KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

A Holonic Logistics Execution System for Cross-docking

Jan Van Belle

Dissertation presented in partial fulfillment
of the requirements for the degree of
Doctor in Mechanical Engineering

October 2013

A Holonic Logistics Execution System for Cross-docking

Jan VAN BELLE

Supervisory Committee:
Prof. dr. ir. W. Sansen, chair
Prof. dr. ir. D. Cattrysse, supervisor
Dr. ir. P. Valckenaers, co-supervisor
Prof. dr. ir. G. Vanden Berghe
Prof. dr. T. Holvoet
Prof. dr. K. Sörensen
(Universiteit Antwerpen)
Prof. dr. ir. J.C. Wortmann
(Rijksuniversiteit Groningen)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Mechanical Engineering

October 2013

D/2013/7515/134
ISBN 978-94-6018-749-0

Voorwoord

De start van mijn doctoraat - waarvan u nu het resultaat in handen hebt - ligt in 2008. Ik wil echter eerst nog een aantal jaar verder terug in de tijd gaan. Tijdens mijn ingenieursstudies werden er bedrijfsbezoeken georganiseerd om kennis te maken met de rol van de ingenieur in het bedrijfsleven. In het kader van zo'n bezoek konden we spreken met een ingenieur die zich helemaal niet bezig hield met technische aangelegenheden, maar vooral met organisatorische zaken en planningsaspecten van productie. Het leek mij een vreemde jobinhoud voor een ingenieur. Maar jawel, in 2008 startte ik dus met een doctoraat dat focuste op organisatorische aspecten in productie en logistiek.

Dit doctoraat is het resultaat van vijf jaar onderzoekswerk binnen de MACC-onderzoeksgroep van het Departement Werktuigkunde. Het was een leerrijke en verrijkende ervaring waarbij ik verschillende onderzoeksdomeinen leerde kennen. Dit werk was echter alleen mogelijk dankzij de hulp en steun van mijn collega's en omgeving. Graag wil ik dan ook iedereen bedanken die, zowel op professioneel als op persoonlijk vlak, heeft bijgedragen tot het realiseren van dit doctoraat.

Eerst en vooral wil ik graag mijn promotoren, prof. Dirk Cattrysse en Paul Valckenaers, hartelijk danken. Dirk en Paul, bedankt voor het vertrouwen dat jullie in mij gesteld hebben, voor de steun en alle advies en feedback op mijn onderzoek. Mijn dank gaat ook uit naar prof. Hendrik Van Brussel voor zijn onmisbare steun aan de MACC-onderzoeksgroep.

Verder wil ik ook de voorzitter en de andere leden van mijn jury bedanken. Prof. Willy Sansen, prof. Greet Vanden Berghe, prof. Kenneth Sörensen, prof. Tom Holvoet en prof. Hans Wortmann, dank u wel voor het grondig nalezen van de tekst, de discussie op de preliminaire verdediging en de interessante feedback op de tekst.

Bedankt ook aan alle medewerkers van het departement, PMA en CIB voor de noodzakelijke administratieve en logistieke steun. Evy Neyens, Eva Vaes, Anja Vansteenwegen, Karin Dewit, Lieve Notré, Carine Coosemans, Regine

Vanswijgenhoven, Ronny Moreas en Jan Thielemans, dank u wel voor alle hulp.

Zonder mijn (oud-)collega's van zowel PMA als CIB zou het vervolledigen van dit doctoraat mij een stuk zwaarder gevallen zijn. Hiervoor wil ik hen hartelijk danken. A special word of thanks to Kris De Moerlooze, Dimitri Coemelck, Tegoeh Tjahjowidodo, Osman Ali, Rudi Bahtiar, Hadeli, Juan Matias Novas, Johan Philips, Paul Verstraete and Bart Saint Germain, among other things for the many coffee breaks, alma lunches and animated discussions. Also many thanks to the (former) colleagues from the MACC group. Unfortunately, I am the last of the Mohicans to graduate, but Paul, Hadeli, Osman, Rudi, Johan, Paul and Bart, thank you for the many fruitful discussions, the joint work, the shared conference experiences and the very nice work atmosphere and friendship.

