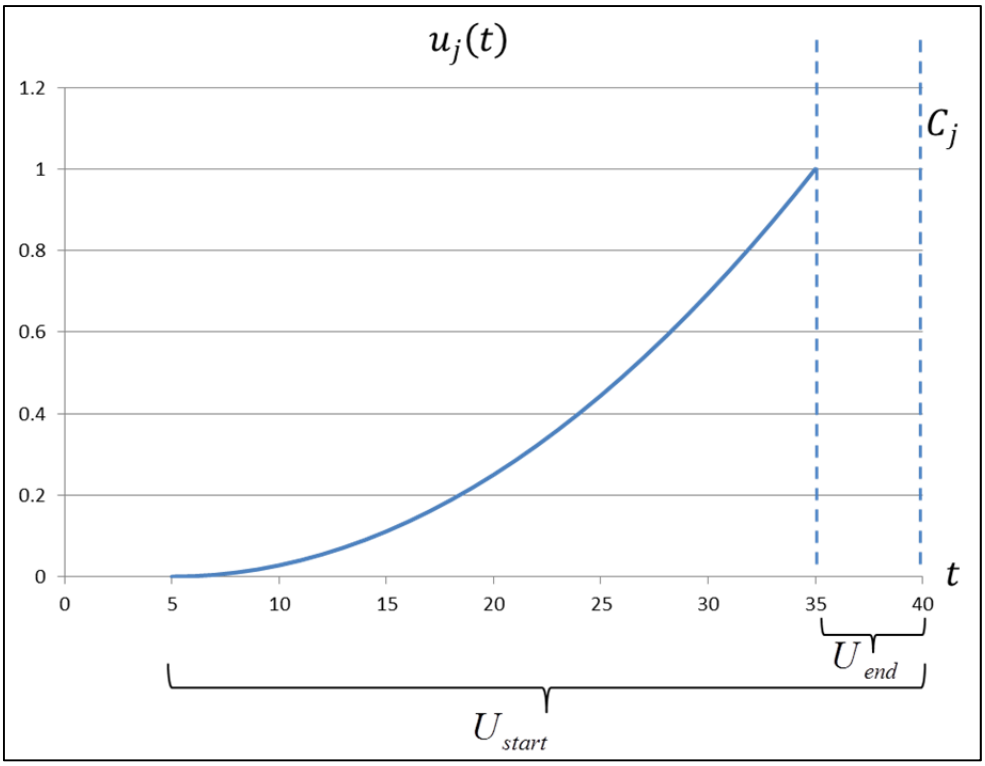


Figure 5.6. The urgency function for a bag j with $U_{start} = 35$, $U_{end} = 5$, and $C_j = 40$ minutes.

In the context of load balancing, we use a priority procedure to select a subset of available containers to which we apply scheduling approaches. We decide to disregard containers with urgency less than 75% of the maximum urgency container. This ensures that a priority container is scheduled, while also balancing the workload among the outfeeds. The DLBA or the ADLBA may accommodate the priority extension by calling the priority procedure (Figure 5.7) when they start searching for a suitable container to assign to infeed i.

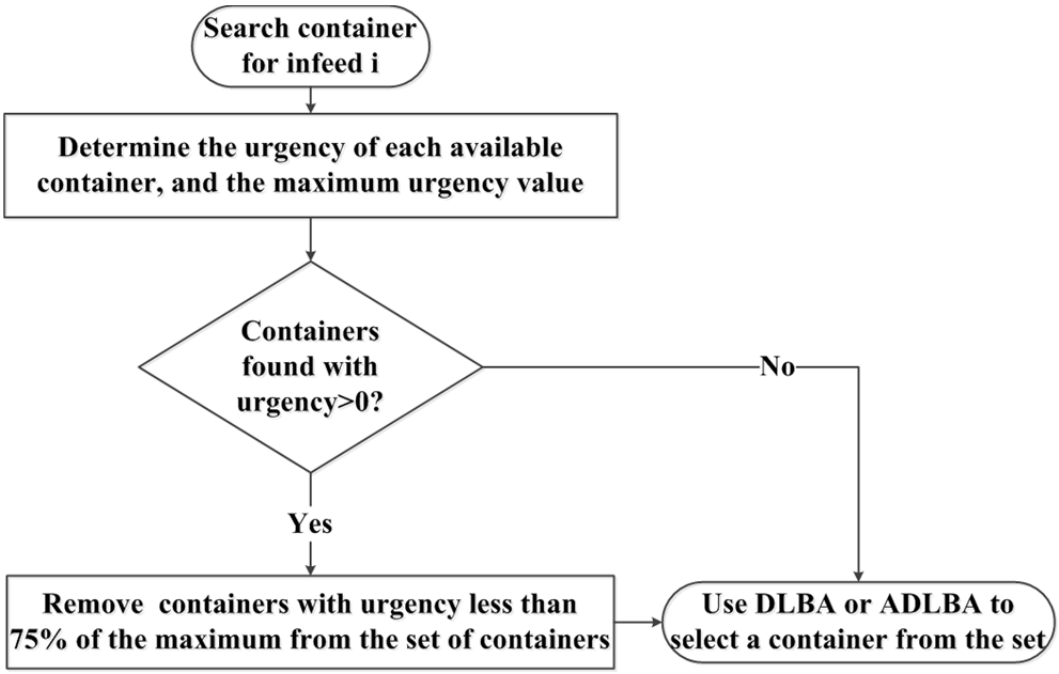


Figure 5.7. Priority scheduling.

Delayability

So far, we have implicitly assumed that every arriving container joins the queue of waiting containers that are announced to the dispatcher. Then, the dispatcher decides which container to unload at which infeed. However, it is possible to deliberately *delay* specific containers and not announce them to the dispatcher. This can be advantageous for two reasons: first, many airports lack infeed capacity during peak hours (usually workdays between 6am and 9am). To reduce these peaks, we may temporarily park some of the inbound containers (carrying transfer baggage) on a remote yard. In doing so, we delay containers that carry only early baggage. This reduces the workload in the BHS and so we expect less congestion on the sorter. Less congestion means less traffic delays, which is beneficial for several reasons. For example, the ADLBA uses an approach that considers travel times on the sorter without possible traffic delays. So when traffic delays are less, the estimated travel times become more reliable.

The second benefit of delaying some containers is that we may be able to reduce the required EBS systems size. In current practice, early bags are stored in expensive EBS systems. Keeping them on the yard, in the containers in which they arrived, is a much cheaper solution.

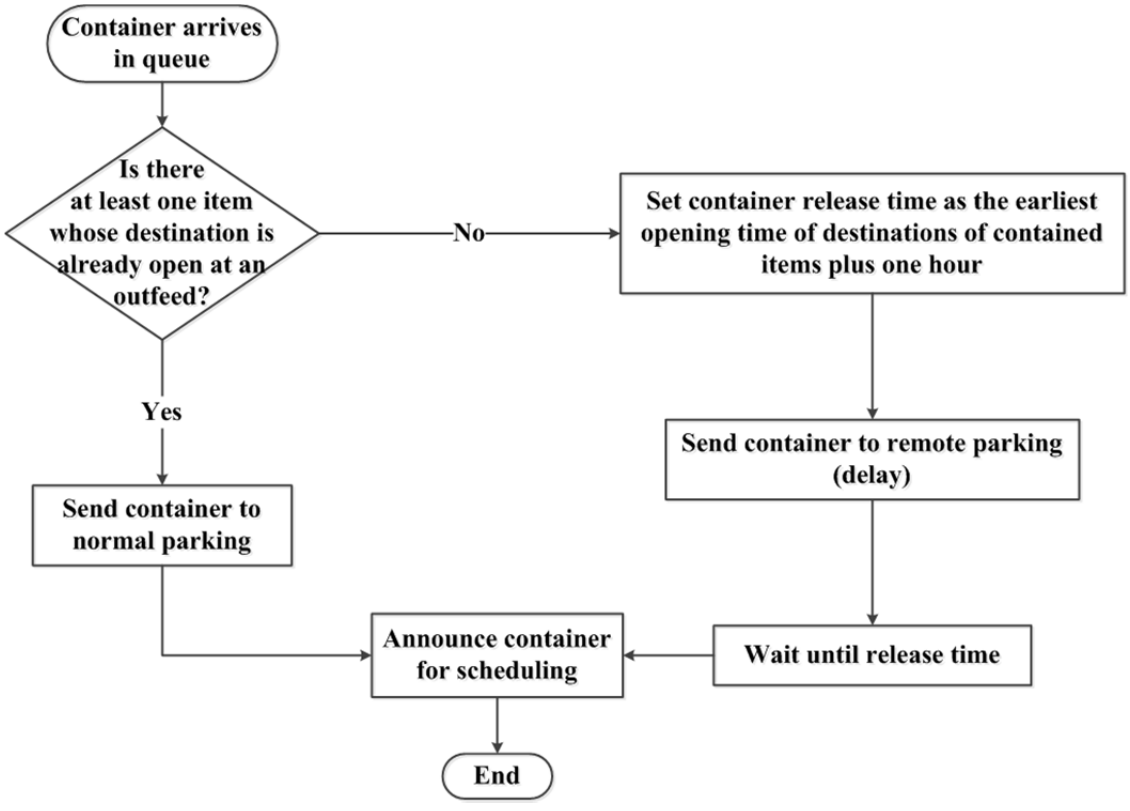


Figure 5.8. Delayable scheduling.

The decision to delay a container is made upon arrival. However, to decide when to bring a container back to the dispatcher, we consider the fact that in the BHSs that we analyze in this chapter, a destination is assigned to a single outfeed for approximately three hours. Therefore, according to experts' opinion, we propose to make a container available one hour after the destination of one of the contained items is open at the

assigned outfeed, leaving two hours to sort the bag(s) that triggered our decision. We call the procedure to delay containers the *delayability* approach and it is executed as soon as a container arrives in the queue (see Figure 5.8).

5.5 Computational studies

In this section we describe the experimental setup for implementing the scheduling algorithms (Section 5.5.1) and discuss the implementation results (Section 5.5.2). The ultimate result of our computational studies is to propose a matrix where we show what algorithm works best for what system model, industrial sector, and operational scenario (Section 5.5.3).

5.5.1 Experimental setup

For our experiments, we test the performance of four algorithms: first-come-first-served (FCFS) as a common current practice, arbitrary scheduling (ARB) merely as an academic benchmark, the DLBA, and the ADLBA. We use the Applied Materials AutoMOD simulation software package to apply the scheduling approaches on sorter systems. Based on layouts that are frequently delivered by our industrial partner, we developed three simulation models of sorter systems with simple built-in local traffic control rules. However, we do not further invest in the control logic of sorter systems as this chapter is concerned with inbound operations scheduling and not with the local traffic control on sorters. The simulation models we use are as follows:

- A single sorter in loop configuration with one infeed area and one outfeed area, where each area consists of three conveyors. Moreover, there is one infeed for check-in baggage and one EBS (Figure 5.9a).
- A single sorter in loop configuration with two infeed areas and two outfeed areas, where each area consists of three conveyors. Moreover, there is one infeed for check-in baggage and one EBS (Figure 5.9b).
- Two sorters in loop configuration, where each sorter consists of one infeed area and one outfeed area. In turn, each area consists of three conveyors. Moreover, each sorter has one infeed for check-in baggage and one EBS. *Crossovers* are conveyors with limited capacity that connect the two sorters (Figure 5.9c).

We use a tuple notation to identify layouts: number of loops, number of infeed and outfeed areas, special transport routes. Using c to denote the crossovers and 0 to denote no specials, the three layouts mentioned above can be identified by the tuples 110, 120, and 22c. The *catchall* outfeed collects items that cannot be sorted due to, e.g., a missed flight.

Regarding datasets, we distinguish both industrial sectors based on their specific characteristics, e.g., the presence or absence of check-in flows, and the size of the containers. A second distinction is based on the quantities of items going to certain destinations inside one container. A *homogeneous* distribution means that inside one container, the number of items for a specific destination is nearly the same for all destinations. A *heterogeneous* distribution means that some containers hold significantly more items for destination a while other containers hold more items for destination b. Before describing how we apply this distinction in generating the

contents of containers, we present the following four scenarios that we use in the experiments:

- Parcel & postal sorting, homogeneous distribution (PP-even).
- Parcel & postal sorting, heterogeneous distribution (PP-uneven).
- Baggage handling, homogeneous distribution (BHS-even).
- Baggage handling, heterogeneous distribution (BHS-uneven).

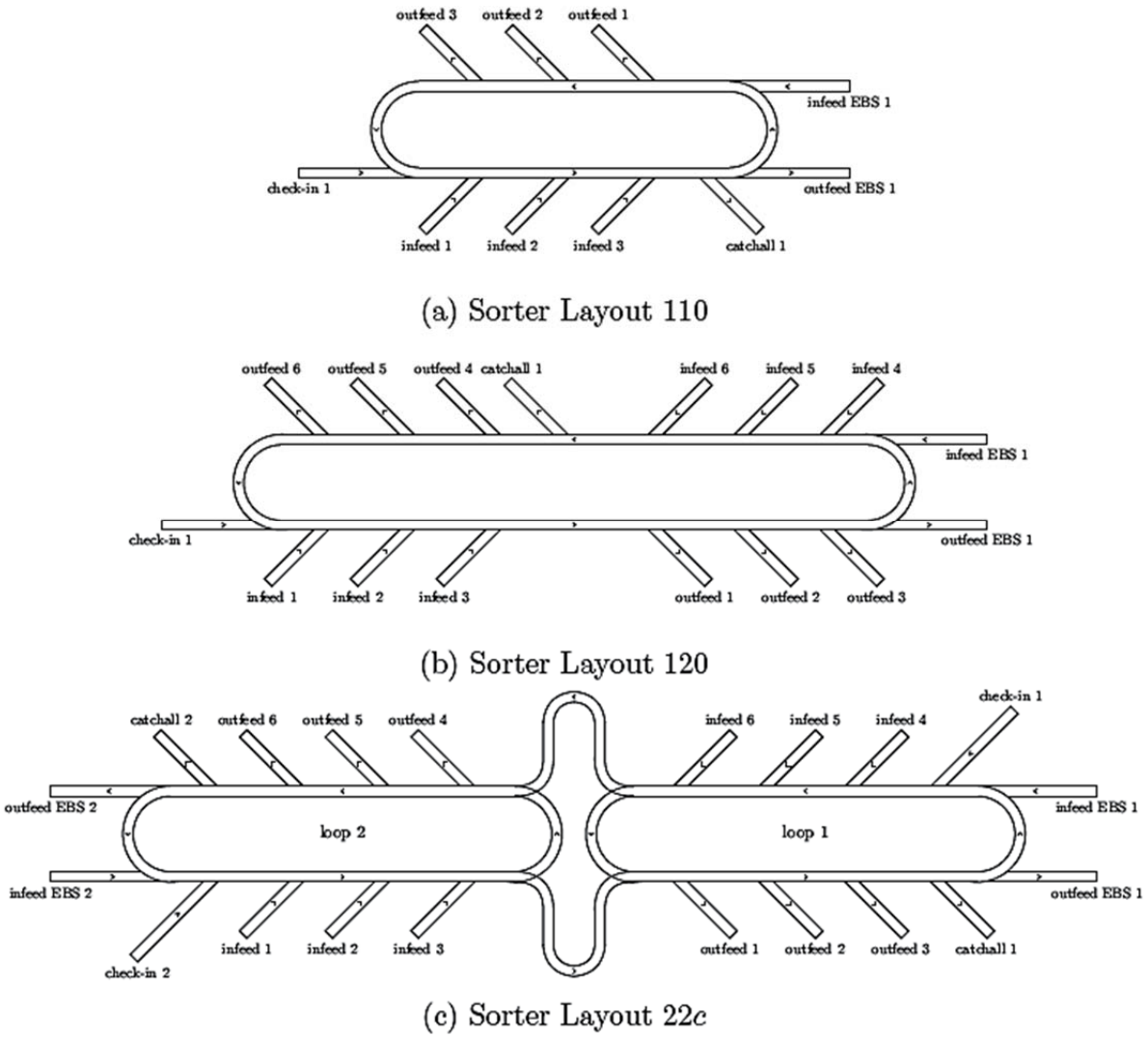


Figure 5.9. Layouts of the three test models.

Let n be the number of outfeeds in a modeled system. In parcel & postal sorting, the number of outfeeds equals the number of destinations modeled. The destinations are open at the outfeeds during the entire simulation. Then for the PP-even scenario, each destination has $\frac{1}{n}$ probability of occurrence. For the PP-uneven scenario, we use $1 + n$ container types, one with an even distribution and n with a preferred destination. For the latter, the preferred destination has 0.50 probability of occurrence while all other destinations have $\frac{0.50}{n-1}$ probability of occurrence. The probability that a specific container type is selected is $\frac{1}{1+n}$.

In baggage handling, the total number of destinations is twice the number of outfeeds. We divide the destinations in two sets with n destinations each. At the start of the simulation, the destinations from the first set of n destinations are opened successively at the outfeeds as follows: the first outfeed in every outfeed area is opened for a destination. After 15 minutes the second outfeed in every outfeed area is opened for another destination and after another 15 minutes the third outfeed in every outfeed area is opened for another destination. Each outfeed is open for a destination for three hours and then the outfeed is left unassigned for half an hour. Thereafter, the outfeed is open for a destination from the second set of n destinations. Given this operational environment, we cannot model the contents of containers similar to parcel & postal sorting (where outfeeds are open for the same set of destinations during the entire shift) because then many bags may arrive after their destinations have closed. Therefore, we introduce time windows in which containers with specific contents may arrive. In this context, the BHS-even scenario consists of two container types, the first type holds mainly (85% of its contents) bags for the first set of destinations, each of these destinations has $\frac{0.85}{n}$ probability of occurrence. The remaining 15% of the container's contents is for the second set of destinations with $\frac{0.15}{n}$ probability of occurrence per destination. The second container type consists solely of bags for the second set of destinations, each with $\frac{1}{n}$ probability of occurrence.

The BHS-uneven scenario consists of $2 + 2n$ container types, where the first two types are the same as in the BHS-even scenario. The other $2n$ container types have 90% of their contents for one preferred destination. Each of these container types has an arrival time window that starts as soon as the preferred destination is open at an outfeed and ends 30 minutes before the outfeed closes. For n container types, the preferred destination is one of the first set of destinations. For these containers, the remaining 10% of their contents is distributed evenly over all other destinations, each with $\frac{0.10}{2n-1}$ probability of occurrence. The preferred destination for the other n container types is one of the second set of destinations. For these containers, the remaining 10% of their contents is evenly distributed over the second set of destinations, each with $\frac{0.10}{n-1}$ probability of occurrence.

Table 5.1 provides an overview of the selected values for the scenario parameters. Note that we generate unrealistically high loads on baggage sorters, which can occur only in high peak hours. We do so to better test the impact under hard operational conditions.

We emphasize that there is no single KPI that holds for both industrial sectors (see Chapter 1). While the KPI of a BHS is the irregularity rate, throughput is the KPI in parcel & postal sorting systems. In addition to these KPIs, we report on other performance indicators that are of interest, i.e., average container waiting time, maximum number of waiting containers, recirculation rate, and the maximum number of bags in the EBS for BHSs.

		PP-even	PP-uneven	BHS-even	BHS-uneven
destinations [#]		3	3	6	6
containers [#]		25	25	70	70
interarrival time [*sec*]	lb	250	250	900	900
	ub	325	325	1500	1500
parcels in container [#]	lb	100	100	20	20
	ub	150	150	30	30
batchsize [#]	lb	1	1	4	4
	ub	1	1	6	6
container types [#]		1	4	2	8

(a) Model 110

		PP-even	PP-uneven	BHS-even	BHS-uneven
destinations [#]		6	6	12	12
containers [#]		50	50	150	150
interarrival time [*sec*]	lb	100	100	800	800
	ub	188	188	1400	1400
parcels in container [#]	lb	100	100	20	20
	ub	150	150	30	30
batchsize [#]	lb	1	1	8	8
	ub	1	1	10	10
container types [#]		1	7	2	14

(b) Model 120

		PP-even	PP-uneven	BHS-even	BHS-uneven
destinations [#]		6	6	12	12
containers [#]		50	50	150	150
interarrival time [*sec*]	lb	100	100	800	800
	ub	188	188	1400	1400
parcels in container [#]	lb	100	100	20	20
	ub	150	150	30	30
batchsize [#]	lb	1	1	8	8
	ub	1	1	10	10
container types [#]		1	7	2	14

(c) Model 22c

Table 5.1. Scenario parameters per simulation model.

5.5.2 Results

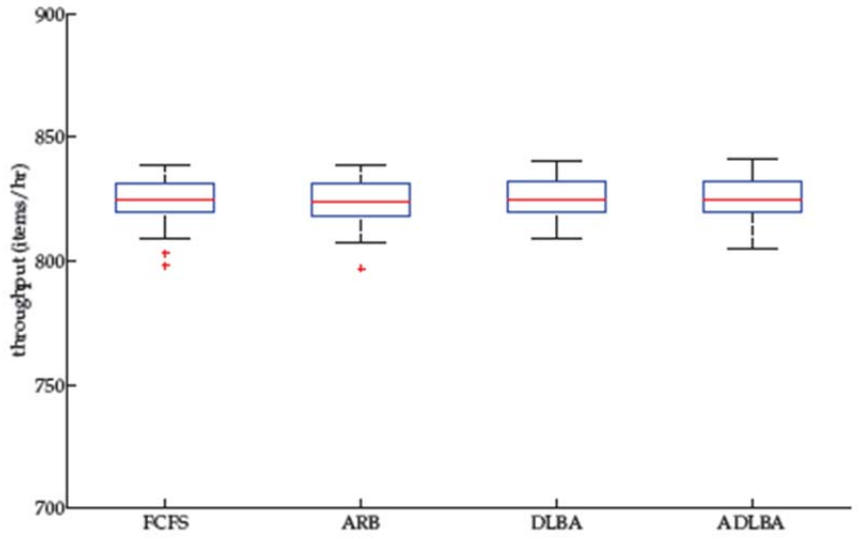
We distinguish between statistical significance and operational significance when comparing the results of different algorithms. A difference is statistically significant, if we can statistically prove it exists with 95% confidence. However, a statistically significant measure may not be relevant from an operational perspective. For example, a statistically significant difference of one item per hour on throughput in a system sorting thousands of items per hour is operationally unimportant.

In the boxplots used to show results, the central rectangle spans the first quartile to the third quartile. The segment inside the rectangle shows the median and the two *whiskers* indicate the extreme values that are not outliers (i.e., within 1.5 times the interquartile range of the first and third quartile). Finally, '+' symbols indicate outliers.

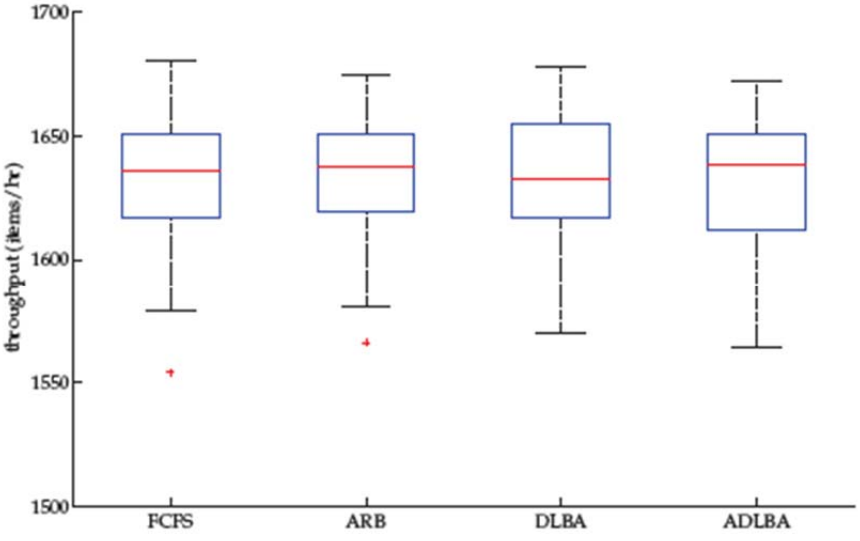
5.5.2.1 Parcel & postal sorting

PP-even

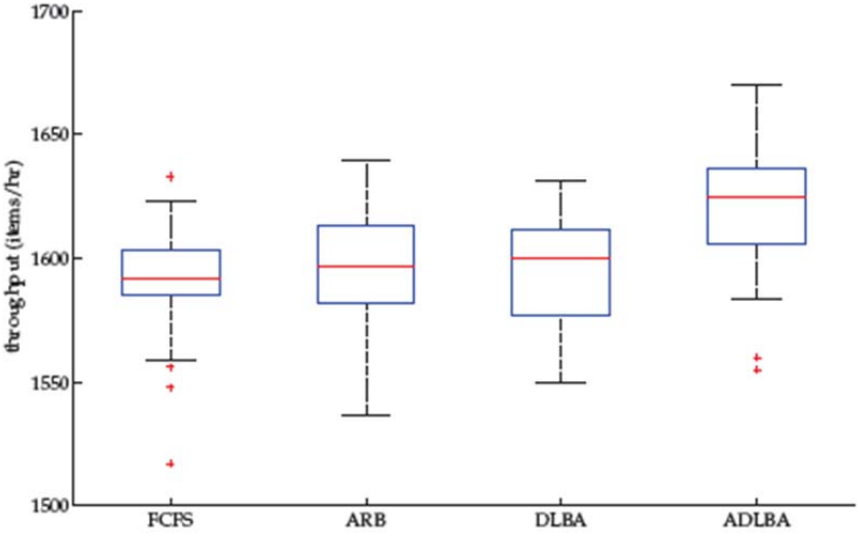
For the evenly distributed scenario in parcel & postal sorting, the simulation studies show that for models 110 and 120 there is no statistical difference between any of the scheduling approaches (Figures 5.10a and 5.10b). However, Figure 5.10c shows that for model 22*c* the ADLBA approach outperforms all others, with a throughput that is approximately 25 items per hour (1.5%) higher. The original DLBA performs statistically just as well as FCFS and ARB and is therefore not considered to be an improvement. A possible explanation for this behavior is related to the infeed assignment problem. FCFS, ARB, and DLBA assign a container to an available infeed irrespective of its location on the sorter system, and so when containers are homogeneous there is no clear optimization criterion. However, the ADLBA assigns the containers to infeeds that are selected based on the load balancing criterion. Therefore, although containers are similar, the ADLBA still balances the workload over the two separate sorters and over time, which yields improvements.



(a) model 110



(b) model 120



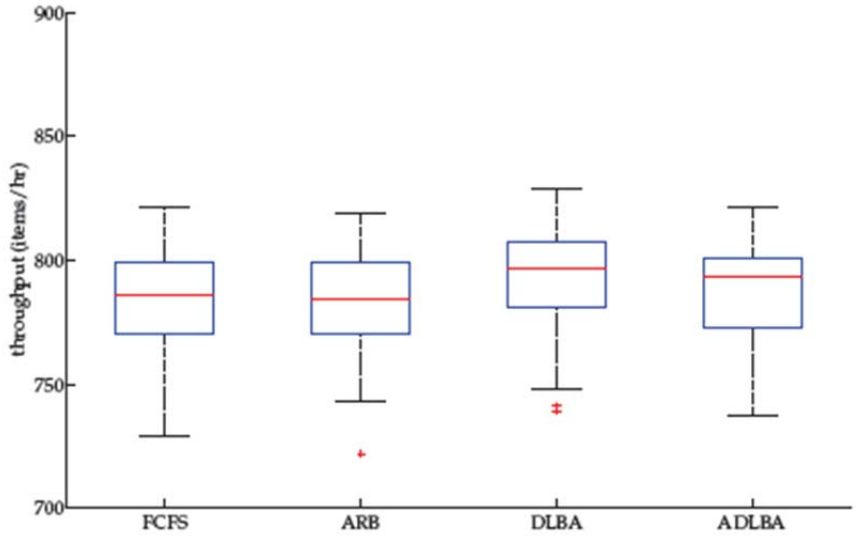
(c) model 22c

Figure 5.10. Throughput using the PP-even scenario.

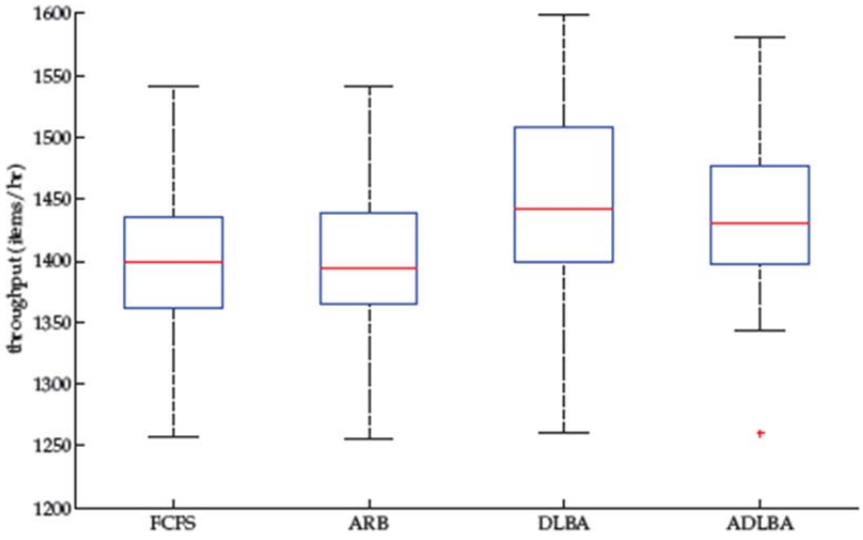
PP-uneven

For the unevenly distributed scenario in parcel & postal sorting, the simulation studies show that the load balancing algorithms (DLBA and ADLBA) outperform the FCFS and ARB in all simulation models (see Figure 5.11). In particular, the DLBA proves to be an interesting approach as it outperforms FCFS by 11, 52, and 63 items per hour (1.4, 3.7, and 4.5%) for models 110, 120, and 22*c* respectively. As containers become differentiable, the original DLBA proves to be a better scheduling approach than the ADLBA for all models, although not from an operational point of view. For models 110, 120, and 22*c* the differences are respectively 6, 15, and 18 items per hour (0.8, 1.0, and 1.3%).

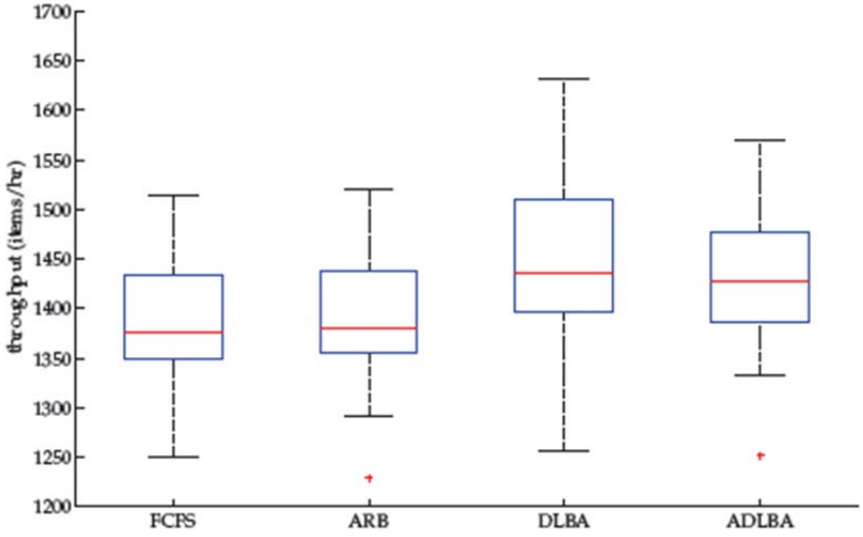
Although not displayed in the figures, the simulation studies also show interesting results regarding the waiting containers. The DLBA is able to reduce the maximum number of containers in the queue, compared to FCFS, by 0.5, 2.5 and 2.7 (4.8, 11.3, and 12.5%) for models 110, 120 and 22*c* respectively. The results of the latter two models, in particular, suggest that significant reductions in required yard space could be achieved. Furthermore, the results show that the DLBA is able to significantly reduce the average waiting time of a container. Specifically, the reductions for models 120 and 22*c* are impressive (approximately 10 minutes and just over 20% of the original waiting time).



(a) model 110



(b) model 120



(c) model 22c

Figure 5.11. Throughput using the PP-uneven scenario.

5.5.2.2 Baggage handling

In baggage handling, we test the scheduling algorithms and also the extensions. We use the suffixes '*u*' and '*d*' to indicate the urgency and the delayability extensions respectively in Figures 5.12 and 5.13 and in the discussion below.

There are key points that make the results in baggage handling incomparable to those in parcel & postal sorting. First, the contents of the containers differ in the destinations of the items, the number of items contained, and more important in urgency. Second, the focus is now on the irregularity rate (number of bags missed per 1000 bags), which is a different KPI. Third, in a parcel sorter system, the impact of assignment decisions is directly realized because parcels are sorted immediately to outfeeds. In baggage handling, however, there is the storage function and flights' schedules that heavily influence the interdependency between the inflow and the outflow.

BHS-even

For the evenly distributed baggage handling problem, the ADLBA approach is preferred in model 110, as it outperforms the DLBA by 0.4 (7.0%). In model 120, FCFS is the best approach. It outperforms the DLBA by 1.7 (2.5%). Note that since containers are homogeneous, the DLBA does not have a significant advantage over the simple FCFS approach. Finally, in model 22*c*, the DLBA is the best approach. It outperforms FCFS and the ADLBA by 2.8 and 2.5 (18.4% and 16.3%) respectively. Apparently, the estimations of the ADLBA with regard to internal travel times are more reliable in simple sorters (model 110). In more complex sorters and given a more stochastic environment, the ADLBA estimations appear to be less reliable and thus simpler approaches that do not depend on estimations of travel times perform better. In this context, an issue for future research is to propose methods that come up with better estimations of internal travel times.

BHS-even with extensions

The urgency extension improves the results of the DLBA and the ADLBA approach. However, the improvements are often only statistically and not operationally significant. On the contrary, the d-extension tremendously improves the performance of the sorter systems. Generally speaking, this extension reduces the irregularity rate by 1.0, 47.6 and 7.3 (20.8, 68.7 and 50.1%) in models 110, 120, and 22*c* respectively (compared to the best performing approach per model). The extreme increase in performance in model 120 is partly due to the unrealistically high irregularity rates caused by a relatively high workload (70% of outfeed capacity), which we used on purpose. The d-extension also affects other PIs, e.g., it reduces the required EBS space by 96, 230, and 204 items (approximately 30% in all cases) for models 110, 120, and 22*c* respectively. These improvements are achieved by delaying on average 5.6 (out of 70) containers in model 110 and 12.2 (out of 150) containers in models 120 and 22*c*.

The simulation studies show that applying the d-extension in combination with FCFS gives results that are comparable to the more complicated scheduling approaches in models 110 and 120. In model 22*c*, however, the DLBA-ud outperforms FCFS-d by 2.4 (28.8%) on irregularity rate, 2.7 containers (14.9%) on the number of waiting containers, and 1 minute (18.3%) on container waiting time.

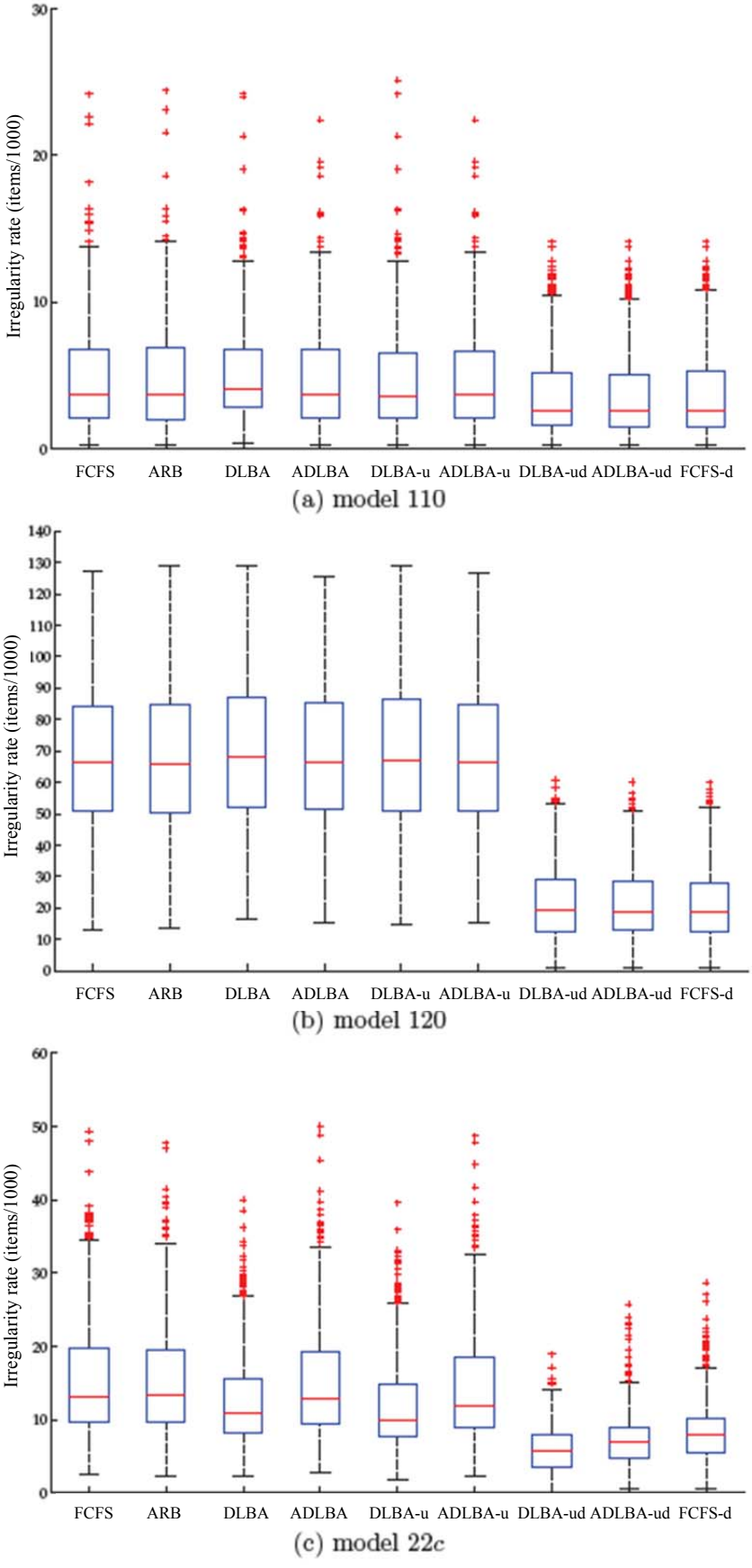


Figure 5.12. Irregularity rate using the BHS-even scenario.

BHS-uneven

In the unevenly distributed baggage handling scenario, the differences between the scheduling approaches become more evident. For models 120 and 22*c*, the DLBA is the best performing approach with differences of 2.0 and 4.6 (3.6 and 29.5%) respectively compared to FCFS. All other PIs show no operationally significant differences except in model 22*c*. There, the DLBA reduces the maximum number of waiting containers and the average container waiting time, compared to FCFS, by 2.3 containers (9.0%) and 2.0 minutes (14.7%) respectively. The ADLBA is the least suitable approach. We observe that the realizations of internal travel times are not in line with the estimations used by the ADLBA, especially for larger baggage handling systems. The highly stochastic environment and the occasionally used storage function make the estimations unreliable.

BHS-uneven with extensions

Including the u-extension does not bring significant improvements. However, the d-extension has a positive effect on the performance of the scheduling approaches. In model 110, the ADLBA-ud is clearly the best approach and outperforms FCFS by 1.7 (19.6%). In models 120 and 22*c*, the DLBA-ud is the best approach, which outperforms FCFS by 33.1 and 4.4 (59.0 and 27.9%) respectively. Not only the irregularity rate, but also the other PIs are affected by the delayability extension. Even with FCFS, the delayability extension is able to reduce the maximum number of waiting containers and the average container waiting time in model 110 by 1.5 containers (17.5%) and 2.2 minutes (32.3%) respectively. The DLBA-ud is the best approach for models 120 and 22*c* as it reduces the maximum number of waiting containers by 4.4 and 1.7 containers (27.3 and 6.9%) respectively, compared to FCFS. In addition, the DLBA-ud reduces the average container waiting time by 2.5 and 2.9 minutes (44.6 and 20.5%) respectively. Furthermore, the simulation studies show that both workload balancing approaches are able to reduce the required EBS space by 97, 213, and 198 items (approximately 40%) for models 110, 120, and 22*c* respectively, by delaying on average 5.6, 12.2, and 12.2 containers.