In addition, a special thank you goes to the people I shared the office with during my years in the MACC group. Hadeli, in fact I hijacked your desk when I started working at the group. Thank you for all your help and your always cheerful character. Matias, you joined the MACC group for several months. Many thanks for the nice and interesting cooperation. Paul, je hebt de groep al een tijdje verlaten, maar ik herinner me nog levendig de vele interessante discussies en inspirerende brainstorms. Heel erg bedankt voor alle hulp, steun en vriendschap. Bart, gedurende al die jaren hebben we samen een bureau gedeeld. Je was naast een goeie collega een goeie vriend die ik enorm waardeer. Bedankt voor alle hulp en feedback op mijn werk gedurende al die jaren, evenals alle steun bij de lastige laatste loodjes van mijn doctoraat.

Naast collega's wil ik ook mijn vrienden danken voor de nodige ontspanning en afleiding tussen het doctoraatswerk door. Eerst en vooral de oud-studiegenoten vanop ESAT. Bedankt voor de vele activiteiten, feestjes en weekendjes. Het was steeds een plezier bij jullie te zijn. Tevens wil ik mijn oud-straatgenoten Michiel en Dieter bedanken voor de vriendschap en gezellige etentjes. Ook bedankt aan Matthias, Maarten en Bert voor de ontspannende momenten samen. Daarnaast wil ik ook graag alle anderen danken: de oud-leiding van KSA, de HRC 'bosklappers', de volleyballers van KU Leuven en al diegenen die ik hier nu vergeet. Tot slot nog een speciaal woordje van dank voor Katrien, die mij, ondanks alles, al die jaren van mijn doctoraat is blijven steunen en motiveren.

Ten slotte wil ik ook mijn ouders samen met Anne en Franky, Mieke en Michael danken. Bedankt voor alle steun, aanmoedigingen en alle kansen die jullie mij gegeven hebben. Ook nog een speciaal woordje voor mijn petekind Remi. Nog zo klein (maar wel al 1 jaar!), en toch slaagde je erin om altijd een glimlach op mijn gezicht te toveren, zelfs tijdens de moeilijke (schrijf)momenten.

Jan Van Belle
Leuven, oktober 2013

Abstract

The coordination and control of cross-docking operations is a complex task and a challenging problem. A severe competition in the logistics sector and ever-increasing traffic (congestion) make that cross-docks should be able to operate efficiently in uncertain and dynamic environments. Moreover, the operational coordination and control is a going concern, so 'one-shot optimization' is not sufficient.

As the logistics and manufacturing domain are characterized by similar properties (e.g. large decision space, nonlinearity, uncertainty, etc.), this thesis applies the concepts and principles of the Holonic Manufacturing Execution System (HMES) in order to develop a Holonic Logistics Execution System (HLES). The HLES is in charge of the organization of the logistic operations (e.g. resource allocation, tracking and tracing, etc.). By combining the PROSA reference architecture and the delegate MAS pattern in its software architecture, the HLES is able to provide a view on the expected short-term future of the system. The provided visibility is a valuable property for the staff, and also allows the control system to take better (informed) decisions.

The objective of this thesis is to show that the HLES concept is a valuable option for a cross-docking logistics execution system. To this end, the thesis explains how such an HLES implementation can be developed. First, several adaptations and extensions to the latest HMES implementation are proposed. The HLES consists of a reusable core, domain-specific models and application-specific decision mechanisms. The introduced adaptations to the core allow the HLES to offer support for mobile resources (e.g. trucks), batching and multi-resource allocation. The thesis also presents the necessary models and decision mechanisms for the entities relevant in the context of cross-docking. It has been shown by simulation experiments that the holonic technology is applicable and that the developed system works as intended.

Second, to support the cross-docking HLES in finding good global solutions, a

vehicle routing scheduling system and truck scheduling system are developed. A cooperation mechanism between the holonic on-line control system and these scheduling systems is also presented. As the developed scheduling systems make use of a simplified and approximated model of reality, the schedules provided by these systems can be (partially) infeasible. The HLES is however able to compensate for several simplifications. Moreover, the HLES also deals with deviations from the schedule, making the scheduling approach more robust against disturbances and uncertainty. Simulation experiments confirm that the cooperation between the HLES and external scheduling systems improves the performance.