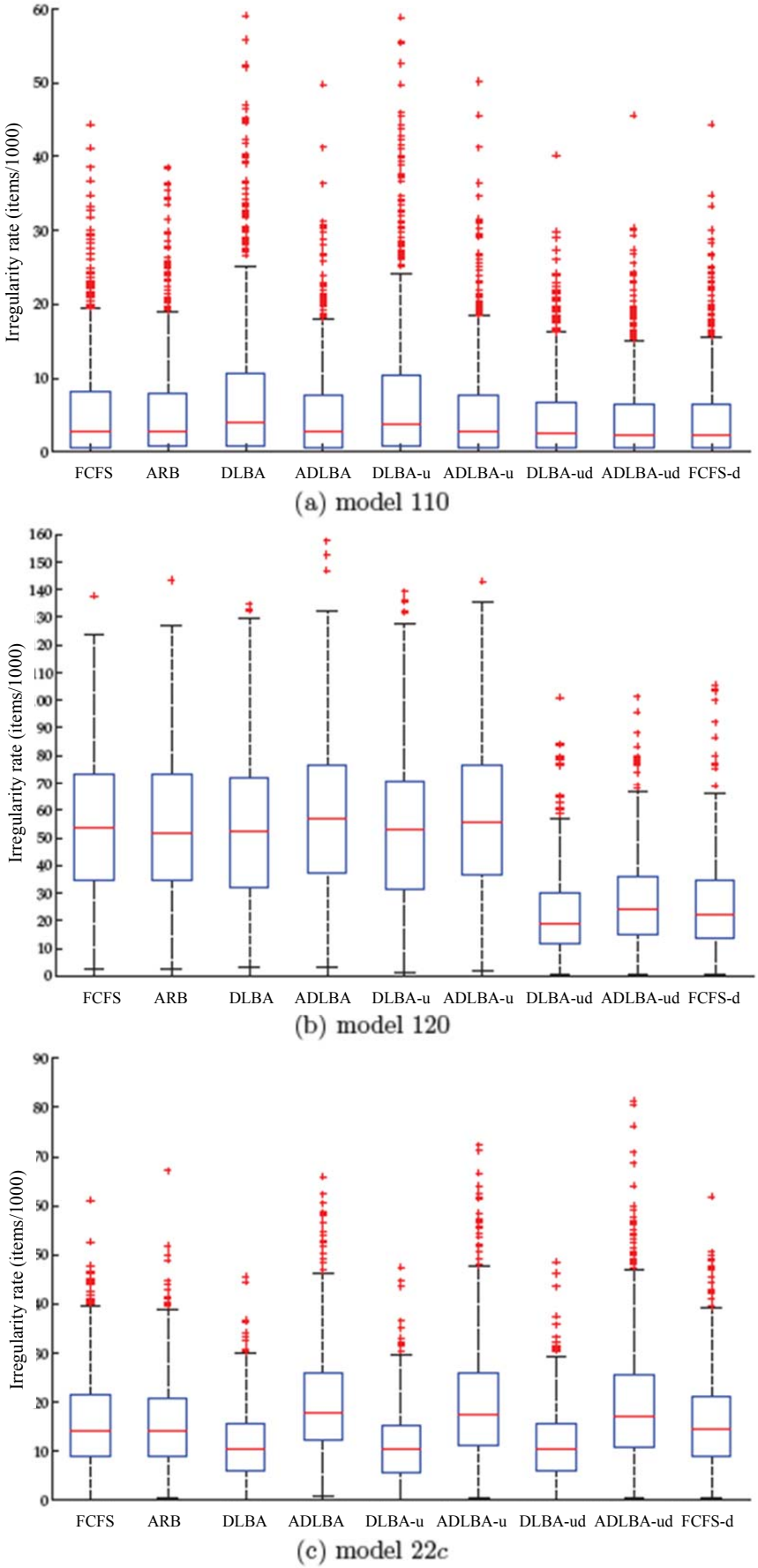


Figure 5.13. Irregularity rate using the BHS-uneven scenario.

5.5.3 Recommended approach per sector, system, and operational scenario

In this section, we construct a matrix (Table 5.2) in which we show the recommended approach for each industrial sector, system model, and operational scenario. This matrix is based on the computational experiments and analysis of the different scheduling algorithms. Moreover, we comment on the results in general.

		System model		
Industrial sector	*Scenario*	*110*	*120*	*22c*
Parcel & postal sorting	*even*	no difference	no difference	ADLBA
	uneven	DLBA	DLBA	DLBA
Baggage handling	*even*	ADLBA-ud	FCFS-d	DLBA-ud
	uneven	ADLBA-ud	DLBA-ud	DLBA-ud

Table 5.2. Recommended approach per sector, system model, and operational scenario.

We observe that the ADLBA uses a detailed approach that performs differently in the different industrial sectors. In baggage handling, the environment is highly stochastic, there is an intermediate storage function, and there are varying plane schedules. These attributes make the estimations of internal travel times less reliable. Therefore, the ADLBA performs better with small sorter systems where the realization of travel times is in line with the estimations.

In baggage handling, the delayability extension is beneficial in reducing the workload on sorters, which in turn helps in reducing the variability of internal transport times. Moreover, using the delayability extension reduces the dependency on the EBS. What happens is that instead of unloading containers carrying early bags and letting these bags proceed to the EBS, we keep these containers unloaded until they are due. Then, when we unload these containers, we have an influence on the outflow figures to outfeeds by the assignment decisions. However, when we release the early bags from the EBS, they result in an uncontrollable flow that creates unpredictable outflows at the destination outfeeds. In other words, the delayability extension helps in replacing the uncontrollable flow from the EBS to destination outfeeds by a controllable flow from the delayed containers to destination outfeeds, which gives us more control over the outflow figures. In the BHS-uneven scenario, for system model 110 in particular, the ADLBA approach is the best performing approach only in combination with the delayability extension.

In parcel & postal sorting, the ADLBA performs better with homogeneous containers and in complex systems. In the more complex systems, the ADLBA logic of balancing the flow over the outfeeds (at different sorters) and over time proves to be beneficial. In this sector, the ADLBA benefits from a more deterministic environment, due to the fixed assignments of destinations to outfeeds, the absence of the storage function, and the single source of inflow, i.e., containers, which we schedule. However, in relatively simple parcel & postal sorter systems, the ADLBA is not likely to contribute much to the performance because in such systems internal travel times are comparable.

In parcel & postal sorting, when containers become more differentiable the DLBA outperforms the other algorithms in all system models. Therefore, we conclude that the differences among the contents of containers overrule the differences among (estimated) travel times on sorters.

5.6 Chapter conclusion

In this chapter, we studied inbound containers scheduling for sorter systems in baggage handling and in parcel & postal sorting. In our analysis, we used the state-of-the-art algorithm (i.e., DLBA) from literature as the first building block and adapted it to incorporate internal travel times on sorters in parcel & postal sorting. Then, we studied the inbound containers scheduling in baggage handling, where we had to incorporate the other sources of inflow from the EBS and check-in desks. As we incorporated the other sources of inflow, we showed how the DLBA and the ADLBA can be applied in baggage handling. However, baggage handling is a sector with special characteristics that have to be considered when scheduling inbound containers. Therefore, we provided two extensions (i.e., urgency and delayability) to the scheduling algorithms in order to improve their performance in baggage handling .

We analyzed the performance of different scheduling algorithms (DLBA, ADLBA, FCFS, and ARB) in the two industrial sectors using different operational scenarios and different system models. Then, we gave advice on which scheduling approach to use for each industrial sector, system model, and operational scenario. Actually, in baggage handling, we found that the delayability extension shows impressive improvements on all performance indicators. Hence, we recommend applying the delayability extension in practice. It is interesting that the delayability extension is applicable as an add-on to current scheduling tools, since we were able to get significant improvements from implementing delayability with the FCFS approach.

In general, we note that in certain cases (at least in the layouts we tested) invoking much detail at the scheduling level by estimating travel times on the sorter, is counterproductive. In fact, the estimated travel times are disrupted by imbalances in travel delays on the sorter, especially when the workload is high. For these cases, we recommend approaches that use less detail at the scheduling level. However, we have to invest in local traffic control rules and algorithms in order to balance the material flow on the sorter. Therefore, in Chapter 6, we analyze the merge operation on conveyor-based sorter systems, since it is the key local traffic problem that affects travel delays on sorters.

Chapter 6

Local Traffic Control In Conveyor Merge Configurations[13]

In Chapter 5, we studied scheduling algorithms for loading sorter systems in baggage handling and parcel & postal sorting. One conclusion was that incorporating too much detail at the scheduling level can be counterproductive in some operational scenarios. In particular, we found that a high workload on sorter systems makes the estimations with regard to the expected travel times on sorters less reliable. In our opinion, the main cause of the unreliability of the estimations is the inconsistency in traffic delays among TSUs (transport stock units) being transported. Therefore, we need to take a closer look at the areas of sorter systems where TSUs face traffic delays.

In this chapter, we analyze an interesting control problem at the local traffic level. This is concerned with the merge area of sorter systems. This area often faces a high workload. As a result, a challenging local traffic control problem arises, which is to allocate empty spaces on the main conveyor to TSUs (on infeeds) waiting to be merged. From a local traffic control point of view, the merge area is the critical area in sorter systems. The other area in these systems is the area where the system diverts TSUs to the pre-assigned outfeeds or chutes.

We develop a generic algorithm that can be applied in merge configurations within different industrial sectors. However, we choose to study the problem in the context of systems in parcel & postal sorting. In this industrial sector, MHSs are basically the sorter systems that have to work under operational conditions that are much more demanding compared to sorter systems in the other sectors (see Section 1.2.2). Therefore, from this point on, we will base our modeling and analysis on systems of parcel & postal sorting.

The structure of this chapter is as follows: First, Section 6.1 introduces the merge configuration in terms of its components, objectives, and the *space allocation problem* (see Section 2.2.3.1). Then, Section 6.2 presents a review of literature that is relevant for the problem at hand. Next, in Section 6.3 we propose a model of the merge area and formulate the space allocation problem mathematically. Thereafter, Section 6.4 describes a generic space allocation algorithm. Section 6.5 deals with the implementation of the generic space allocation algorithm in a simulation environment and provides a performance analysis in different operational settings. Finally, Section 6.6 ends with concluding remarks.

[13] *This chapter is based on Haneyah et al. (2011).*

6.1 The merge configuration

In this section, we sequentially describe the relevant elements and components of the merge configuration (Section 6.1.1) and define its objectives (Section 6.1.2). Then, we formulate the space allocation problem (Section 6.1.3).

6.1.1 Elements of the merge configuration

In this section, we describe the main elements of a merge configuration. Some of these elements are hardware components while others are abstract control points. Figure 6.1 outlines the layout of a merge configuration.

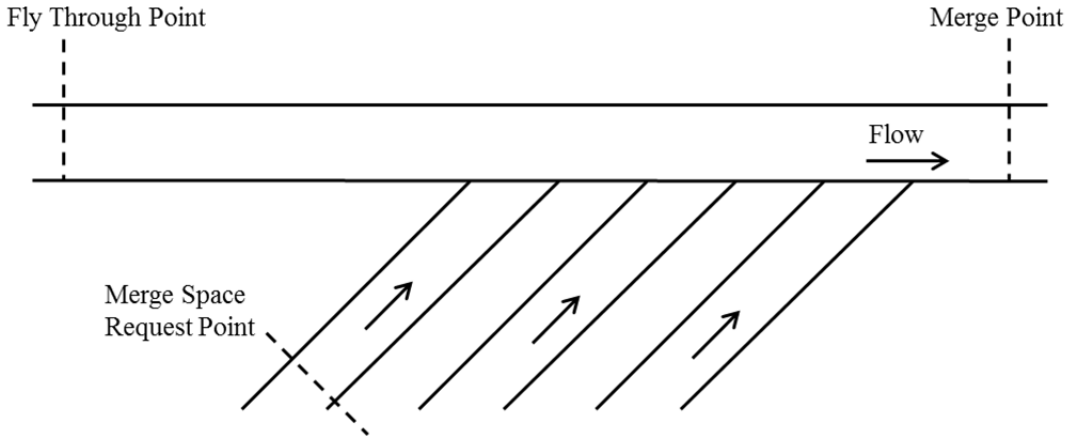


Figure 6.1. Merge configuration.

The merge conveyor

The merge conveyor or the *main* conveyor is represented horizontally in the layout of Figure 6.1. On this conveyor, all parcels coming from the infeeds have to be merged. The merge conveyor is responsible for transporting parcels from the merge area to the sorting area where parcels are sorted to the assigned outfeeds. On the merge conveyor, the direction of the flow is referred to as *downstream* and the opposite direction as *upstream*. The structure of this conveyor can either be a continuous space (e.g., a continuous belt) or consist of discrete space units.

An example of a discrete space merge conveyor is a *tilt-tray* conveyor. Tilt-trays (simply trays) are discrete space units that can accommodate at most one parcel. A parcel occupies at least one tray, but may occupy more trays depending on its length. In the sorting area, the tray(s) transporting a parcel is (are) tilted to allow the parcel to fall into the assigned chute without interfering with other parcels on the merge conveyor.

On a continuous space merge conveyor, a parcel in transport occupies a space called the *merge space*. This space consists of the space the parcel occupies physically and additional free spaces in front and at the back of the parcel that no other parcels can occupy. This free space is known as the *gap*. The *leading gap* is the required free space in front of the parcel; the *trailing gap* is the required free space at the back. Defining the required sizes of the leading gap and the trailing gap for each parcel depends on the dimensions of the parcel and the characteristics of the equipment.

The gaps between parcels that are transported on continuous space conveyors are required mainly due to the later sorting operation. In order to sort these parcels, a hardware component (e.g., a mechanical arm) is used to push them into their chutes. Pushing parcels in this way, while the merge conveyor is moving, makes them slide in a curved path on the merge conveyor which may disrupt the parcel in front if no proper gap is preserved. However, in this chapter we focus mainly on discrete space conveyors, although our results apply to both types. Generally, the merge conveyor runs at a constant speed denoted by v_{Merge}.

Infeed conveyors (infeeds)

Infeeds (the parallel conveyors in Figure 6.1) are responsible for transporting parcels from the point where they are loaded by operators, to the merge conveyor. Parcels are loaded on an infeed in a sequence that is preserved when they are delivered onto the merge conveyor. However, distances between them on the merge conveyor can differ from the distances that were between them on the infeeds. The distance may differ because, in contrast to the merge conveyor, the infeeds do not move at a constant speed, e.g., an infeed can deliver a parcel and then stop and deliver the next parcel later. For most systems, the infeeds in a merge area are of equal length.

There are two possible structures for the infeeds: a continuous belt, and a segmented belt consisting of sections. Figure 6.2 sketches these two variants. For the continuous belt case, any point on the belt has the same speed. For the segmented belt case, the infeed consists of several small conveyor segments (which may run at different speeds) transferring parcels to each other in sequence. However, for both cases there is one special segment (which is always decoupled) at each infeed, indicated by the shaded sections in Figure 6.2. This is the last segment of the infeed at the connection with the merge conveyor. It is always running at a constant speed, so once a parcel arrives at this segment it will be merged immediately on the merge conveyor. The speed of this section is called the downstream speed and denoted by $v_{downstream}$, whose horizontal component is equal to the speed of the merge conveyor $v_{Merge} = v_{downstream} \cdot \cos\alpha$, where α $(0 < \alpha < \frac{\pi}{2})$ is the angle between the merge conveyor and the infeed (see Figure 6.2).

Parcels on the infeed have certain distances between them. These distances have to be taken into account as they can restrict the possible locations of parcels when merged onto the merge conveyor. If the infeed conveyor is a continuous belt conveyor, then the distance between any two parcels remains constant as long as they are on the belt. All points on the belt have the same speed, so when the belt is stopped all parcels on it stop, and when it moves all parcels on it move at the same speed. In this case, if a parcel (parcel 1) from an infeed is merged, it may not be possible to merge the next parcel (parcel 2) directly next to it on the merge conveyor. Parcel 2 has to travel until it reaches the last segment of the infeed to be merged as well. While it moves to reach the last segment, parcel 1 also moves on the merge conveyor. In addition, note that the last segment can only receive parcel 2 after it has delivered parcel 1 to the merge conveyor. As a consequence, the distances between parcels on the infeeds result in certain minimum distances between the parcels when merged on the merge conveyor. These distances are called *follow-up* distances.

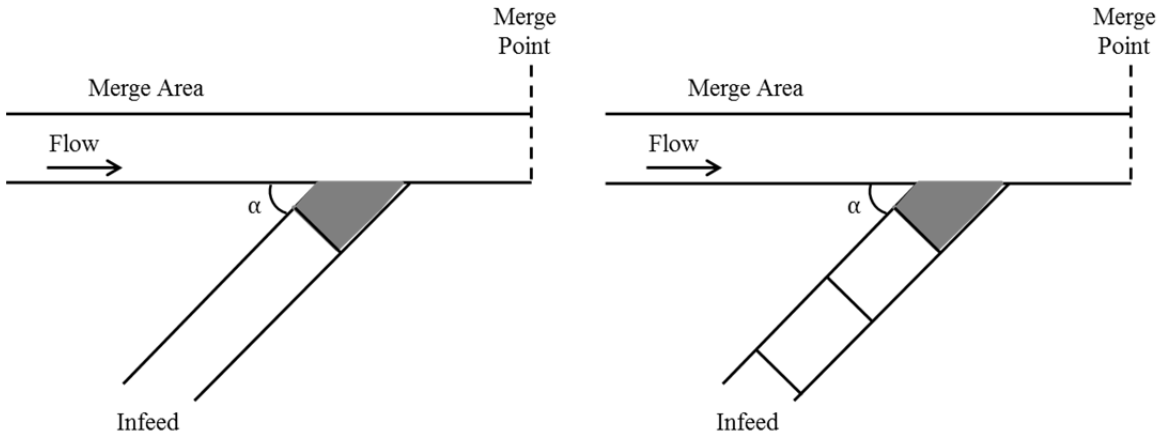


Figure 6.2. Continuous versus segmented infeed conveyors.

Merge space request point

This is the point at which the parcel is placed on the infeed. It is also referred to as the *announcement point.* Sensors detect the placement of the parcel on the infeed and then the parcel is *announced* in the system. In general, at the time the parcel is announced, it requests a merge space at the merge conveyor. The request consists of two elements. The first is the time at which the parcel is expected to arrive at the *merge infeed point*, which is the point where the infeed and the merge conveyor meet. This time is called the delivery time and it is calculated based on the highest possible speed profile to deliver the parcel. Second, the required merge space is announced. If the merge conveyor is a tilt-tray conveyor, then the merge space is represented by the number of trays needed.

Fly through point

This is the starting point of the merge area. The status of the merge conveyor upstream this point is unknown for the local traffic controller of the merge area. The fly through point is the point where circulating parcels re-enter the merge area. These are parcels that have not been sorted in the sorting area. They are unpredictable and uncontrollable by the controller of the merge area. The configuration and locations of these parcels cannot be changed. The parcels passing into the merge area at this point are called *flying through* parcels.

Merge point

This is the end point of the merge area, downstream all merge infeed points (see Figure 6.1). At this point, the resulting parcels structure built by the space allocation algorithm within the merge area can be observed.

6.1.2 Merge configuration objectives

In this section, we introduce the objectives of the merge configuration and explain the relevance of each of them to the performance of the sorter system as a whole. There are two main objectives within the merge configuration. The first one is throughput maximization (Section 6.1.2.1), which follows directly from the overall system objective (see Chapters 1 and 5). The second is workload balancing, which does not

have a direct relation to the objective of the sorter system as a whole. In Section 6.1.2.2, we explain why this objective is nevertheless important.

Before going through these objectives, we note that the objectives are to be achieved by the space allocation algorithm, which is the underlying logic for the local traffic controller that is responsible for allocating the space on the merge conveyor to the parcels from the infeeds. The algorithm uses the available information in the merge area. The input information for the algorithm is: first, available spaces on the merge conveyor, i.e., *space windows*, and second, parcels announced on the infeeds.

6.1.2.1 Throughput

Throughput of the merge area is related to space utilization on the merge conveyor. In other words, it refers to the extent to which the space on the merge conveyor is occupied by parcels after leaving the merge area, i.e., when passing the merge point. In this context, it is required to merge as many parcels as possible and fill the space windows between flying through parcels.

6.1.2.2 Workload balancing

Workload balancing corresponds to the infeed conveyors. It is important to keep a balance in the delivery of parcels among all infeeds. The case where one infeed fills all possible spaces while another infeed struggles to get a space for its parcels should be avoided. Two main reasons make this objective interesting. First, there are operators that load parcels on each infeed. The distribution of work among these operators should be fair. Second, when distinct batches are placed at the beginning of each infeed, all these batches have to be served. For this reason, balancing the workload among infeeds makes it possible for all batches to be served in reasonable time. In fact, this objective is concerned with labor efficiency as incorporated in the MHSs' requirements model (Section 1.3.2).

In current practice, some MHSs' suppliers set a space utilization limit for infeeds. For example, in a merge system with two infeeds, each infeed can use at most 50% of the available space on the merge conveyor. This strategy makes sense when both infeeds are working with similar loads. However, assume that infeed 1 has one parcel and that infeed 2 has a long queue; then the space utilization limit on infeed 2 forbids it from delivering more parcels, while there are still free space windows that are not used by infeed 1. This leads to starvation (see Section 1.3.2), which is a phenomenon that we should avoid in the control architecture.

The main KPI of the sorter system is high throughput, and so workload balancing is a secondary objective. Workload balancing is not a goal on its own, but it has to do with the system functionality, e.g., operators' task division. Moreover, balancing waiting times for parcels on infeeds may have a positive effect on higher level scheduling approaches. For example, the estimations of travel times used by the ADLBA (see Chapter 5) become more reliable when traffic delays are evenly distributed among parcels being transported.

6.2 Theoretical context and key literature

Theoretical context

We may classify space allocation in the merge area of the sorter system as a real-time scheduling problem. In real-time scheduling, schedules have to be created with incomplete information about the system, which also applies to online scheduling. However, the main difference is that the time available to create schedules is very limited in real-time scheduling, much more than in online scheduling. In sorter systems, information about incoming parcels and incoming space windows is incomplete. Moreover, allocation decisions have to be made in few milliseconds. Therefore, we focus the literature review of this section on real-time scheduling since the time available for computations does not allow us to use complex online scheduling algorithms.

Lu et al. (1999) classify real-time scheduling algorithms into two categories: *static* and *dynamic* scheduling. In static scheduling, the scheduling algorithm has complete knowledge about the tasks to be scheduled, their properties, and all constraints. In dynamic scheduling, the scheduling algorithm does not have complete knowledge about tasks and their properties. In this case, future new tasks are arriving, which the algorithm was not aware of while scheduling the previous set of tasks. Lu et al. (1999) further divide dynamic scheduling to be working in either *resource sufficient* or *resource insufficient* environments. Resource sufficient environments are environments in which there are sufficient resources to serve all arriving tasks, even if they arrive dynamically. System designers try to ensure that the system has sufficient resources. However, due to highly uncertain environments, it is sometimes impossible to serve all tasks, which yields a resource insufficient situation. Moreover, in terms of schedules' adjustability, scheduling algorithms can be either *closed-loop* or *open-loop.* Closed-loop algorithms are those that can be adjusted based on continuous feedback. Similarly, open-loop scheduling algorithms refer to the fact that once schedules are created, it is not possible to modify them based on incoming feedback (Lu et al., 1999).

We classify the space allocation problem in merge configurations as a dynamic, resource sufficient, and closed-loop scheduling problem.

Key literature

There is an extensive quantity of papers discussing real-time systems from a software implementation point of view. Koster and Wijnen (1998) describe the merge problem and study several simple control rules using simulation and under relatively simplistic assumptions. Their study is very useful for understanding the problem at hand and for different control rules. However, Koster and Wijnen (1998) do not propose a real-time algorithm that adapts to the operational changes and works beyond the assumptions the present. Audsley and Burns (1990) review the application of scheduling theory to dependable real-time systems. Their review takes the form of an analysis of the problems presented by different application requirements and characteristics. They cover issues such as uniprocessor and multiprocessor systems, periodic and aperiodic processes, static and dynamic algorithms, transient overloads, and resource usage.

Stankovic et al. (2001) discuss scheduling in distributed real-time systems. They develop a distributed real-time scheduling approach based on an analytical model, with feedback control design techniques. The emphasis they place on software architectures is not within the focus of this chapter. Other studies in this context are Lu et al. (2002) and Stankovic et al. (1999), to list a few.

Ramamritham and Stankovic (1994) classify scheduling algorithms and operating systems support for real-time systems in four paradigms:

- *Static table-driven approaches:* these approaches create static schedules (offline) in order to execute a set of predictable tasks. The resulting schedule, which is often in the form of a table, is used to dispatch tasks during execution.
- *Static priority-driven preemptive approaches:* these approaches are similar to the previous category in creating static schedules. However, a difference is that tasks have priorities that may be revealed dynamically. In this case, during execution, high priority tasks are handled first and may preempt low priority tasks. This paradigm, in its use of priorities, can be a starting point to the space allocation problem at hand.
- *Dynamic planning-based approaches:* in these approaches, there are tasks arriving during execution. When a new task arrives the scheduling approach attempts to schedule it while keeping the schedules of earlier tasks intact. If the attempt fails then new schedules are created. The feasibility of schedules continuously checked online. One of the results of the feasibility analysis is to decide when a task can be executed.
- *Dynamic best effort approaches:* in these approaches, there are no feasibility checks. The scheduling approach tries to meet the tasks' deadlines. However, there is no guarantee that all tasks can be processed. Therefore, some tasks may be canceled.

Regarding conveyor systems, most of the studies we reviewed were simulation based, and scheduling was not evident beyond simple rules or policies. Nazzal and El-Nasher (2007) state that all of the studies have been simulation-based, application-specific, and cannot be generalized. An example is a simulation modeling study, based on queuing theory, by Jing et al. (1998) on a conveyor merge configuration. They model a merge configuration with the objectives of high throughput on the merge conveyor and minimizing imbalance among infeeds. They try different control rules for assigning space windows to parcels: FCFS, longest queue first, highest priority first, random, natural (no control), and cyclic. However, no discussion of the algorithms or the application of these general rules is involved.

Other studies focus on the mechanical properties and the physical forces that are involved in the merge operation, e.g., speeds, friction, layout angles, where the aim is to optimize the technical details of the merge operation, usually for a single infeed. Landschützer et al. (2013) and Jodin and Wolfschluckner (2010) provide mathematical models for these problems, where the higher level space allocation algorithms are not part of these studies.

Bozer and Hsieh (2005) analyze a closed-loop conveyor system, which consists of machines, loading and unloading stations, and jobs to be processed. They follow an

analytical approach based on queuing theory. They first derive the stability condition for the conveyor system and then find the layout of the loading and unloading stations around the conveyor loop. Next, they study the tradeoffs between the objectives of high throughput and low work-in-process. Their study uses conveyors as a transport mode while the focus is on the other system elements. Moreover, there are no merge configurations involved.

Schmidt and Jackman (2000) claim that no analytical models for closed-loop conveyors were available and present an analytical model for such systems. They show that the results of their analytical model are similar to simulation results. However, they focus on manufacturing environments and jobs to be processed. Other analytical approaches are found in the modeling of closed-loop conveyors with load recirculation by Hsieh and Bozer (2005).

Many studies on merge configurations focus on systems that are more analogous to railways than to conveyors. For example, Shladover (1980) analyzes the operation of merge junctions in dynamically entrained automated guide-way transit systems. The author analyzes a merge junction with two input lanes and one output lane. The aim of this study is to provide a realistic estimate of the lane capacity that can be used when merge restrictions are involved. Using simulation, the author analyzes the effects of varying inputs on the capacity usage of the output lane. He distinguishes between *isocapacity* and *concatenating* merges. The first refers to merging carts on one railway after a junction, while the latter refers to combining two separate carts at the merge point into one connected cart. The relevance and application of traffic theory is evident in his study, which falls in a context different from merge configurations in MHSs.

Conclusion

We find that studies focusing on the software development aspect in real-time systems rather than scheduling algorithms are not of much use in this thesis. Moreover, studies that focus on analytical approaches are not relevant to the space allocation problem as we consider a real-time system, for which we are interested in an algorithm that can make allocation decisions in real-time, given different operating conditions. An analytical approach is a rigid approach that cannot handle the dynamic nature of the problem and cannot update decisions and system performance measures online. Moreover, it is not applicable to several merge system configurations. As a conclusion, we have not found a specific study focusing on real-time space allocation algorithms. Moreover, we could not find studies discussing the control and scheduling of conveyors in MHSs. The study of conveyors is limited to their role as a support element within a manufacturing environment.

6.3 Problem formulation

In this section, we formulate the space allocation problem in more concrete terms. Section 6.3.1 deals with modeling aspects of the merge area. Then, Section 6.3.2 formulates the objectives in a quantitative manner. Finally, Section 6.3.3 presents a static approach for space allocation in the merge area.

6.3.1 Modeling the merge area

In this section, we list the main modeling points and constraints for space allocation in the merge area. Figure 6.3 presents a schematic overview of the merge area.

- The boundary of the merge area is the fly through point upstream and the merge point downstream. The status of the merge conveyor upstream the fly through point is unknown and the status downstream the merge point is of no further interest.

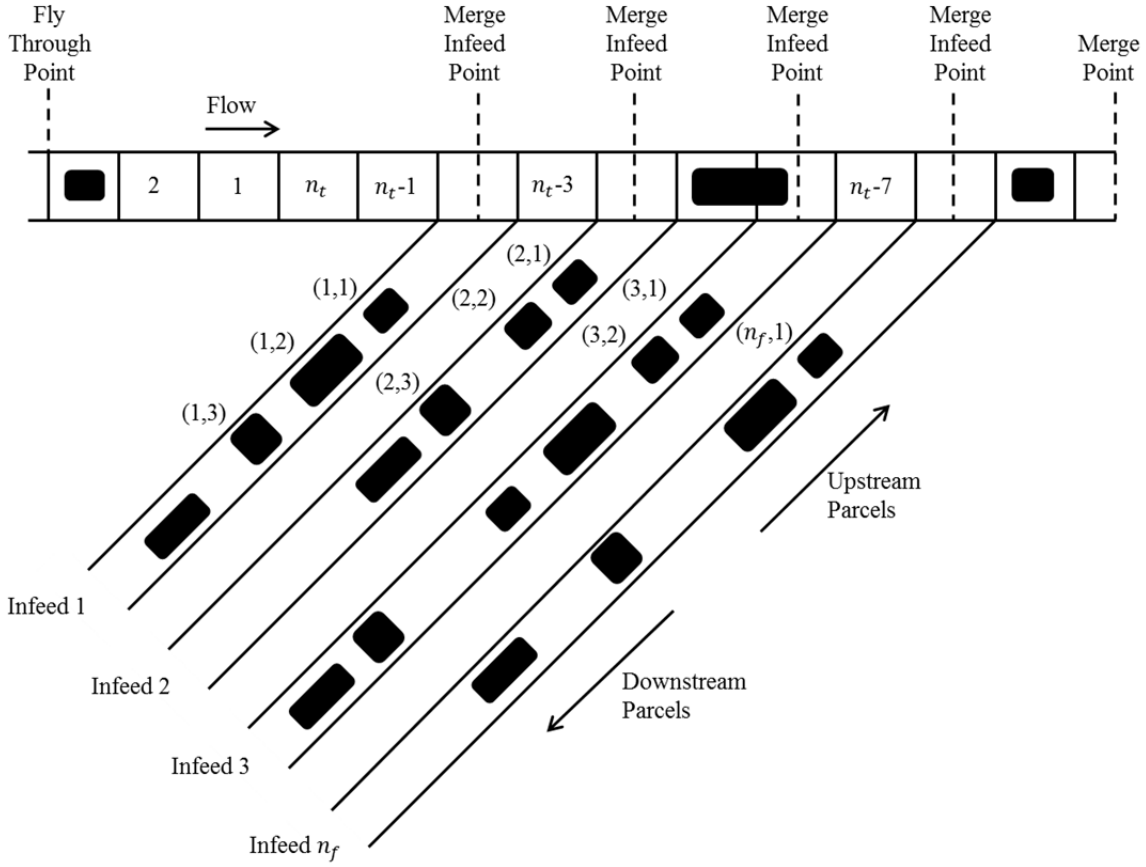


Figure 6.3. Modeling the merge area.

- The merge conveyor is a closed-loop conveyor that consists of a finite number of trays $T = \{1, .., n_t\}$ that are numbered in order of their occurrence (ascending upstream).
- The subset $O \subset T$ specifies which trays are already occupied by flying through parcels. These trays are unavailable for allocation.
- There is a finite set of infeeds $F = \{1, .., n_f\}$. For each infeed $f \in F$, a sequence of parcels is given. Let P be the set of all parcels.
- Parcels on an infeed $f \in F$ are denoted by tuples (f, p). The first entry refers to the infeed where they are located, and the second refers to their sequence number on the infeed. The positions are numbered in an ascending order starting from the most downstream parcel (see Figure 6.3).

- Each parcel (f,p) requires a number of trays based on its length. This number is denoted by $l_{f,p}$. Each parcel (f,p) on an infeed needs to be allocated to $l_{f,p}$ consecutive trays on the merge conveyor (at least one). Therefore, the output of the space allocation algorithm is given by an allocation of the parcels from the infeeds to trays on the merge conveyor.
- The set of consecutive trays assigned to a parcel (f,p) cannot be assigned to any other parcel.
- For parcel (f,p), a follow-up distance $FUD_{f,p}$ is given, which denotes the minimum number of trays possible between the most upstream tray (last tray) to which parcel (f,p) is allocated and the most downstream tray (first tray) to which parcel $(f,p+1)$ is allocated. Follow-up distances result from the distances between parcels on the infeeds. The distances are taken as an input in our model. This constraint also preserves the sequence of parcels on the infeeds.
- Each parcel (f,p) has an expected delivery time to the merge conveyor. This is expressed by the expected delivery tray $d_{f,p}$. The expected delivery tray is calculated as the first tray on the merge conveyor where the parcel can be delivered using the highest possible speed of the infeed. The calculations of delivery trays respect downstream parcels (on the same infeed) by accounting for their possible delivery trays and follow-up distances between parcels. However, parcels on the merge conveyor and parcels on other infeeds are not incorporated in this measure.

6.3.2 Objectives formulation

The output of a space allocation algorithm is given by an allocation of the parcels from the infeeds to trays on the merge conveyor. This allocation has the objectives of maximizing throughput at the merge point and balancing workload among infeeds (see Section 6.1.2).

Maximizing throughput requires that the space on the merge conveyor is utilized as much as possible as it passes the merge point. In this context, assume that we have an allocation for a set of parcels (from infeeds) on the merge conveyor. Then, measuring the total number of empty trays from the first assigned tray until the last assigned tray provides a representative measure of low utilization. We can define this number as the number of empty trays in $\{1,..,L\}$, where L is the last tray a parcel is allocated to, given an allocation of a set of parcels on the merge conveyor.

With regard to workload balancing, all infeeds should be treated fairly in terms of waiting times. In other words, if there are not enough space windows on the merge conveyor to serve the requests from the infeeds, then waiting time on infeeds is inevitable. Waiting time occurs when an infeed has to wait for a space window on the merge conveyor to arrive at the merge infeed point (see Figure 6.3), by either slowing down or completely stopping. When an infeed slows down or stops, the parcels on it are not delivered to their first possible delivery trays identified by $d_{f,p}$, but to later trays. In this case, the waiting time for a parcel (f,p) can be represented by the difference between the first tray actually allocated to the parcel and the first possible

delivery tray $d_{f,p}$. This difference is referred to as the *delay* in the delivery of the parcel.

6.3.3 Static modeling for a formal problem description

In this section, we consider a static view of the space allocation problem in the merge configuration. This means that we take at some moment in time a sort of snapshot of the system and consider the problem of allocating all parcels visible at that moment. This static problem may be used in an iterative approach. In each iteration, the static approach deals with a set of parcels on the infeeds and a set of available space windows on the merge conveyor. The result is an allocation of all parcels in the given data set. However, applying the static approach iteratively may result in an overlap between two different sets of parcels or space windows, which may require changing some allocation decisions.

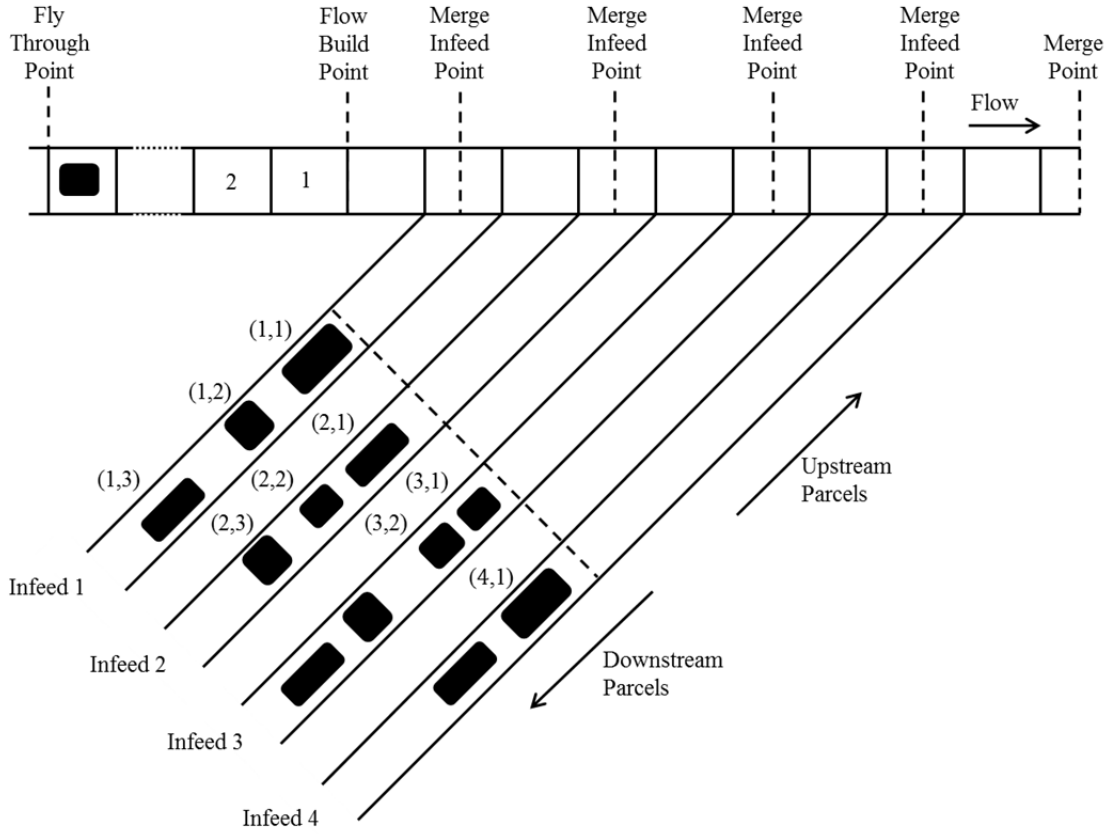


Figure 6.4. Static view of the merge area.

The static approach uses a planning horizon that includes a finite set of parcels on each infeed and a part of the merge conveyor to allocate these parcels on. As an input, the static approach has deterministic data about the parcels that need to be allocated, and space windows available on the merge conveyor. We call the set of parcels in a specific planning horizon a *batch*. The static approach calculates an allocation for the batch of parcels within the planning horizon at once. Figure 6.4 shows a static view of the merge area. The following points explain the static approach:

- We introduce the *flow build point* as the starting point for space allocation of a batch on the merge conveyor, i.e., upstream this point all trays $t \notin O$ are open for allocating parcels of the batch.
- The tray at the flow build point is given the index 1.
- For each infeed $f \in F$ a *finite* sequence of parcels $(f,1)..(f,m_f)$ is given.
- A position upstream the merge infeed point of infeed 1 is chosen for the flow build point. The position of the flow build point leads to a virtual line that crosses through the infeeds (see Figure 6.4). Parcels upstream this line arrive at the flow build point (tray 1) or any tray upstream of it, when delivered at the highest possible speed.
- The area between the flow build point and the fly through point is the part of the merge conveyor where the batch of parcels has to be allocated. We assume that this part of the merge conveyor is long enough to accommodate all parcels in the batch.

The static approach leads to a deterministic optimization problem. Such a problem is often modeled using mathematical programming. A mathematical program has a set of formally written constraints and an objective function to optimize. All input data to the program has to be given before solving. We now present an ILP (integer-linear programming) model for the problem of finding a feasible allocation of parcels with respect to a given objective function.

Decision variables

The model makes decisions for parcels' allocations. In order to know the trays allocated to parcel (f,p) it is enough to know the first tray allocated to the parcel. The other trays occupied directly result from the parameter $l_{f,p}$. The output of the ILP is an assignment of the first occupied tray for each parcel. This can be achieved by binary assignment variables as follows:

$$X_{t,f,p} = \begin{cases} 1 & \text{if tray t is assigned as the first tray for parcel } (f,p) \\ 0 & \text{otherwise} \end{cases}$$

Auxiliary variables

$W_{f,p}$: waiting time of parcel (f,p) measured in trays.

$AvgW_f$: Average waiting time on infeed f measured in trays.

$TotW_f$: Total waiting time on infeed f measured in trays.

Constraints for a feasible allocation

1. *First tray assignment:* each parcel from the infeeds needs to be allocated. Therefore, each parcel needs to be assigned to a first tray. However, the first tray cannot be a tray downstream the first possible delivery tray. Mathematically:
   $$\sum_{t \geq d_{f,p}} X_{t,f,p} = 1 \quad \forall (f,p) \in P$$
   $$X_{t,f,p} = 0 \quad \forall (f,p) \in P, t < d_{f,p}$$
2. *Overlap prevention:* no tray can be used by more than one parcel. Trays of the set O are occupied by flying through parcels and cannot be assigned to any

parcel from the infeeds. The following constraints prevent the usage of any tray by more than one parcel.

$$\sum_{f,p} \sum_{t'=t-l_{f,p}+1}^{t} X_{t',f,p} \leq 1 \quad \forall t \notin O$$

$$\sum_{f,p} \sum_{t'=t-l_{f,p}+1}^{t} X_{t',f,p} = 0 \quad \forall t \in O$$

3. *Follow-up constraints:* Follow-up distances have to be respected for two successive parcels (f,p), $(f,p+1)$ from infeed f.

$$\sum_t t \cdot X_{t,f,p+1} \geq \sum_t t \cdot X_{t,f,p} + l_{f,p} + FUD_{f,p} \quad \forall (f,p) \in P, p < m_f$$

ILP objective formulation

For the static case, total waiting time and average waiting time on the infeeds are both important since we want to have the work balanced in terms of total waiting and the average waiting time of parcels on the infeeds. Therefore, both measures need to be incorporated in the objective function of the ILP model.