Beknopte samenvatting

De coördinatie en controle van cross-docking activiteiten is een complexe taak en een uitdagend probleem. Een scherpe concurrentie in de logistieke sector en het steeds toenemende verkeer (files) maken dat cross-docks in staat moeten zijn om efficiënt te functioneren in onzekere en dynamische omgevingen. Bovendien vereist de operationele coördinatie en controle van deze activiteiten een ononderbroken zorg en is eenmalige optimalisatie niet voldoende.

Aangezien logistiek en productie gekenmerkt worden door vergelijkbare eigenschappen (bv. grote beslissingsruimte, niet-lineariteit, onzekerheid, enz.), past deze thesis de concepten en principes van het holonisch productie-uitvoeringssysteem (Holonic Manufacturing Execution System of HMES) toe om een holonisch logistiek uitvoeringssysteem te ontwikkelen (Holonic Logistics Execution System of HLES). De HLES staat in voor de organisatie van de logistieke operaties (bv. het toewijzen van hulpmiddelen, 'tracking' en 'tracing', enz.). Door het combineren van de PROSA referentie-architectuur en het 'delegate MAS' patroon in haar software architectuur, is de HLES in staat om een beeld te geven van de verwachte kortetermijntoekomst van het systeem. De verschafte zichtbaarheid is een waardevolle eigenschap voor het personeel en maakt het ook mogelijk voor het uitvoeringssysteem om beter (geïnformeerd) beslissingen te nemen.

Het doel van deze thesis is om aan te tonen dat het HLES-concept een waardevolle optie is voor een cross-docking logistiek uitvoeringssysteem. De thesis legt uit hoe een dergelijke HLES-implementatie kan worden ontwikkeld. Ten eerste worden een aantal aanpassingen en uitbreidingen van de meest recente HMES-implementatie voorgesteld. De HLES bestaat uit een herbruikbare kern, domeinspecifieke modellen en toepassingsspecifieke beslissingsmechanismes. De geïmplementeerde aanpassingen aan de kern laten de HLES toe om ondersteuning te bieden voor mobiele hulpmiddelen (bv. vrachtwagens), groepering ('batching') en het gelijktijdig toewijzen van meerdere hulpmiddelen ('multi-resource allocation'). De thesis presenteert ook de nodige modellen en

beslissingsmechanismes voor alle entiteiten die relevant zijn in het kader van cross-docking. Met behulp van simulatie-experimenten werd aangetoond dat de holonische technologie toepasbaar is en dat het ontwikkelde systeem werkt zoals bedoeld.

Ten tweede worden er, om de cross-docking HLES in het vinden van goede globale oplossingen te ondersteunen, twee planningssystemen ontwikkeld: om de ritten te plannen en om de vrachtwagens toe te wijzen aan laadkaaien. Er wordt eveneens een samenwerkingsmechanisme tussen het holonisch online controlesysteem en deze planningssystemen voorgesteld. Aangezien de ontwikkelde planningssystemen gebruik maken van een vereenvoudigd en benaderend model van de werkelijkheid, zijn de planningen die door deze systemen worden gegenereerd (deels) onuitvoerbaar. De HLES is echter in staat om verschillende vereenvoudigingen te compenseren. Bovendien pakt de HLES ook afwijkingen van de planning aan en maakt het zo de planningsaanpak meer robuust. Simulatie-experimenten bevestigen dat de samenwerking tussen de HLES en de externe planningssystemen de prestaties verbetert.

Abbreviations

ACO	Ant Colony Optimization
AS/RS	Automated Storage and Retrieval System
ASN	Advance Shipping Notice
B&B	Branch-and-Bound
BDI	Belief-Desire-Intention
CEP	Courier, Express and Parcel
D-MAS	Delegate Multi-Agent System
DDAP	Destination-Door Allocation Problem
DRS	Dependency Ranking Search
EA	Evolutionary Algorithm
EDD	Earliest Due Date
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
FCFS	First-Come, First-Served
FSA	Finite State Automaton
GA	Genetic Algorithm
GDP	Gross Domestic Product
GRASP	Greedy Randomized Adaptive Search Procedure
HLES	Holonic Logistics Execution System
HMES	Holonic Manufacturing Execution System
ICA	Imperialist Competitive Algorithm