The waiting time for parcel (f,p) is specified as the difference between the first tray actually assigned to parcel (f,p) and the first possible delivery tray $d_{f,p}$. Mathematically:

$$W_{f,p} = \sum_t t \cdot X_{t,f,p} - d_{f,p} \quad \forall (f,p) \in P$$

Given the waiting times for parcels, the average waiting time for infeed f can be calculated as:

$$AvgW_f = \frac{\sum_p W_{f,p}}{m_f} \quad \forall f \in F$$

Note that waiting time has a cumulative effect, e.g., if a parcel (from a batch of successive parcels on an infeed) is delayed by one tray, then all parcels upstream are delayed by one tray as well. Therefore, for a static problem with a pre-defined set of parcels, the *total waiting time* on an infeed is determined by the delay of the last parcel (f,m_f) on the infeed within the planning horizon, which is:

$$TotW_f = \sum_t t \cdot X_{t,f,m_f} - d_{f,m_f} \quad \forall f \in F$$

However, for systems' users, it is not always sufficient to balance the total waiting time. Figure 6.5 shows two parallel infeeds; infeed 1 with a low load of parcels and infeed 2 with a high load. The aforementioned total waiting time formulation overlooks the differences in the sizes of the parcels, and hence the different loads on the infeeds (see Figure 6.5). In fact, systems' users may prefer to give advantage to the highly loaded infeeds. To account for such cases, we may multiply the total waiting time of the infeed by the total load on the infeed. Mathematically, this yields:

$$TotW_f' = \left(\sum_{p=1}^{m_f} l_{f,p}\right) \cdot \left(\sum_t t \cdot X_{t,f,m_f} - d_{f,m_f}\right) \quad \forall f \in F$$

The total and the average waiting times of an infeed should be minimized simultaneously. However, we should be aware that the total waiting time has always a greater numerical value than the average waiting time, because it represents the total delay (and possibly the load on the infeeds). In order to make the two measures comparable, we multiply the total waiting time by the parameter δ $(0 \leq \delta \leq 1)$. However, determining a good value for δ may depend on the instance given.

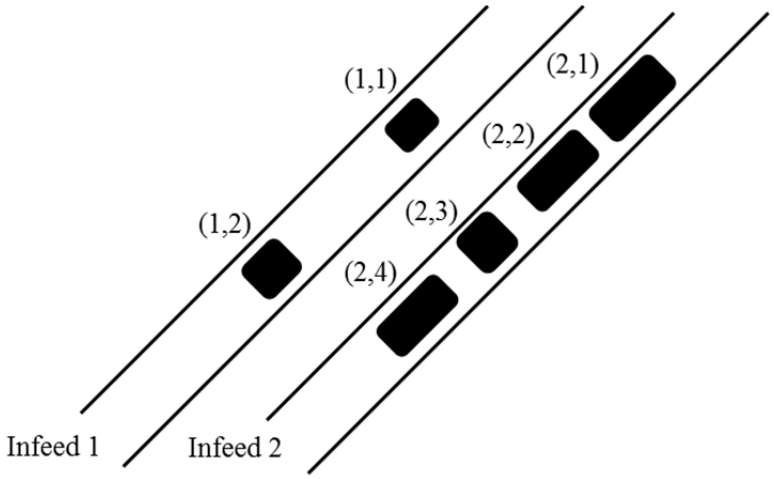


Figure 6.5. Highly loaded versus lowly loaded infeeds.

In a qualitative sense, the objective of the ILP is to distribute waiting times evenly among infeeds. In a quantitative sense, we may formulate this problem as a *MinMax* problem, e.g., to minimize the maximum waiting time. In a MinMax problem, a global value that represents the maximum measure in the problem is introduced in the model, and the objective is then to minimize this value. In this specific problem, we introduce the value *imbalance* as the value to be minimized in the objective function. In order to account for the objectives of total waiting time and average waiting time simultaneously, we propose the following formulation to characterize the imbalance:

$$\delta \cdot TotW_f + AvgW_f \leq imbalance \quad \forall f \in F$$

The objective function of the ILP model now becomes:

$$Min\ Z = imbalance$$

In this formulation, the infeed with the largest waiting time determines the imbalance value. Therefore, in order to minimize this value, the model tends to make waiting times on the different infeeds as equal as possible, which is what we want to achieve.

This objective function is concerned with workload balancing. However, the average waiting time and the total waiting time of an infeed are variables that depend on waiting times of the parcels. In turn, the waiting times of parcels are directly linked to the utilization of trays on the merge conveyor. Therefore, minimizing the imbalance variable *tends to* minimize the number of empty trays in $\{1, .., L\}$, where L is the last tray a parcel is allocated to, given an allocation of a set of parcels on the merge conveyor.

The reason is that a deterioration in throughput that occurs by leaving empty trays in the solution, directly causes a deterioration in parcels' waiting times, because empty trays increase waiting times for parcels. If one tray is left empty, then the waiting times of all parcels allocated upstream of it may increase by one. However, the model does not grant maximizing throughput when minimizing the imbalance variable. In some cases, minimizing the imbalance variable may impose leaving out some empty trays. Therefore, the objective of minimizing the imbalance variable acts as a sufficient approximate measure to realize the objectives of throughput and workload balancing.

ILP model implementation

Let us use a small test instance, which consists of sequences of in total 16 parcels on 4 infeeds, and a merge conveyor with 4 flying through parcels. For this test instance, we solve the ILP problem using special-purpose software (GAMS) that uses the CPLEX solver. In order to select a value for δ, we solve the ILP problem with the sum of the total waiting times (including the loads on the infeeds) and the average waiting times as the objective. The result shows that the value of total waiting time is approximately ten times the value of the average waiting time. Therefore, we use $\delta = 0.1$ to make total waiting time and average waiting time comparable. Further experimentation with other values of δ showed that $\delta = 0.1$ indeed gives good results for this test instance. The resulting allocation shows that the space is 100% utilized (i.e., no empty trays are present). On the other hand, the average waiting time for the infeeds ranges between 8.7 and 9.5 trays, while the total waiting time ranges between 80 and 85. These results show that the workload is balanced.

The running time for this test instance is *32.23 seconds* using CPLEX, which is a powerful commercial solver, on a workstation with an Intel® Core™2 Duo CPU T9300 @ 2,50GHz processor and 4.00 GB RAM. This running time is certainly unacceptable for the sorter system. Furthermore, the test instance was for a small merge area with a small number of parcels to allocate (16 parcels) and only 4 infeeds. Therefore, we expect the running time of an ILP problem for a large merge area that may have up to eight infeeds to be much longer. In the merge area of the sorter system, allocation results have to be retrieved in milliseconds, and so finding an optimal solution is not an option. Although the ILP can be interrupted and a solution can be retrieved, there is no guarantee that a feasible and good solution is ready in less than a second. Moreover, the static approach is not applicable to different layouts and possibly different objectives of the merge area. Finally, the static approach assumes that the part of the merge conveyor within the planning horizon is long enough to accommodate all parcels in the batch. This assumption does not hold in most of the existing configurations of the merge area as we will describe in Section 6.4. Parcels to be allocated are also not all known in batches as assumed. These points result in unreliable and incomplete input data for the ILP model, which makes a dynamic allocation approach more reasonable.

6.4 A dynamic space allocation approach

The dynamic approach works with information revealed in real-time. The dynamic approach deals with a narrower view of the merge area compared to the static approach (Section 6.3.3). Therefore, we are not aware of a large batch on parcels on the infeeds. Moreover, we are not aware of the availability of space on the merge conveyor. In this approach, we investigate one space window at a time. For each space window, we look for candidate parcels from infeeds. A candidate parcel is a parcel that fits in the space window and can be delivered to the space window. Hence, an iteration in the dynamic approach is concerned with allocating an available space window and only considers the most downstream unallocated parcel from each infeed.

6.4.1 A priority-based algorithm (PBA)

In this section, we embody the dynamic approach in the context of a *priority-based algorithm (PBA)* for the basic configuration of the merge area (Section 6.3.1). The PBA uses real-time information when making allocation decisions. Information at the fly through point specifies available space windows on the merge conveyor, where a space window consists of a number of consecutive empty trays that can be allocated to parcels from the infeeds. Thereafter, infeeds having candidate parcels, which can be allocated to incoming space windows, are identified. We denote the set of candidate parcels by C ($C \subset P$).

An iteration of the PBA is executed every time a space window is available. After each iteration, the algorithm updates the available space windows. The length of a space window (sw) is given by a number of empty trays. Each sw is measured by counting the number of empty trays appearing at the fly through point until one of the following two cases occurs:

1. A tray occupied by a flying through parcel is reached.
2. The number of trays required by the parcel ($(f,p) \in C$) of largest size is reached.

The second case avoids counting large numbers when the next tray occupied by a flying through parcel is far upstream (if there are flying through parcels at all). Moreover, once the number of trays required by a parcel of the largest size is reached, further counting has no added value in the calculations that follow.

Parcels (f,p) that are candidates to be allocated in the given space window of length sw must satisfy the following conditions:

1. Fit in the available space window ($l_{f,p} \leq sw$).
2. Their downstream parcels have been already allocated.
3. They can arrive at the first tray of the available space window at the merge infeed point.

Therefore, at most one candidate from each infeed can be in the set C of candidates, which is the most downstream unallocated parcel of that infeed. If the set C contains more than one candidate, a priority is calculated for each candidate parcel, and the algorithm allocates the available space to the candidate parcel with the highest priority. The method to calculate priorities is critical. First, the priority calculation for a parcel $(f,p) \in C$ is relative to the other candidate parcels. Second, the priority of a parcel $(f,p) \in C$ depends on the contribution of its infeed f to the workload balancing objective and the contribution of the candidate parcel to the throughput objective. To capture both objectives in the priority calculations, the formula to calculate the priority gives weight to each of the two objectives. It is of the form:

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

where α is a weighing parameter. In this way, we give attention to both objectives. The default value of α has to be tuned by simulation.

The throughput measure depends on the candidate parcel itself. We use the extent to which a candidate parcel can occupy an available space window on the merge conveyor as a measure. This measure gives priority to large parcels, which is desirable in practice because delaying the delivery of a large parcel creates a risk of not finding another space window that can accommodate this parcel for a long time. The balance measure of a parcel (f,p) depends only on the infeed f transporting the candidate parcel. We use the total (accumulating) waiting time of the infeed to calculate the balance measure. Let $F' = \{f \in F | \exists p: (f,p) \in C\}$, then the proposed measures are as follows:

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

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

It remains to explain how we calculate $TotW_f$. The basic idea is that the total waiting time of an infeed is calculated as the sum of waiting times for all parcels on an infeed, starting from the first parcel that has been allocated up to the most downstream unallocated parcel. The waiting time for a parcel (f,p), i.e., $W_{f,p}$, is calculated as the number of trays between the tray actually assigned to the parcel and the first possible delivery tray of the parcel. Clearly, the first possible delivery tray of a parcel is based only on the allocation of its downstream parcel. In other words, the first possible delivery tray of parcel (f,p) is the tray which is $FUD_{f,p-1}$ trays after the most upstream tray allocated to the downstream parcel $(f,p-1)$. However, if there is no downstream parcel that is restricting delivery, then the first possible delivery tray is the first tray reachable using the highest possible speed on the infeed. With regard to the most downstream unallocated parcel, we do not know the tray assigned to it. Therefore, instead of its assigned tray, which is unknown, we use the most downstream unallocated tray that can be allocated to this parcel to calculate a lower bound for its waiting time. If no such tray is found downstream the fly through point, then we use the tray at the fly through point. In this manner, we can trace the increase in waiting time as long as a parcel is waiting and is unallocated, given that the possible delivery tray is not upstream the fly through point.

The calculation of total waiting time for an infeed is then represented as follows:

$$TotW_f = \sum_p W_{f,p}$$

The PBA is a suitable option for a constructive heuristic for space allocation. It is flexible as it can handle any number of infeeds or appearance of space windows on the merge conveyor in the same generic steps. In addition, merge configurations in other operational environments or with different layouts may be handled, either by changing the method to calculate the priorities or by incorporating additional measures in the priority calculations. For example, in baggage handling, the priority may depend on the urgency of a bag.

The PBA is based on the basic configuration of the merge area, which we discussed so far. However, in practice there are certain layout restrictions of the merge area, which make space allocation more challenging. Section 6.4.2 explains these restrictions.

6.4.2 Layout restrictions and the early reservations phenomenon

So far, we have implicitly assumed that for an available space window on the merge conveyor, we are aware of all parcels that request to be allocated to this space window. This assumption represents the *predictable case*. The validity of this assumption depends on the layout characteristics of the merge area. Basically, when the infeeds are of a sufficient length, then at the fly through point all parcels from all infeeds that request the tray at the fly through point as their delivery tray are known. Therefore, the allocation decision is made using the information about alternative parcels.

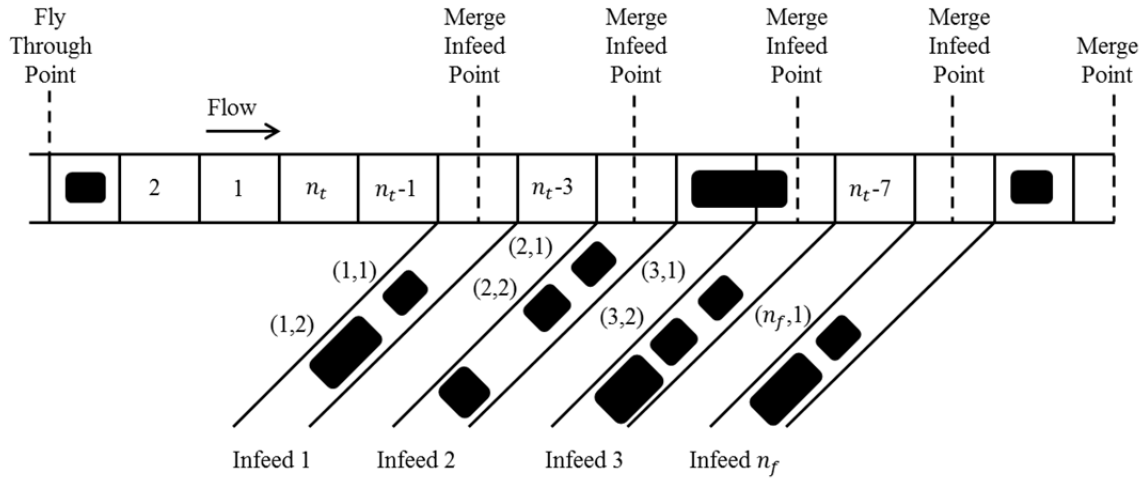


Figure 6.6. Merge area layout leading to the unpredictable case.

However, in most sorter systems the layout of the merge area does not lead to the predictable case. Figure 6.6 gives a layout of the merge area with short infeeds. In this case, we are not aware of all parcels that may arrive at a certain tray when the allocation decision for this tray is made. Therefore, if space windows on the merge conveyor are allocated by handling incoming requests from parcels on a FCFS basis, a deterioration in the workload balance among infeeds occurs due to the *early reservations phenomenon*. In order to describe this phenomenon, we sketch, in Figure 6.7, a merge configuration with two infeeds of limited length; each infeed has a parcel. As soon as a parcel is loaded on an infeed, it is announced in the system and requests a merge space. The parcel from infeed 1 can arrive at point A, and therefore can reserve it. The parcel from infeed 2 can arrive at point B. However, if all infeeds are busy with parcels, then already at an earlier decision moment, point B could have been allocated to some previous parcel from infeed 1. This phenomenon induces the dedication of most of the space as requested by parcels from infeed 1, while forcing parcels from infeed 2 to wait for a space at later points than requested. The main idea is that parcels from infeed 1 can reserve spaces on the merge conveyor earlier than parcels from infeed 2, due to the restricted look ahead horizon. Therefore, as the system operates for a long time, the total waiting time for parcels of infeed 2 accumulates. Moreover, in a larger system with more infeeds, this phenomenon propagates and may result in high imbalance measures.

This phenomenon mainly occurs when infeeds farther downstream are not long enough to see all incoming parcels that compete for the same merge space at the time

of making allocation decisions. Then, parcels from those infeeds are forced to wait before being merged, more than parcels from upstream infeeds. The PBA (Section 6.4.1) deals with the predictable case of the merge area, but it does not work for this *unpredictable case* because it depends on a set of announced parcels when making allocation decisions. As a result, the PBA needs to be adapted to cover this case.

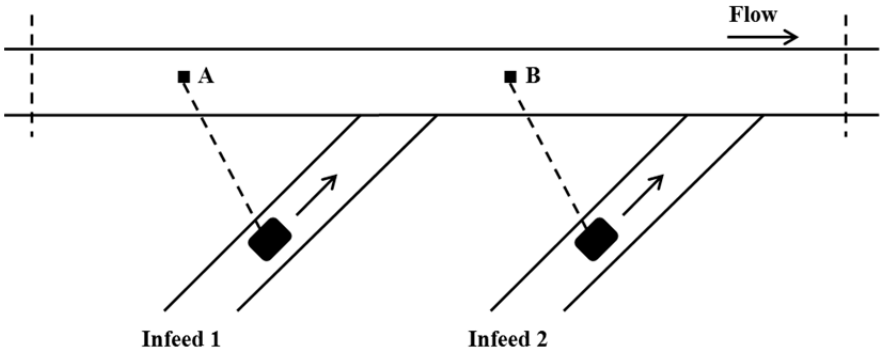


Figure 6.7. Layout leading to the early reservation phenomenon.

In this section, we discussed why a FCFS control principle does not work in practice for the unpredictable case of the merge area. Note that the FCFS is a good approach if we only strive for high throughput. However, it is not suitable in terms of workload balancing. Another basic control principle that is applied in practice is Round-Robin. Using the Round-Robin principle, we reserve space windows appearing at the fly through point for infeeds sequentially. However, the main problem is that a space window reserved for a certain infeed may not be used by this infeed when it arrives at the merge infeed point. This may occur because there may be no parcels to merge by the infeed or there may be a parcel with a size that does not fit in the reserved space window. In this case, space on the merge conveyor is lost (i.e., the space window cannot be used by upstream infeeds anymore because it has already passed them). This results in a deterioration in throughput, which is the main objective. Section 6.4.3 develops the PBA into a generic PBA that aims for high throughput and at the same time keeps the workload balanced.

6.4.3 A generic PBA

In order to extend the PBA to the unpredictable case, we propose to make allocation decisions based on available merge space requests. Thus, we do not reserve spaces on the merge conveyor in anticipation of parcels that are not yet announced. Thereafter, as new parcels are announced on infeeds, we improve the space allocation decisions by executing a *reallocation* procedure downstream the fly through point. Making space allocation decisions based on the available parcels tends to maximize throughput as the main objective. Thereafter, as new parcels appear, a reallocation procedure interferes to balance the workload by changing some allocation decisions. In this case, it may be possible to maintain a high throughput as the main objective and then balance the workload among infeeds by reallocation.

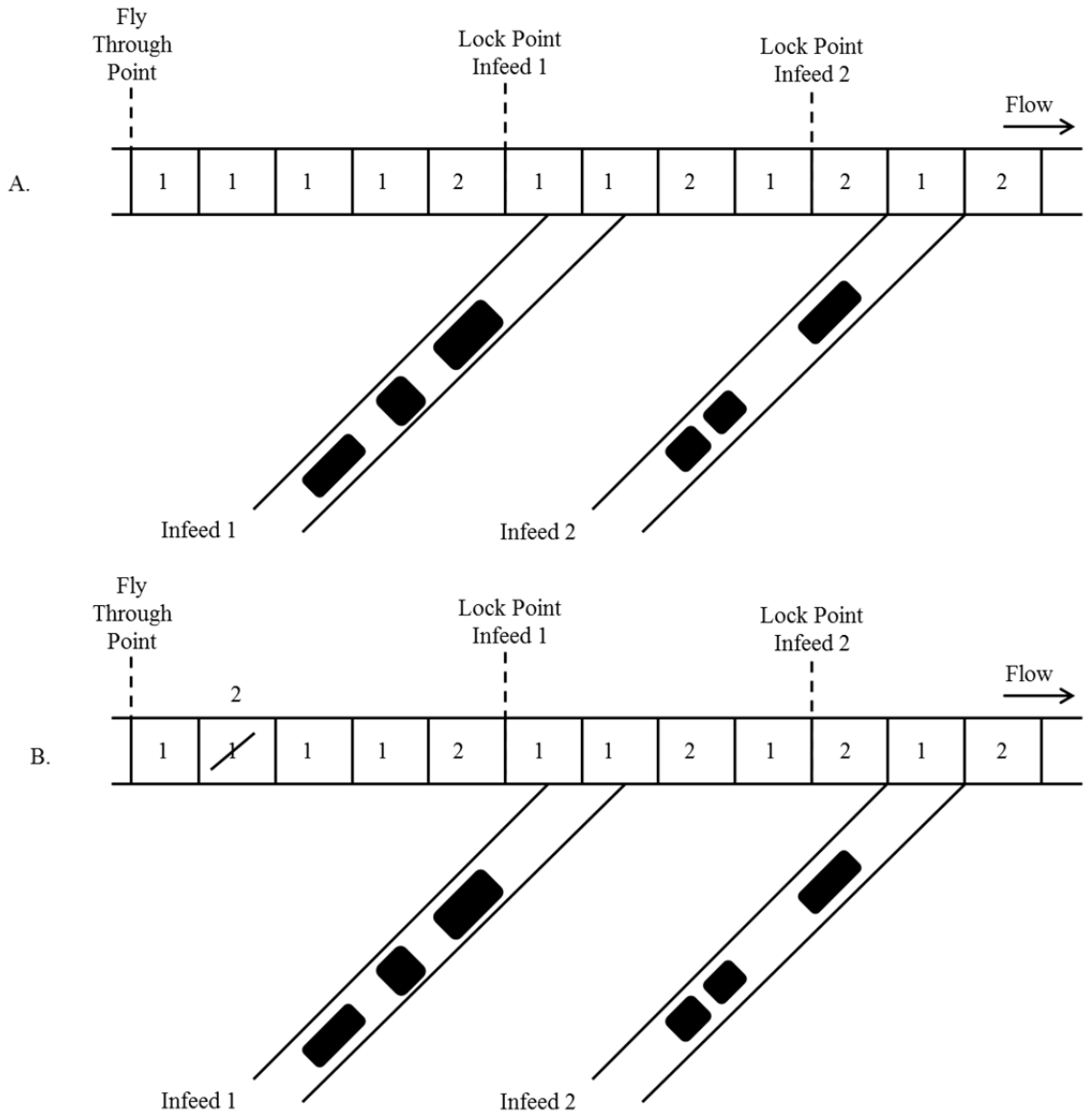


Figure 6.8. Reallocation of space windows in the merge area.

Figure 6.8 clarifies the basic concept of reallocation. At the fly through point, trays are allocated to parcels that have merge space requests. Figure 6.8A illustrates the first results of space allocation at the fly through point. The numbers on the trays refer to the infeed to which the tray is allocated. As the workload balance deteriorates due to the early reservations by infeed 1, reallocation is needed to retain the balance. Reallocation is achieved by canceling some of the trays allocated to parcels from infeed 1 and reallocating them to parcels from infeed 2 (see Figure 6.8B). An important point to mention is that reallocation is not possible for all trays. A tray is unavailable for reallocation if it is *locked*. For each infeed there is a point on the merge conveyor called the *lock point* (see Figure 6.8). When a tray that is allocated to infeed 1 passes the lock point of infeed 1, it becomes locked and cannot be reallocated to another infeed even though it is still physically empty. The reason is that after passing the lock point, the parcel assigned to this tray is already moved to the last segment of the infeed that is responsible for merging the parcel (see Section 6.1.1) and so

downstream the lock point the parcel assigned to this tray cannot be stopped from being merged. The reallocation procedure is based on searching possible spaces on the merge conveyor and uses priority calculations as described in the remainder of this section.

We extend the PBA (Section 6.4.1) to the *generic PBA* that is able to handle a merge area with infeeds of any length. To achieve this extension, we develop additional procedures to be added to the basic steps discussed in Section 6.4.1. Moreover, for the generic PBA, space windows appearing at the fly through point are no more the only trigger of allocation decisions. Announced parcels that request merge spaces downstream the fly through point also trigger allocation decisions. To clarify these concepts, we state the three main procedures that compose the generic PBA:

A. *The reallocation procedure:* this procedure aims at balancing the workload among infeeds by changing allocation decisions after more information becomes available in the system. This is an improvement procedure that is activated by parcels and not by available space windows. We detail this procedure later on.
B. *The queue procedure:* this procedure is the PBA described in Section 6.4.1, which allocates space windows, appearing at the fly through point, to parcels in the pending requests queue. Therefore, this procedure is activated by appearing space windows. The allocation of spaces is based on priorities.
C. *The search procedure:* an announced parcel may activate this procedure to search for an available merge space. The search area for a merge space starts from the first possible delivery tray of the parcel and ends at the fly through point upstream.

The algorithm only takes into account announced parcels whose downstream parcels are all allocated. If parcel (f, p) is announced but its downstream parcel $(f, p-1)$ is not allocated yet, then we cannot allocate parcel (f, p) as we do not know its first possible delivery tray on the merge conveyor. We only consider parcel (f, p) when parcel $(f, p-1)$ is allocated.

If parcel (f, p) is announced and all of its downstream parcels are allocated, it requires a merge space. Moreover, a specific delivery tray that is the first tray reachable by the parcel is determined. The search procedure starts at the specified delivery tray and searches upstream for a possible space window for the parcel. The parcel is assigned to the first possible space window. If no space window is found up to the fly through point, the parcel remains unassigned and is added to the pending requests queue.

The reallocation procedure is executed only for *priority parcels*. These are parcels that are announced on *priority infeeds*. An infeed has priority if the difference between its total waiting time and the minimum total waiting time among all infeeds exceeds a preset *threshold value*. Consequently, if a parcel is announced on an infeed with high priority, then the reallocation procedure is started and not the search procedure.

The reallocation procedure differs from the search procedure in one main point. This point is that the merge space search does not only consider empty trays, but it can also consider trays that are allocated to parcels with low priority. In this case, the priority

parcel can take the spaces assigned to parcels with low priority (as long as the trays assigned to low priority parcels are not locked), resulting in canceling the allocations for the parcels with low priority. Next, the canceled parcels are also processed by the reallocation procedure to find new merge spaces for them upstream the canceled merge spaces. The result of the reallocation procedure can be that a merge space downstream the fly through point is found to allocate the parcel, or that no merge space is found. In the latter case, the merge space request of the parcel ends up in the pending requests queue, as happens with a parcel that does not find a merge space by the search procedure.

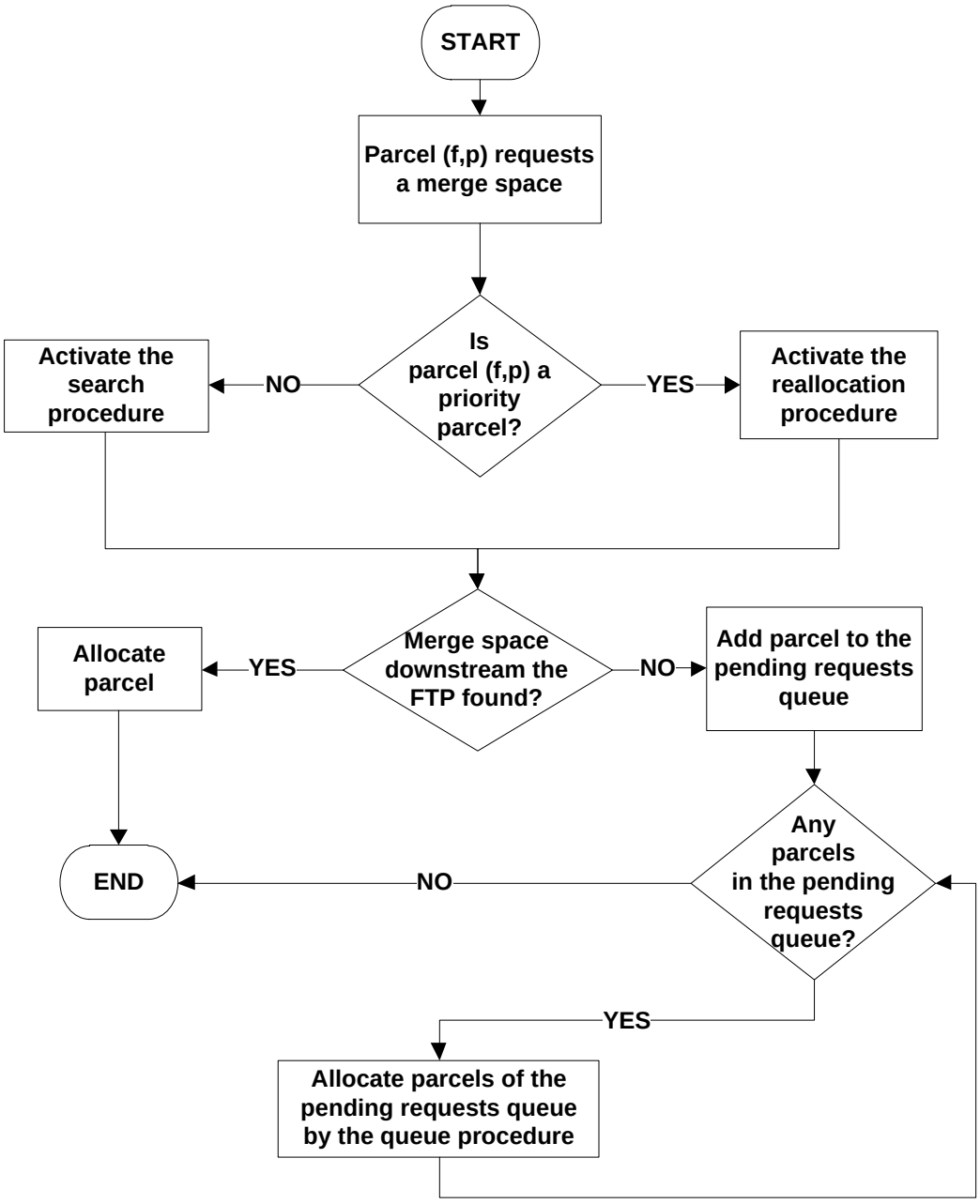


Figure 6.9. The basic scheme of the generic PBA.

The queue procedure is executed as long as there are parcels in the pending requests queue. It examines space windows appearing at the fly through point and allocates

them to parcels in the pending requests queue. Whenever a new space window appears, the queue procedure tries to allocate a parcel to it. This may not be possible if the space window is too small to fit any of the parcels of the queue. As the reallocation and the search procedures may be executed for incoming parcels that are not in the pending requests queue, the reallocation procedure can still cancel the allocations done by the queue procedure, while the search procedure can use only unallocated trays.

We note that the queue procedure deals with parcels that could not be allocated by the reallocation or the search procedures. The link between the reallocation and the search procedures on the one hand, and the queue procedure on the other hand, is the pending requests queue. The reallocation and the search procedures can add parcels to the pending requests queue; the queue procedure then deals with these parcels. We stress that the pending requests queue has at most one parcel from each infeed, since any parcel whose downstream parcel is unallocated is not considered for allocation. Therefore, when a merge space for parcel (f, p) is canceled, merge spaces of all parcels upstream parcel (f, p) are canceled. Moreover, if the merge space request of a parcel upstream parcel (f, p) is in the queue, then it is removed from the queue. Figure 6.9 presents the basic scheme of the generic PBA (where FTP stands for the fly through point).

6.5 Implementation

In order to implement the generic PBA and test its performance under varying operating conditions, we build a simulation model (in Delphi 2009) that represents a merge configuration with four infeeds.

6.5.1 Experimental setup

In this section, we present the setup to generate the input data and the method to measure the performance of the generic PBA. The relevant inputs are: parcels' lengths, parcels' inter-arrival times, and the *density* of flying through parcels. For these inputs we use data based on practice.

Parcels' lengths

Parcels' lengths are drawn from uniform distributions. Possible lengths are 1, 2, or 3 trays. In general, using a constant parcel length in an experiment is not realistic, and overlooks the challenges brought by the inconsistency in parcels' lengths.

Parcels' inter-arrival times

We represent time in terms of trays, and so the time measure results directly in a distance measure. We know that in the merge area, batches of parcels are successively placed next to the infeeds and operators load these parcels on the infeeds. Therefore, we model two possibilities to determine the inter-arrival time of a parcel.

1. The next parcel is a parcel from the same batch with 97% probability.
2. The next parcel is a parcel from a new batch with 3% probability.

If a parcel is from a new batch, then the inter-arrival time is taken as 30 trays, which is a constant value. If a parcel is from the same batch, then inter-arrival times are drawn from a uniform distribution according to one of the following three possible ranges:

- {1,2}
- {2,3,4}, which is the standard range unless mentioned otherwise,
- {4,5,6}

These ranges represent variable factors in the simulation experiments. Moreover, inter-arrival times directly result in follow-up distances on the merge conveyor.

Density of flying through parcels

For trays appearing at the fly through point, the density of flying through parcels refers to the fraction of trays occupied by flying through parcels from the total number of trays. We examine the following densities of flying through parcels:

- 0%
- 5%
- 15%, which is the standard value unless mentioned otherwise,
- 30%

For a certain layout of the merge area, the number of possible combinations of input parameters is 12; 3 (ranges of inter-arrival times) times 4 (densities of flying through parcels). We use 11 trays as it is the standard length of the infeeds in practice. Moreover, in each simulation experiment, we generate 2500 parcels on each infeed.

Performance measurement

The two objectives of the merge system are throughput maximization and workload balancing. We measure the performance of the algorithm in achieving these objectives as follows:

- Throughput maximization is measured by the *utilization* of space on the merge conveyor.
- Workload balancing is measured by the *imbalance in waiting times* between the infeed with maximum total waiting time and the infeed with minimum total waiting time, and expressed as a percentage as follows:

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

Hence, the algorithm aims at maximizing the utilization and minimizing the imbalance in waiting times.

6.5.2 Model parameterization

In this section, we tune the values of the main parameters in the generic PBA, which are the parameter α used in calculating the priority of parcels (see Section 6.4.1) and the threshold value for the reallocation procedure (see Section 6.4.3).

The threshold value for the reallocation procedure

This parameter is selected by the MHS supplier. It is relevant to how much difference in waiting times is acceptable and when is it necessary to execute a reallocation procedure. This also relates to the load on the control software, because calling the reallocation procedure more frequently results in more executions by the control software. In general, tuning the threshold value may depend on the specific system

modeled. For the merge configuration under study, we propose a standard threshold value of five.

The parameter α

We want to investigate how the weights of the objectives in priority calculations influence the results. This is technically done by varying the value of the parameter α, which plays a role in prioritizing the pending requests queue. To this end, we conduct 1320 experiments to tune the value of α. We use the average results of 10 test instances, for each of the 12 possible combinations of input data, and run the simulation for 11 different values of α ($\alpha = \{0, 0.1, 0.2, .., 1\}$). In these experiments, we use a threshold value of 5. For every combination of input parameters and every value of α, we calculate the resulting utilization and imbalance in waiting times (see Table 6.1).

α	Utilization (%)	ImbalanceWT (%)
0.0	86.38	17.01
0.1	86.84	4.20
0.2	86.84	4.20
0.3	86.84	4.22
0.4	86.82	4.22
0.5	86.83	4.21
0.6	86.81	4.25
0.7	86.84	4.28
0.8	86.88	4.23
0.9	86.85	4.27
1.0	86.03	4.27

Table 6.1. Average results for different values of α.

In order to determine a standard value for α, we first investigate the two extreme values (0 and 1). Table 6.1 shows that giving full weight to the throughput objective by setting α equal to 0 causes a dramatic deterioration in workload balancing, with no improvement in throughput. On the other hand, giving full weight to workload balancing by setting α equal to 1 causes a limited deterioration in throughput, but with no improvement in workload balancing as compared to values smaller than 1 (but larger than 0). Note that minimizing the imbalance tends to maximize throughput (see Section 6.3.3). We observe that the effect of changing the value of α between 0 and 1 is low. This is because α only plays a role in the queue procedure, while in the unpredictable case of the merge area, the reallocation and the search procedures play a vital role. The maximum throughput (which is the main objective) is achieved by setting α equal to 0.8, and the workload balancing at that value is also good. Therefore, we use this standard value of α.

6.5.3 Experimental results

This section reports on the performance of the generic PBA under different operating conditions.

Performance of the generic PBA under varying operating conditions

We examine the performance of the generic PBA under varying operating conditions, where an operating condition is specified by a combination of input data (see Section 6.5.1). Table 6.2 shows the average results for each combination.

Density of flying through parcels	Performance indicators	Range of inter-arrival times		
		$\{1, 2\}$	$\{2, 3, 4\}$	$\{4, 5, 6\}$
0 %	Utilization (%)	*90.9*	*86.2*	*79.5*
	ImbalanceWT (%)	*2.1*	*1.2*	*21.8*
	Number of reallocations	*3339.9*	*4461.7*	*2826*
5 %	Utilization (%)	*91.8*	*86.7*	*81.1*
	ImbalanceWT (%)	*2*	*1.3*	*11.1*
	Number of reallocations	*2766.7*	*4044.2*	*3293.3*
15 %	Utilization (%)	*92.1*	*87.7*	*83*
	ImbalanceWT (%)	*2.2*	*1.7*	*1.2*
	Number of reallocations	*1642.4*	*3050.6*	*3330.3*
30 %	Utilization (%)	*91.3*	*88*	*84.2*
	ImbalanceWT (%)	*2.6*	*2.1*	*1.4*
	Number of reallocations	*632.7*	*1520.2*	*2273.6*

Table 6.2. Average results under varying operating conditions.

Based on the results in Table 6.2, we make the following general remarks:

- As the range of inter-arrival times increases, the utilization of space on the merge conveyor drops. In this case, the system is not busy with parcels and so there is an unavoidable deterioration in utilization.
- Given a fixed range of inter-arrival times, increasing the density of flying through parcels increases the utilization of the merge conveyor in most of the cases. As the density of flying through parcels increases, the parcels on the infeed are more likely to wait for merge spaces. When parcels wait, it is likely that they are added to the pending requests queue. Therefore, allocation by the queue procedure may utilize the space on the merge conveyor more efficiently as prioritizing the pending requests queue considers space utilization, especially filling the gaps between flying through parcels. However, in some cases, when the density of flying through parcels is too high, the utilization may drop. This may happen because when the merge conveyor is highly loaded with flying through parcels, it becomes more difficult to merge parcels from the infeeds between these flying through parcels. For example, it is difficult to merge a

large parcel when there are many flying through parcels with small spaces between them on the merge conveyor.

- For low and medium ranges of inter-arrival times, {1,2} and {2,3,4}, varying the density of flying through parcels has a minor effect on the imbalance in waiting times.
- For a high range of inter-arrival times, {4,5,6}, varying the density of flying through parcels has a big impact on the performance of the algorithm with regard to the workload balancing objective. In this operating condition, the effect of the queue procedure is limited, because the possibility that parcels from different infeeds are present in the queue is small. Moreover, the effect of the reallocation procedure is limited because trays allocated to low priority parcels may become locked before high priority parcels appear, due to long inter-arrival times and short waiting times of low priority parcels. Short waiting times are caused by no (or low density of) flying through parcels.
- When long inter-arrival times are combined with no flying through parcels, or to a lesser extent, a very low density of flying through parcels, workload balancing is not as efficient as for other conditions. However, this condition may not create a problem for the system and better workload balancing may not be achievable with other algorithms as well. In this case, the merge conveyor is not busy and the inter-arrival times of the parcels are long. Therefore, most of the parcels may find merge spaces with minimal waiting times. Some parcels may wait for a short time when the first possible delivery trays of parcels from different infeeds are coincidentally the same. However, the overall performance is acceptable because this operating condition describes a system that is not overloaded, where all infeeds have low waiting times. In this case, differences in waiting times among infeeds may not represent a real problem.

The effect of the length of the infeeds

In order to test the performance of the generic PBA algorithm for different lengths of the infeeds, we perform experiments under two modes of the generic PBA: (i) with the reallocation procedure, and (ii) without the reallocation procedure. Figure 6.10 shows the results for different lengths of the infeeds under modes (i) and (ii), using the standard values of input data (where L(x) means infeeds of the length x trays).

Figure 6.10 shows that longer infeeds yield slightly better results with regard to utilization. This is mainly due to the effect of the queue procedure, as longer infeeds make it more likely that parcels are added to the pending requests queue. Moreover, Figure 6.10 verifies that the reallocation procedure has big impact on balancing the workload among infeeds while not deteriorating throughput. It also shows that as the infeeds get longer, the impact of the reallocation procedure decreases. This is an intuitive result because when infeeds are long, we are aware of incoming parcels early, and the chance that parcels are added to the pending requests queue is higher. Consequently, prioritizing the queue by the queue procedure balances the workload and minimizes the need for further balancing via the reallocation procedure.

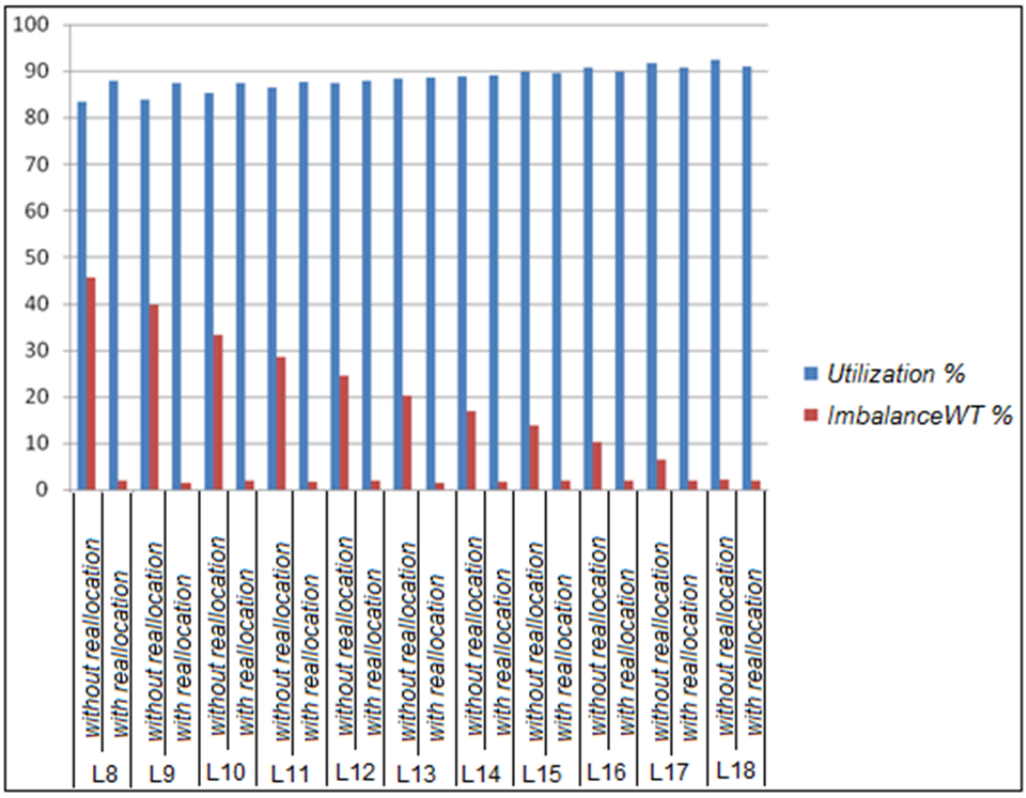


Figure 6.10. Effect of the length of the infeeds on the performance of the algorithm under standard operating conditions.