LES	Logistics Execution System
LTL	Less-Than-Truckload
MAS	Multi-Agent System
MAUD	Modified All-Unit Discount
MCFP	Minimum Cost Flow Problem
MES	Manufacturing Execution System
MIP	Mixed Integer Programming
MIQP	Mixed Integer Quadratic Programming
MOM	Manufacturing Operations Management
MST	Minimum Slack Time
PHSP	Parcel Hub Scheduling Problem
PLC	Programmable Logic Controller
PROSA	Product-Resource-Order-Staff Architecture
PSO	Particle Swarm Optimization
QAP	Quadratic Assignment Problem
RFID	Radio-Frequency IDentification
SA	Simulated Annealing
SBSA	Simulation-Based Scheduling Algorithm
SCADA	Supervisory Control And Data Acquisition
SCM	Shipping Container Marking
SPT	Shortest Processing Time
TS	Tabu Search
TSS	Truck Scheduling System
UPC	Universal Product Code
VRP	Vehicle Routing Problem
VRPCD	Vehicle Routing Problem with Cross-Docking
VRSS	Vehicle Routing Scheduling System
WMS	Warehouse Management System

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background and motivation

Logistics as a field of study finds its origin in military science. Since even the strongest army cannot survive very long without the timely supply of food and weapons, a good logistic preparation strongly affects the outcome of a war. This was already pointed out by the Greek historian Thucydides (460 until 400 BC) in his description of the 10-year Trojan war, but it was only until Napoleon introduced the function of 'Maréchal de Logis' that the word logistics really came in use [160].

Briefly explained, logistics is concerned with having the right item in the right quantity at the right time at the right location for the right price [123]. It is the management of the flow of goods from the point of origin to the point of consumption. Logistics is not only involved with transportation, but also includes warehousing, material handling, packaging and labeling, etc.

Logistics is important in our daily lives. Supermarkets, pharmacies, restaurants, music festivals or sport events, they all need logistics for their activities. Logistics plays also a key role in the economy, as commerce and industry rely crucially on the supply of raw materials and finished products. Without logistics, everything would come to a standstill, resulting in an economic disaster. Moreover, logistics also has a direct contribution to the economy and employment. The logistics industry in Europe accounts for at least 10 % of the gross domestic product (GDP) [94], while the global logistics industry is estimated at roughly € 5.4 trillion, or 13.8 % of global GDP [66].

As a result of globalization of production and trade, increased outsourcing and offshoring, demand for transportation and logistics has risen during the last years. Moreover, this demand will still rise in the coming decades. For instance, freight transport is expected to grow by 80 % by 2050 [94]. At the same time, logistics has become more and more complex. Reasons are increased transportation regulations, globalization leading to tough international competition, the markets that become customer-driven (smaller volumes of customized products, shorter product life cycles), the implementation of manufacturing strategies like just-in-time and lean manufacturing, etc.

The logistics industry also has to face some other important challenges. First of all, the logistics sector should reduce its environmental impact. The sector highly depends on fossil fuels and is responsible for a large amount of the emission of greenhouse gases and particulates. For instance, the transport sector is considered responsible for 22 % of global CO_2 emissions and the fuel demand is expected to increase by nearly 40 % by 2035 [43]. The fast growth of freight transport also contributes significantly to road congestion, certainly in densely populated areas like Western Europe. This can result in high costs. For instance, the congestion costs in Europe amount for about 1 % of GDP [96]. The extension of the available transport infrastructure could address this, but the social support for such extensions is lacking. To stay competitive, manufacturers continuously try to reduce costs. As logistic costs account for 10 to 15 % of the final cost of the finished product [66], also the logistic costs should be reduced where possible. A reduction of energy costs is one aspect of this concern which is also linked to the environmental challenge. Another challenge is the recruitment of competent staff members in the logistics sector, as many tasks are repetitive and not very attractive. Likely, there will be a shortage of staff (in Europe) and the same amount of work will have to be performed with 25 % fewer people [42].

These issues can be (partially) addressed by increasing the efficiency of the logistic operations. One aspect hereby is the increase of the average load of the transport vehicles, which is currently quite low. For instance, in 2010, as many as 23.9 % of all vehicle-km of heavy road goods vehicles in the EU involved an empty vehicle [158]. A well-known solution is backhauling, i.e. picking up freight in the neighborhood of the destination in order to prevent an empty return trip. Backhaul loads can be found with the help of a freight brokerage service. The backhauling solution is not always successful. The delay and costs caused by the pick-up and delivery of backhaul loads can be too high. Another approach is consolidation: the combination of several small shipments into a single large shipment. By improved consolidation, it should be possible to use fewer vehicles, each carrying more freight. A logistic strategy that enables (rapid) consolidation is cross-docking. In so-called cross-docks, freight with the

same destination is consolidated, with little or no storage between unloading and loading of the goods.