Distribution of waiting time among infeeds

Figure 6.11 displays the distribution of waiting times for each of the four modeled infeeds $\{Inf0,..,Inf3\}$ and for each input combination $\{C1,..,C12\}$. We observe that we are able to overcome the early reservations phenomenon (see Section 6.4.2). The first infeed has less waiting time than other infeeds only in combinations 9 and 10, which are the combinations that limit the effects of the reallocation and the queue procedures as described earlier.

Number of reallocations per infeed

Figure 6.12 shows the average number of reallocations per infeed for each input combination. We observe that parcels of the fourth infeed ($Inf3$ in Figure 6.12) are less frequently reallocated in all input combinations. We expect this result since the fourth infeed suffers the most from the early reservations phenomenon and so its parcels are normally high priority parcels. Similarly, parcels of the first and second infeeds ($Inf0$ and $Inf1$ in Figure 6.12) are expected to be low priority parcels, and so are frequently subject to reallocations.

6.6 Chapter conclusion

In this chapter, we have taken a closer look at the space allocation problem in merge configurations, which is a local traffic problem that occurs in all industrial sectors. The objective was to develop a generic space allocation algorithm that provides the necessary control for merge configurations.

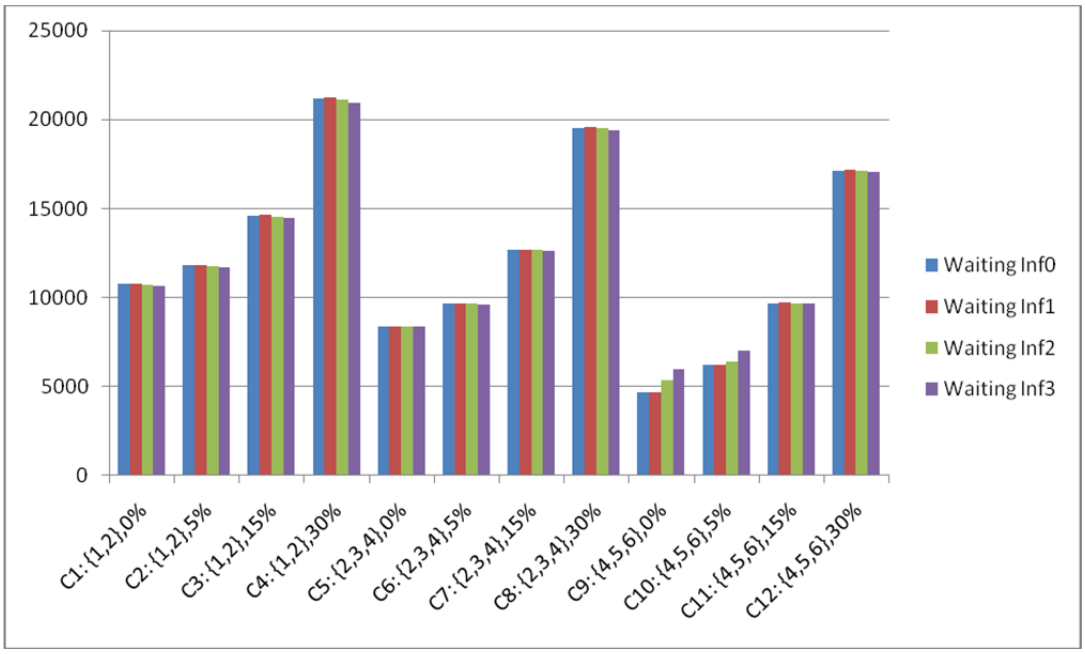


Figure 6.11. Distribution of waiting times among infeeds (measured in trays).

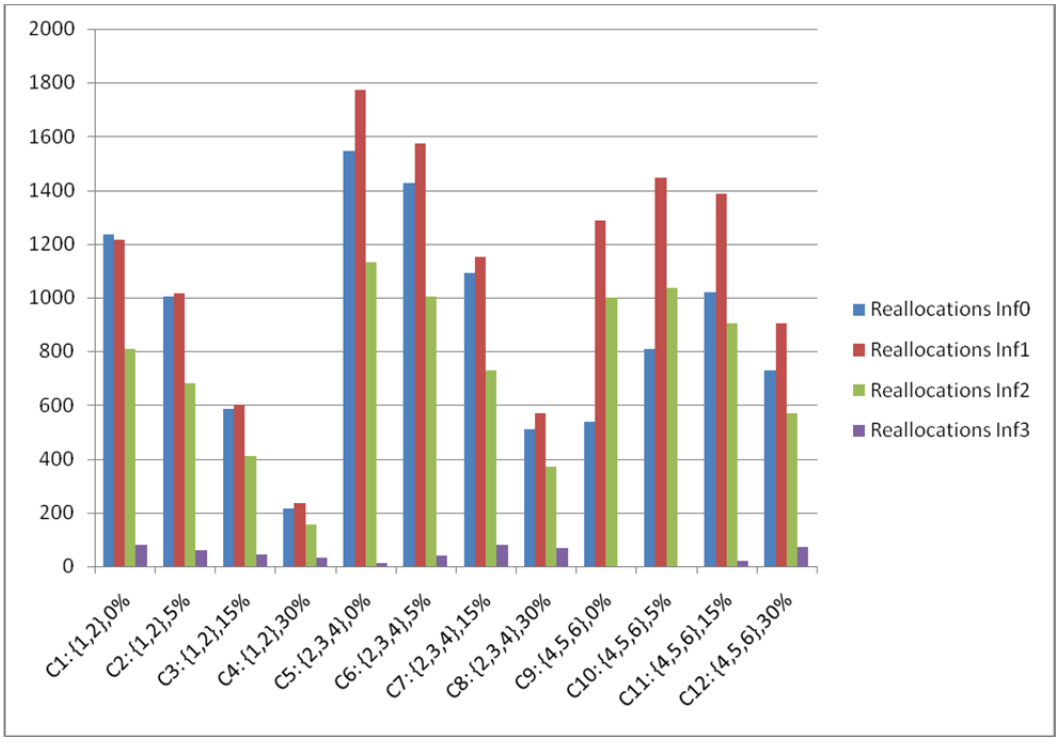


Figure 6.12. Number of reallocations executed per infeed.

Although we took a discrete modeling approach of the conveyors, the generic PBA can be easily implemented on a continuous belt conveyor (e.g., instead of counting empty trays for a space window, we can measure empty distances). Moreover, we have studied the space allocation problem in the context of parcel & postal sorter systems, but the results are generic to applications in the other industrial sectors. For example, in baggage handling or in distribution, the outfeeds of storage aisles in ASRSs connecting to a merge conveyor represent a merge configuration.

The generic PBA is flexible in terms of incorporating possibly different objectives and different prioritization rules, depending on the application. For example, in baggage handling, the priorities may be based on urgent bags.

As a final word, we comment on how the generic PBA fits in the generic control architecture. Space allocation is a local traffic control problem. Therefore, as mentioned in Chapter 2, decision-making algorithms for these problems can be incorporated in a scheduling controller, e.g., the loop controller (see Section 2.2.3.1), which can execute local traffic control procedures. Otherwise, a dedicated local traffic controller in merge areas can be part of the control architecture to accommodate this algorithm.

Chapter 7

Conclusions, Recommendations, And Future Research

In Chapter 1, we introduced the research problem which concerns the development of a generic planning and control structure for MHSs as they occur in different industrial sectors, i.e., parcel & postal sorting, baggage handling, and distribution. Moreover, we discussed both the scope and limitations of the research problem on generic control. In the subsequent chapters, we investigated the problem on various planning and control levels.

In this chapter, we conclude the thesis by first presenting a summary of our results (Section 7.1), tracing them back to the research agenda that we proposed in Section 1.5.2. Second, we point out the main contribution of this thesis (Section 7.2). Third, we present general conclusions (Section 7.3). Fourth, we provide recommendations and guidelines for future applications of the generic control architecture (Section 7.4). Finally, we highlight directions for future research (Section 7.5).

7.1 The research agenda revisited

In Section 1.5.1, we weighed the practical requirements for a generic control architecture against contributions from the literature. Consequently, in Section 1.5.2, we proposed a research agenda of five points.

In Chapter 2, we dealt with the first two points on the research agenda, i.e., *proposing a concept control architecture and detailing it in terms of control levels and control units*. The concept control architecture we proposed was motivated by a number of design choices. Subsequently, we defined a set of generic control units distributed among three levels of control, i.e., planning, scheduling, and local traffic control. In Chapter 2, we presented our proposal for a general structure that covers all decision-making processes encountered in the control architecture. In this sense, Chapter 2 defined the necessary building blocks of this generic control architecture. We devoted the subsequent chapters to the development of these building blocks.

In Chapter 3, we dealt with the third and fourth element on the agenda, i.e., *translating the concept into a concrete control architecture and validating it*. In Chapter 3, we mainly developed planning and scheduling control modules and included local traffic control modules only in an aggregate way. We also showed implementations of the generic control modules that we developed on generic system models, which we tuned to simulate MHSs in different industrial sectors. A key deliverable in Chapter 3 is the analysis on the generality of the control modules given the implementation in different industrial sectors.

In Chapter 4, we dealt with the fifth item on the research agenda, i.e., *proving the adequacy of the control architecture*. In other words, we provided a proof-of-concept for the applicability of generic control on a practical case. To this end, we presented a comprehensive application of the control architecture on a challenging business case of a major European airport. In this large implementation, we showed how to handle new system elements in a generic manner and how to standardize control approaches for these different elements.

In Chapter 5, we extended our scope of analysis to propose scheduling algorithms for system users. We tackled the problem of scheduling inbound containers when loading them to MHSs that consist primarily of sorting elements. This scheduling problem influences the operation of sorter systems and requires sound scheduling tools. To this end, we analyzed different scheduling algorithms for different industrial sectors, different operational scenarios, and different system models. Then, we investigated which scheduling approach works best in which setting.

In Chapter 6, we tackled a local traffic problem that occurs in various industrial sectors. This is the space allocation problem in conveyor merge configurations. Common practice control rules for this local traffic problem do not perform well in satisfying the objectives of merge configurations. Therefore, we presented a detailed model of merge configurations and proposed a generic space allocation algorithm that satisfies system objectives. The algorithm is generic and can be implemented in different merge configurations within different industrial sectors.

7.2 Main contribution

In practice, control methods are customized for each industrial sector, and even for different MHSs within the same industrial sector. In this thesis, we focused on the high potential of generic control. The value of the proposed control architecture is that it helps MHSs' suppliers to improve on their services. In particular, MHSs' suppliers may benefit from the standardization in control methods to develop the control architecture of an MHS in shorter time and with less costs and effort than before, and hence facilitating the bidding process as well. A standardized and modular control architecture is also useful for MHSs' users for a number of reasons. For example, the operational environment and requirements of systems may vary over time, which necessitates the adaptation of the planning and control procedures. Obviously, a generic control structure may significantly simplify the implementation of new control strategies or the adjustment of existing ones, as compared to adaptations of a highly customized system. Moreover, a generic control architecture makes it easier to install software updates, maintain the control software, solve implementation problems, or even extend the system base.

In this thesis, we provided methods and modeling techniques that allowed us to deal with different MHSs in a generic manner. Consequently, we showed that generic control of MHSs in different industrial sectors is possible. The use of generic control is two-fold: first, from a scientific perspective, this thesis provides a basis to build a general control theory for MHSs in different industrial sectors. Second, from a practical perspective, we offer an approach for MHSs' suppliers that improves on the

MHSs' design aspects in terms of system development time, maintainability, and upgradeability.

The generic control architecture identifies the main decision-making processes at the right level of control, while layout-specific details are handled by configurable parameters. As a result, the control architecture is scalable and tunable to different system layouts and designs.

In general, one may expect that implementing generic control methods instead of customized control methods may bring benefits from the aforementioned perspectives, but at the cost of a decline in performance. However, we showed that the proposed generic control methods perform as good as the customized control approaches and in a number of cases outperform them.

7.3 General conclusions

In this section, we formulate the overall conclusions of this thesis.

- Our main conclusion is that a generic control system for MHSs in different industrial sectors is possible. To this end, we have shown what control tools are needed to model decision-making processes in a generic manner. For example, we proposed control procedures that bring one industrial sector (distribution) to the same level of detail as another industrial sector (baggage handling). After bridging that gap, standardized control procedures could be implemented.
- The control modules of the generic control architecture, which we developed throughout this thesis, can function together in the control architecture of a specific MHS. For example, in a large airport, we may use the planning and scheduling control modules as developed in Chapter 3 in combination with the local traffic control module of Chapter 6. By putting these modules together, the control architecture is comprehensive in terms of covering all decision making processes at the different levels of control. Moreover, for inbound containers in such a large airport, the system user may incorporate a scheduling algorithm for inbound containers as discussed in Chapter 6.
- Sector-specific processes can be built as functional add-ons to the control architecture. To this end, we showed how these processes can be integrated in a plug-and-work like mechanism.
- With regard to the generality of control, we experience more similarity among the different industrial sectors at the lower levels of control. As we move to the higher levels of control, we experience more situations that require adaptations to model the specificities of the different industrial sectors.
- The extent to which we can develop generic decision-making algorithms is dependent on the level of control. For example, at the local traffic level, we have more freedom to model systems similarly in all sectors. Therefore, in Chapter 6, we were able to develop a space allocation algorithm that is generic and can be easily reused in the different industrial sectors. As we go one level higher to the scheduling level, we analyzed algorithms for inbound containers scheduling. We discussed these algorithms in a generic framework, but the effect of the different sectors was more tangible than in the local traffic case

(e.g., the EBS and plane schedules affect the algorithmic design). Another example is the scheduling process for cranes' retrievals (see Chapter 3). For this process, we had to adapt to sector-specific characteristics when scheduling retrievals, although within a generic process structure.

- The highest level of differentiation among sectors is at the planning level. At this level, the interfaces with the system-user's functions play an important role and the different operational environments are more prevailing. An algorithmic analysis at this level is less generic. However, we presented a generic structure for planning processes. This is embodied by the modular planning control units and the standardized forms of communication. The standardized forms of communication occur between the planning control units at the same level and also with the scheduling control units at a lower level of control. This generic structure can accommodate sector-specific planning algorithms. For example, in baggage handling, a customized algorithm can be implemented in the storage planner to select a set of bags to compose a ULD for robots.

7.4 Recommendations and guidelines for practice

In this thesis, we presented a number of applications of the generic control modules. For future applications, we list some of the lessons learned, which are points to commit to when implementing the generic control architecture on other MHSs:

- The use of generic control modules is essential, since the introduction of customized modules would hamper the generic structure. For example, we model any type of 'build' workstations by the generic workstation module.
- The standard interfaces between different controllers should be maintained.
- System size and layout characteristics should not affect the implementation of the generic scheduling processes. This is shown by our application of the generic dynamic routing both in the screening area of the baggage handling business case (see Chapter 4) and in the orders build area of the distribution center modeled in Section 3.2.
- As mentioned before, the planning level of control is generic, but may include system-specific business rules or algorithms, since it is the interface to the users' processes.
- A distributed decision-making structure is necessary, as it supports the modularity, robustness, and the generic nature of the control architecture. For example, if the routing decisions are executed by a central controller, this controller would not be easily applicable to different system layouts.
- The pull-concept in the control architecture is beneficial to keep the workload under control and reduce variability. Moreover, the pull-concept facilitates more generality among different sectors. For example, we proposed a pull-concept for the retrieval process of bags from the EBS rather than the current practice push-concept. This resulted in a process than is analogous to the retrieval process in the distribution sector.
- Local traffic controllers can accommodate algorithms that can be easily integrated as add-ons to the architecture and do not affect the communication at the higher levels of control.

- Going back to the modular structure of control, we believe that using all control modules for an MHS may improve the overall system performance and the performance of individual modules. For example, the ADLBA is an approach that uses information about internal travel times in MHSs. For such an approach, it is beneficial when the MHS's control includes the planning, scheduling, and local traffic control modules as developed in the other chapters. In this case, the variability resulting from current practice local traffic control rules and EBS control is reduced by implementing the generic control methods. This makes internal travel times on sorters more reliable, and thus the performance of the ADLBA may improve. In this case, the ADLBA may be preferable in more settings than those recommended in Chapter 5.

7.5 Future research

In this thesis, we developed a generic control architecture for MHSs in different industrial sectors. We based our analysis, system models, and implementations on three different industrial sectors, i.e., parcel & postal sorting, baggage handling, and distribution. In fact, the issues we dealt with in this thesis open directions for future research. In this section, we highlight three possible research directions.

Material handling equipment

Throughout this thesis, we argued that we can model different equipment in abstract terms depending on the role they play in the system, and therefore generic control modules could be applied. For example, a build workstation can refer to a robot, a manned-lateral, or a manned-pick station. However, these arguments were based on the range of material handling equipment that we studied. In fact, it would be interesting to analyze the applicability of the generic control architecture on material handling equipment that were not within our scope of analysis.

In this context, an important area to investigate is autonomous vehicles, e.g., AGVs (automated guided vehicles). These material handling elements may be employed to perform tasks that some of the equipment we modeled perform. However, we cannot assume that the control approaches we proposed apply easily to these new equipment types. A key difference is that autonomous vehicles have more flexible movement trajectories than the equipment we modeled. This may result also in an overlap between the areas where each of these autonomous vehicles is active. We did not face such overlaps in the systems we modeled, e.g., each crane was operational in a dedicated aisle and a build workstation is functional at a designated area in the system. However, autonomous vehicles may access the same areas of operation and in this case there are additional issues to deal with. For example, the assignment of tasks to these resources may be affected by their locations at certain points in time. At the local traffic level, collisions between different autonomous vehicles and vehicles blocking each other are main control issues for which generic local traffic rules or algorithms should be developed and incorporated in the control architecture.

Generic control methods in different settings

In this thesis, we studied generic control of MHSs within a certain facility (Scope 1; see Section 1.2.1). For future research, it may be interesting to study the applicability

of the control methods that we proposed within different scopes and in other settings. For example, a question for future research is how to implement the generic control architecture not only in a single facility, but in multiple facilities that are connected, e.g., several terminals of a large airport. In this case, merely installing the control software for each facility may not result in a satisfactory system performance. In particular, the planning processes among different facilities may need coordination. A possibility is to have multiple layers of the planning control units. For example, a primary storage planner that supervises several subordinate storage planners might be an option. In this case, the primary storage planner has a system view on all facilities while each subordinate planner is responsible for the storage area of a single facility.

Another area of interest for future research may be to control the flow of people (instead of TSUs) in networks during big events. For example, when a major event takes place in some city and large crowds of people are expected to arrive from different cities via, e.g., trains, then it would be interesting to implement control methods for balancing flows over different railways (instead of conveyors) and to give real-time estimations on expected travel times.

On the other hand, it is also interesting to study control approaches that are developed for other settings and study whether it is possible to adapt them and incorporate them in the generic control architecture for MHSs.

Design for controllability

The aim of this thesis was to study generic control methods without interfering with the layout and process designs of MHSs. We dealt with MHSs as given in terms of processes and layouts. Then, we developed and applied generic control modules and adapted them, when necessary, to deal with system-specific characteristics.

The generic control architecture provides a sound basis for analyzing what system designs fit more easily in a generic context, and thus result in easier applicability of the generic control methods. Likewise, we may define what designs deviate from the generic context and make it more difficult to apply generic control methods. In this context, an interesting area for future research is to start with the deliverables of this thesis, i.e., the generic control methods, and then study how to design MHSs in a way that serves the generic control objectives. This is especially relevant when alternative layout designs are available to achieve a certain functional purpose. At the local traffic level, we gave an example in Chapter 6, where we showed how longer infeeds result in better system performance when using the generic space allocation algorithm.

Foreseeing the consequences of design decisions is an important issue not only in the context of MHSs, but in a more general context for systems in different areas, e.g., manufacturing or construction. The main reason is that the flexibility in decision-making is higher in the early stages than in the later stages of systems development. Therefore, certain design choices that may cause difficulties in later stages of systems' development or systems' functionality should be avoided. In fact, the attention paid to foreseeing the future effects of early design choices is growing. For example, in the manufacturing world, we encounter concepts such as *design for manufacturability* and *design for assembly*. Similarly, it might be interesting to propose the concept of *design for controllability* for future research.