Compared to traditional distribution, cross-docking operations are more difficult to manage. For instance, a better information flow is necessary for the coordination between inbound and outbound vehicles. Cross-docking poses complex and challenging problems, during the design phase as well as during operations. Cross-docks can be subject to different organizational and management approaches and different objectives can be aimed for. Particularly, cross-docks have to operate today in an uncertain and dynamic environment, among others due to a tough competition in the transport and logistics sector and ever-increasing traffic (congestion). Dealing with uncertainty is important and flexibility becomes a major topic. Cross-docks should be able to operate efficiently in disturbed and changing environments. Unrealistic assumptions and too rigid approaches prevent an efficient cross-dock operation. As the operational control of a cross-dock is a going concern, 'one-shot optimization' is not sufficient. Because of these complicated problems, it is worthwhile to consider approaches that proved to be useful in other domains.

In this thesis, concepts and approaches which are successful in the manufacturing domain are transferred to the field of logistics and especially cross-docking. In manufacturing, a production manager can be supported by a Manufacturing Execution System (MES). This software system is responsible for the real-time execution of the production. It handles the internal material flow in a manufacturing system and has to be capable to cope with disturbances like rush orders and machine breakdowns. A Holonic MES or HMES makes use of the concepts and principles of the holonic manufacturing paradigm [13, 125, 212] to organize the manufacturing control. Holonic architectures try to combine the high and predictable performance promised by hierarchical systems with the robustness against disturbances of heterarchical systems [24, 210]. An HMES can also cooperate with a scheduling system [203, 215]. The control system follows the schedule when it performs well, otherwise the autonomous decision making mechanisms of the HMES execute alternatives that resemble the original schedule. An HMES tries to improve the responsiveness, proactiveness, robustness and flexibility of a production system, properties that are also of high interest to manage cross-docking operations. This thesis will apply the same concepts to improve the coordination and control of cross-docks.

1.2 Goals and overview of the thesis

The goal of this thesis is the development of a Logistics Execution System (LES) for cross-docking. To this end, the concepts and principles of the holonic manufacturing paradigm will be applied. The thesis describes how a Holonic LES or HLES is developed, starting from a state-of-the-art Holonic MES realized by the MACC research group[1]. In addition, simulation experiments (on a limited scale) are executed to verify some of the benefits promised by the holonic paradigm. These experiments show that the HLES concept is a valuable option for the development of a cross-docking LES.

Similar to an MES, an LES is responsible for the real-time coordination and control of the logistic operations. An LES should combine a high performance with robustness against disturbances. Robustness against disturbances means that the degradation in performance caused by disturbances (like truck breakdowns or delayed trucks) is as small as possible. Conversely, if new opportunities arise, the performance should increase as much as possible. To obtain a robust control system, the holonic concepts and principles of the HMES are applied. Several adaptations and extensions to the basic HMES functionality (e.g. multi-resource allocation) are realized in order to account for specific properties of the logistics domain. By cooperating with one or more external scheduling systems, this HLES aims to have a predictable performance (in relation to the schedule). Moreover, the schedules provided by these scheduling systems are optimized with respect to global performance measures, for instance throughput or total tardiness. In this thesis, two scheduling systems for cross-docking are developed. Both systems make use of a metaheuristic approach to obtain good quality results. Further possible (minor) improvements to the proposed solution methods are not considered, as the thesis is more concerned with the cooperation between these scheduling systems and the HLES.

In the next chapter, the concept and principles from the HMES will be explained from an as general as possible logistic point of view. In the following chapters, the focus will be on cross-docking. The thesis is organized as follows.

Chapter 2 describes the basic principles of the Holonic Logistics Execution System. These concepts and ideas, previously applied in the manufacturing domain, are explained in detail from a logistic point of view. The chapter introduces holonic systems and multi-agent systems before describing the main aspects of the software architecture of the HLES: the PROSA reference architecture and the 'delegate MAS' pattern. By using this pattern, the HLES is able to provide a view on the expected short-term future of the system.