References

Abdelghany, A., Abdelghany, K., & Narasimhan, R. (2006). Scheduling baggage-handling facilities in congested airports. *Journal of Air Transport Management*, **12** (2), 76 - 80.

Alsafi, Y. & Vyatkin V. (2010). Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. *Robotics and Computer-Integrated Manufacturing*. **26**(4): p.381-391.

Audsley, N. & Burns, A. (1990). Real-time system scheduling. Technical Report YCS 134, Department of Computer Science, University of York

Amato, F., Basile, F., Carbone, C., & Chiacchio, P. (2005). An approach to control automated warehouse systems*. Control Engineering Practice*. **13**(10): p. 1223-1241.

Anthony, R.N. (1965). *Planning and control systems: a framework for analysis*. Harvard Business School Division of Research. Boston

Babiceanu, R.F. & Chen, F.F. (2005). *Performance evaluation of agent-based material handling systems using simulation techniques*. Proc. of the Winter Simulation Conference, Orlando, FL, December 2005.

Babiceanu, R.F., Chen, F.F., & Sturges, R.H. (2004). Framework for the control of automated material-handling systems using the holonic manufacturing approach. *International Journal of Production Research*. **42**(17): p. 3551-3564.

Boysen, N., & Fliedner, M. (2010). Cross dock scheduling: classification, literature review and research agenda. *Omega*. **38**: p. 413 - 422

Bozer, Y.A. & Hsieh, Y. (2005). Throughput performance analysis and machine layout for discrete-space closed –loop conveyors. *IIE Transactions*. **37**(1): p. 77-89.

Chen, X.W. & Nof, S.Y. (2007). *Prognostics and diagnostics of conflicts and errors over e-Work networks*, in Proceedings of the 19th international conference on production research (ICPR-19). Valparaiso, Chile, July 2007.

Claes, R., Holvoet, T., & Weyns, D. (2011). A decentralized approach for anticipatory vehicle routing using delegate multi- agent systems. *IEEE Transactions on Intelligent Transportation Systems*. **12**(2): p. 364-373.

Cohen, Y. & Keren, B. (2009). Trailer to door assignment in a synchronous cross-dock operation. *International Journal of Logistics Systems and Management*, **5**(5): p. 574 - 590.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik.* **1**(1): p. 269-271.

Dilts, D.M., Boyd, N.P. & Whorms, H.H. (1991). The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems*. **10**(1): p. 79-93.

Faber, N., Koster, M.B.M. de & Velde, S.L. van de (2002). Linking warehouse complexity to planning and control structure. *International Journal of Physical Distribution and Logistics Management*. **32**(5): p. 381-395.

Faber, N., Koster, M.B.M. de & Smidts, A. (2013). Organizing Warehouse Management, *International Journal of Operations & Production Management*, **33**(9): in press.

Frey, M., Artigues, C., Kolisch, R., & Lopez, P. (2010). Scheduling and planning the outbound baggage process at international airports. In *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*. Macao, China.

Furmans, K., Schönung, F., & Gue, K.R. (2010). Plug-and-Work material handling systems. In Progress in Material Handling Research: Proceedings of 2010 International Material Handling Research Colloquium, eds. K.P. Ellis, K. Gue, R. de Koster, R. Meller, B. Montreuil, & M. Ogle, p. 132-142.

Giret, A. & Botti V. (2004). Holons and agents. *Journal of Intelligent Manufacturing*. **15**(5): p. 645-659.

Gue, K.R. (1999). The effects of trailer scheduling on the layout of freight terminals. *Transportation Science*, **33**(4): p. 419–428.

Gue, K.R., Furmans, K., Seibold, Z., & Uludağ, O. (2013). GridStore: A Puzzle-based storage system with decentralized control, forthcoming in *IEEE Transactions on Automation Science and Engineering.*

Gu, J., Goetschalckx, M., & McGinnis, L.F. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*. **203**(3): p. 539-549.

Gue. K.R. & Kim, B.S. (2007). Puzzle-based storage systems, *Naval Research Logistics.* **54**(5): p. 556-567.

Hadeli, Valckenaers, P., Kollingbaum, M., & Van Brussel, H. (2004). Multi-agent coordination and control using stigmergy. *Computers in Industry*. **53**(1): p. 75-96.

Hallenborg, K. (2007a). Domain of impact for agents collaborating in a baggage handling system. *Artificial Intelligence and Innovations 2007: from Theory to Applications*. C. Boukis, A. Pnevmatikakis & L. Polymenakos, Springer US. 247: p. 243-250.

Hallenborg, K. (2007b). Decentralized scheduling of baggage handling using multi-agent technologies. In E. Levner (Ed.), *Multiprocessor scheduling*. Vienna, Austria: I-Tech Education and Publishing.

Hallenborg, K. & Demazeau, Y. (2004). *Dynamical Control in Large-Scale Material Handling Systems through Agent Technology*, IAT '06. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, p. 637-645.

Haneyah, S., Hurink, J., Schutten, M., Zijm, H., & Schuur, P. (2011). Planning and Control of Automated Material Handling Systems: The Merge Module. In: B. Hu, K. Morasch, S. Pickl, & M. Siegle, editors. *Operations Research Proceedings 2010*, Part 8, pages 281-286, Springer Heidelberg Dordrecht London New York, 2011. ISBN 978-3-642-20008-3.

Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., & Zijm, W.H.M. (2013a). Generic planning and control of automated material handling systems. *Computers in Industry*, **64**(3): p. 177-190.

Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., & Zijm, W.H.M. (2013b). A generic material flow control model applied in two industrial sectors. *Computers in Industry*, **64**(6): p. 663-677.

Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., & Zijm, W.H.M. (2013c). Application of a generic control architecture for automated material handling systems to a baggage handling system. In: J.-L.Ferrier, O.Y. Gusikhin, K. Madani, & J. Sasiadek, (Eds). *Proceedings of ICINCO 2013 - 10th International Conference on Informatics in Control, Automation and Robotics*, Reykjavik, Iceland, 29-31 July, 2013, Volume 1, pages 121-130, SciTePress 2013 ISBN: 978-989-8565-70-9.

Haneyah, S.W.A., Schutten, J.M.J., & Fikse, K. (2013d). Throughput Maximization of Parcel Sorter Systems by Scheduling Inbound Containers. To appear: in *International Logistics Science Conference 2013 proceedings*, part of the series Lecture Notes in Logistics (LNL); published by Springer

Haneyah, S.W.A., Schutten, J.M.J., & Fikse, K. (2013e). *Improving the Performance of Sorter Systems by Scheduling Inbound Containers.* submitted for publication (working paper: http://beta.ieis.tue.nl/node/2092)

Hax, A.C., & Meal, H.C. (1975). Hierarchical integration of production planning and scheduling. *Geisler M (ed) TIMS Studies in the management sciences: logistics.* North Holland-American Elsevier, Amsterdam: p. 53-69.

Hsieh, Y.J. & Bozer, Y. (2005). Analytical Modeling of Closed-Loop Conveyors with Load Recirculation. *Computational Science and Its Applications – ICCSA 2005*. O. Gervasi, M. Gavrilova, V. Kumaret al, Springer Berlin Heidelberg. **3483**: p. 437-447.

Hunter, T. (1994). *Simulation model evolution a strategic tool for model planning*. Proc. of the Winter Simulation Conference, Lake Buena Vista, FL, December 1994.

Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L. K., & Young, T. (2010). Simulation in manufacturing and business: A review. *European Journal of Operational Research.* **203**(1): p. 1-13.

Jing, G.G., Kelton, W.D., Arantes, J.C., & Houshmand, A.A. (1998). Modeling a controlled conveyor network with merging configuration. *Proceedings of the 30th conference on Winter simulation*. Washington, D.C., USA, IEEE Computer Society Press: p. 1041-1048.

Jodin, D. & Wolfschluckner, A. (2010). Merge problems with high speed sorters, *Progress in Material Handling Research: 2010*, Charlotte, NC, USA, p. 186−196.

Johnstone, M., Creighton, D., & Nahavandi, S. (2010). Status-based routing in baggage handling systems: Searching verses learning. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 2010. **40**(2): p. 189-200.

Kamagaew, A., Stenzel, J., Nettsträter, A., & ten Hompel, M. (2011). Concept of Cellular Transport Systems in Facility Logistics. *Proceedings 5th International Conference on Automation, Robotics and Applications (ICARA),* Wellington, New Zealand, p. 40-45

Kim, B.I., Heragu, S., Graves, R.J., & Onge, A.St. (2003). A hybrid scheduling and control system architecture for warehouse management. *IEEE Transactions on Robotics and Automation*. **19**(6): p. 991-1001.

Koster, R. de & Wijnen, R. (1998). How to obtain maximum capacity on high-capacity sorters (Published on CD-ROM). In R.J. Graves et al. (Ed.), *Progress in Material Handling Research: 1998* (pp. 143-158). Charlotte: Material Handling Institute.

Landschützer, C., Wolfschluckner, A., & Jodin, D. (2013). CAE for a high performance in-feed processes at sorting systems. *Proceedings in Manufacturing Systems*. **8**(2): p. 79-86

Lau, H.Y.K. & Woo, S.O. (2008). An agent-based dynamic routing strategy for automated material handling systems. *International Journal of Computer Integrated Manufacturing*. **21**(3): p. 269-288.

Li, Z.P., Low, Y.H., Shakeri, M., & Lim, Y.G. (2009). Cross docking planning and scheduling: Problems and algorithms. *SIMTech technical reports*, **10**(3): p. 159-167.

Lu, C., Stankovic, J.A., Tao, G., & Son, S.H. (1999). Design and Evaluation of a Feedback Control EDF Scheduling Algorithm. *Proceedings of the 20th IEEE Real-Time Systems Symposium*, IEEE Computer Society: **20**: p. 56-67.

Lu, C., Stankovic, J.A., Tao, G., & Son, S.H. (2002). Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms. *Real-Time Systems* **23**(1/2): p. 85-126.

Mayer, H.S. (2009). *Development of a completely decentralized control system for modular continuous conveyors*, Ph.D. dissertation, Universiẗat Karlsruhe (TH)

McAree, P., Bodin, L., & Ball, M. (2002). Models for the design and analysis of a large package sort facility. *Networks*, **39** (2): p. 107-120.

McAree, P., Bodin, L., Ball, M., & Segars, J. (2006). Design of the federal express large package sort facility. *Annals of Operations Research*. **144**(1): p. 133-152.

McWilliams, D. L., Stanfield, P.M., & Geiger, C.D. (2005). The parcel hub scheduling problem: A simulation-based solution approach. *Computers & Industrial Engineering*, **49** (3): p. 393-412.

McWilliams, D.L. (2005). Simulation-based scheduling for parcel consolidation terminals: a comparison of iterative improvement and simulated annealing. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, & J. A. Joines (Eds.), *Proceedings of the 2005 Winter Simulation Conference*. Orlando, FL, USA.

McWilliams, D.L. (2009a). Genetic-based scheduling to solve the parcel hub scheduling problem. *Computers & Industrial Engineering*, **56**(4): p. 1607-1616.

McWilliams, D.L. (2009b). A dynamic load-balancing scheme for the parcel hub scheduling problem. *Computers & Industrial Engineering*, **57**(3): p. 958-962.

McWilliams, D.L. (2010). Iterative improvement to solve the parcel hub scheduling problem. *Computers & Industrial Engineering*, **59**(1): p. 136-144.

Meinert, T.S., Don Taylor, G., & English, J.R. (1999). A modular simulation approach for automated material handling systems. *Simulation Practice and Theory*. **7**(1): p.15-30.

Mo, D.Y., Cheung, R.K., Lee, A.W., & Law, G.K. (2009). Flow diversion strategies for routing in integrated automatic shipment handling systems. *IEEE Transactions on Automation Science and Engineering*. **6**(2): p. 377-384.

Nazzal, D. & El-Nasher, A. (2007). Survey of research in modeling conveyor-based automated material handling systems in wafer fabs. *Proceedings of the 2007 Winter Simulation conference*, p. 1781-1788.

Parunak, H.V.D., *Generation and analysis of multiple futures with swarming agents* (2010). Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10), Toronto, Canada, May 10-14, 2010. **1**: p. 1549-1550

Rajan, V.N. & Nof, S.Y. (1996). Cooperation Requirements Planning (CRP) for Multiprocessors: Optimal Assignment and Execution Planning. *Journal of Intelligent and Robotic Systems*. **15**: p. 419-435.

Ramamritham, K. & Stankovic, J.A. (1994). *Scheduling algorithms and operating systems support for real-time systems*, Proceedings of the IEEE , **82**(1): p.55-67

Robusté, F. & Daganzo, C.F. (1992). Analysis of baggage sorting schemes for containerized aircraft. *Transportation Research part A: Policy and Practice*, **26** (1): p. 75-92.

Rouwenhorst, B., Reuter, B., Stockrahm, V., van Houtum, G. J., Mantel, R. J., & Zijm, W.H.M. (2000). Warehouse design and control: Framework and literature review. *European Journal of Operational Research*. **122**(3): p. 515-533.

Schmidt, L.C. & Jackman, J. (2000). Modeling recirculating conveyors with blocking. *European Journal of Operational Research*, **124**(2): p. 422-436

Shladover, S.E. (1980). Operation of merge junctions in a dynamically entrained automated guideway transit system. *Transportation Research Part A: General*, **14**(2): p. 85-112.

Stankovic, J.A., Tian, H., Abdelzaher, T., Marley, M., Tao, G., Son, S., & Chenyang, L. (2001). *Feedback control scheduling in distributed real-time systems*. Proceedings of the 22nd IEEE Real-Time Systems Symposium, 2001. (RTSS 2001): p. 59-70.

Stankovic, J.A., Chenyang, L., Son, S., & Tao, G. (1999). *The case for feedback control real-time scheduling*. Proceedings of the 11th Euromicro Conference on Real-Time Systems, 1999: p.11-20.

Tařau, A., De Schutter, B., & Hellendoorn, H. (2009a). *Centralized versus decentralized route choice control in DCV-based baggage handling systems*. Proceedings of the IEEE International Conference on Networking, Sensing and Control, Okayama, Japan, March 26-29, 2009

Tařau, A. N., De Schutter, B., & Hellendoorn, H. (2009b). *Hierarchical route choice control for baggage handling systems*. Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis, MO,USA, October 3-7, 2009

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., & Peeters, P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry*. **37**(3): p. 255-274.

Van den Berg, J.P. (1999). A literature survey on planning and control of warehousing systems. *IIE Transactions*. **31**(8): p. 751-762.

Vrba, P. & Mařík, V. (2006). Simulation in agent-based control systems: MAST case study. *International Journal of Manufacturing Technology and Management*. **8**(1): p.175-187.

Werners, B. & Wülfing, T. (2010). Robust optimization of internal transports at a parcel sorting center operated by Deutsche Post World Net. *European Journal of Operational Research*, **201**(2): p. 419-426.

Weyns, D., Holvoet, T., & Helleboogh, A. (2007). *Anticipatory Vehicle Routing using Delegate Multi-Agent Systems*, Intelligent Transportation Systems Conference ITSC 2007. IEEE , vol., no., p.87-93, Sept. 30-Oct. 3, 2007

Wurman, P.R., D'Andrea, R. & Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, **29**(1): p. 9-19.

Yu, W. & Egbelu, P.J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, **184** (1): p. 377-396.

Zijm, W.H.M. (2000). Towards intelligent manufacturing planning and control systems. *OR Spectrum*, 2000. **22**: p. 313-345.

Zimran, E. (1990). *Generic material handling system.* Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing

List of abbreviations

ADLBA	Adapted Dynamic Load Balancing Algorithm.
AGV	Automated Guided Vehicle.
ARB	Arbitrary scheduling.
ASRS	Automated Storage and Retrieval System.
BHS	Baggage Handling System.
DC	Distribution Center.
DCV	Destination Coded Vehicle.
DHL	Dalsey, A., Hillblom, L., and Lynn, R.
DLBA	Dynamic Load Balancing Algorithm.
EBS	Early Bags Storage.
FCFS	First-Come-First-Served.
ILP	Integer Linear Program.
ipm	items per minute.
IR	Irregularity Rate.
KPI	Key Performance Indicator.
MAS	Multi-Agent System.
MC	Machine Cluster.
MHS	Material Handling System.
PBA	Priority-Based Algorithm.
PHSP	Parcel Hub Scheduling Problem.
SKU	Stock Keeping Unit.
TNT	Thomas Nationwide Transport.
TSU	Transport Stock Unit.
ULD	Unit Load Device.
UPS	United Parcel Service.
WMS	Warehouse Management System.

Terminology

O

P

R

S

T

U

W

Acknowledgements

Accomplishing this PhD project would not have been possible without the help and support of a number of people. First of all, I want to thank my teachers and supervisors during my master's studies, who believed in me and encouraged me enthusiastically to pursue my PhD studies, in particular I thank Erwin Hans, Marco Schutten, Matthieu van der Heijden, Johann Hurink, and Leo van der Wegen. I am very grateful for your encouragement and very pleased with my decision to pursue my PhD studies, without which I would have often felt that something is incomplete. In this context, I express my thanks and appreciation to Henk Zijm, who has put a lot of effort with the university and with the industrial partner to set up this PhD project. This also leads me to thank our industrial partner and all of the mentors and team members I had there. You have put a lot of effort and provided all means to support the project from the beginning to the end. Concealing the names and details (in an attempt to protect confidentiality) does not underestimate your role.

Writing this thesis is a challenging process whose completion is not attributed only to myself. I cannot imagine that I could have better supervisors than Henk Zijm, Marco Schutten, and Peter Schuur. Henk, thanks for helping me formulate complex concepts in few accurate sentences that can be understood. Marco, thanks for keeping an eye on the discipline and accuracy of the text, and for highlighting those statements that could have a double meaning. Peter, thanks for always digging up those subtle textual details that could confuse the reader. Moreover, I want to thank Niels Fikse, whose master's thesis was part of my PhD project, without your work, Chapter 5 might not have been there.

If life has been only work, then I would not have been able to complete this project. I have been very fortunate to work in a pleasant environment with great colleagues. I will certainly miss your company and the nice discussions during our coffee breaks. Moreover, I thank my friends, for making my life very enjoyable and for the support you provided in all life matters. Most of you have already left Enschede, some of you are still here, but I prefer not to mention names in order not to miss any one. Finally, I would like to thank my precious family, thank you for your unlimited support, for always putting my interests and development as your top priority, and for tolerating my distance. In particular, I thank my parents, Waleed and Rasha, and my siblings, Deema, Ola, and Ehab. Last but not least, I thank my adorable wife Nadine, you have joined me in this journey and have always been there to support me and to make my life beautiful.

Sameh

Enschede, the 12th of August, 2013

About the author

Sameh Haneyah was born in Ramallah, Palestine, on the 1st of November, 1983. In 2001, he completed his pre-university education at the Evangelical Lutheran School of Hope in Ramallah. Afterwards, he studied Industrial Engineering at the Eastern Mediterranean University in North Cyprus, where he earned his bachelor of science degree in 2006. After a short working period in industry, Sameh moved, in 2007, to the Netherlands to pursue his master's studies in Industrial Engineering and Management at the University of Twente, with Logistics and Production Management as a specialization. For his master's thesis, Sameh collaborated with a major industrial partner to develop a real-time scheduling tool for parcel sorting systems. The results of his master's thesis led to the initiation of a PhD project on generic planning and control of material handling systems. Sameh worked on his PhD project at the University of Twente in collaboration with the same industrial partner. This thesis presents the results of the project.

List of publications

Haneyah, S., Hurink, J., Schutten, M., Zijm, H., & Schuur, P. (2011). Planning and Control of Automated Material Handling Systems: The Merge Module. In: B. Hu, K. Morasch, S. Pickl, & M. Siegle, editors. *Operations Research Proceedings 2010*, Part 8, pages 281-286, Springer Heidelberg Dordrecht London New York, 2011. ISBN 978-3-642-20008-3.

Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., & Zijm, W.H.M. (2013a). Generic planning and control of automated material handling systems. *Computers in Industry*, **64**(3): p. 177-190.

Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., & Zijm, W.H.M. (2013b). A generic material flow control model applied in two industrial sectors. *Computers in Industry*, **64**(6): p. 663-677.

Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., & Zijm, W.H.M. (2013c). Application of a generic control architecture for automated material handling systems to a baggage handling system. In: J.-L.Ferrier, O.Y. Gusikhin, K. Madani, & J. Sasiadek, (Eds). *Proceedings of ICINCO 2013 - 10th International Conference on Informatics in Control, Automation and Robotics*, Reykjavik, Iceland, 29-31 July, 2013, Volume 1, pages 121-130, SciTePress 2013 ISBN: 978-989-8565-70-9.

Haneyah, S.W.A., Schutten, J.M.J., & Fikse, K. (2013d). Throughput Maximization of Parcel Sorter Systems by Scheduling Inbound Containers. To appear: in *International Logistics Science Conference 2013 proceedings*, part of the series Lecture Notes in Logistics (LNL); published by Springer

Haneyah, S.W.A., Schutten, J.M.J., & Fikse, K. (2013e). *Improving the Performance of Sorter Systems by Scheduling Inbound Containers.* submitted for publication (working paper: http://beta.ieis.tue.nl/node/2092)