[1]Multi-Agent Coordination and Control (MACC) research group, Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300A, 3001 Leuven, Belgium.

Chapter 3 explains the cross-docking concept and gives an extensive overview of the available literature about this logistic strategy. The discussed papers are classified based on the problem type (e.g. layout design). Limitations, gaps and opportunities in the field of cross-docking research are made visible.

Chapter 4 proposes two scheduling approaches related to cross-docking. The HLES will cooperate with these approaches in order to improve its performance. The first approach solves the truck scheduling problem which assigns the inbound and outbound trucks to the different dock doors of a cross-dock in an optimal way. The second approach tackles the so-called vehicle routing problem with cross-docking. This problem is concerned with the assignment of orders to trucks: pick-up at the orders' origin and delivery to their destinations after consolidation at the cross-dock. Both problems are formulated as mixed integer programming models and heuristic methods are developed to solve these problems in a reasonable amount of time.

Chapter 5 presents the adaptations and extensions to the basic HLES functionality described in Chapter 2. It describes in detail how the HLES offers support for mobile resources, batching and multi-resource allocation and how cooperation with the scheduling approaches from Chapter 4 is possible. This chapter also presents domain-specific models and application-specific decision mechanisms of the entities relevant for the cross-docking application.

Chapter 6 evaluates the proposed Holonic Logistics Execution System by means of simulation. First, the simulation platform and the experimental set-up are described. Then, the results of some specific scenarios are described to show the value added of the proposed holonic control system. To demonstrate the benefits of the cooperation with the proposed scheduling approaches, experiments are carried out to compare different control modes (without or with (re)scheduling).

Chapter 7 summarizes the work presented in the previous chapters and gives the general conclusions. Moreover, specific suggestions for future work are formulated.

1.3 Research contributions

The main contributions of this thesis can be summarized as follows:

A review and classification of the existing literature about cross-docking
This thesis gives an extensive overview of the available literature about cross-docking. The discussed papers are classified based on the problem type that is tackled (ranging from strategic or tactical to operational problems). The thesis

also presents characteristics that can be used to differentiate between alternative cross-dock types, such as the cross-dock shape. For the papers included in the literature review, these characteristics are listed in detail.

The development of a scheduling method for the truck scheduling problem The thesis defines a truck scheduling problem which is concerned with the assignment of trucks to the different dock doors of a cross-dock. In contrast to many approaches in the literature, both inbound and outbound trucks are considered and can be assigned to multiple dock doors. The problem is formulated as a mixed integer programming model and a heuristic method is developed to solve it in a reasonable amount of time.

The development of a scheduling method for the vehicle routing problem with cross-docking This thesis also defines a vehicle routing problem with cross-docking. This problem is concerned with the assignment of orders to trucks. Both the pick-up of the orders at their origins and the delivery to their destinations are considered. To define the problem formally, a mixed integer programming model is presented. A heuristic method is developed to solve the problem in an acceptable amount of time. The thesis also describes how the heuristic approach can be adapted for rescheduling purposes.

Application of PROSA and delegate MAS in the logistics domain The PROSA reference architecture was originally developed for manufacturing applications. Together with the delegate MAS pattern, it targets manufacturing execution systems and has been applied in several industry-academia cooperative research projects. This thesis shows that PROSA and delegate MAS can also be applied for logistic applications, in a domain well beyond the original application range.

The development of a holonic on-line control system for cross-docking The thesis explains how an HLES implementation to coordinate and control the cross-docking operations can be developed. First, adaptations and extensions to the HMES functionality are described, as the HLES has to offer support for mobile resources, batching and multi-resource allocation. Secondly, the thesis proposes domain-specific models and application-specific decision mechanisms required to implement a functioning prototype. Based on the generated short-term forecasts, better (informed) decision making is possible and the HLES cannot only work reactively, but also proactively.

The design of a cooperation mechanism between the holonic on-line control system and scheduling systems This thesis describes how the cross-docking HLES can cooperate with two scheduling systems based on the developed scheduling methods. By this cooperation, the HLES is supported in finding good global solutions (e.g. with efficient batching). On the other hand, the HLES is able to execute the provided schedules, while accounting for deviations and possible simplifications. The cooperation mechanism for the vehicle routing problem is based on existing work, while the cooperation mechanism for the truck scheduling problem is a new development.

Testing of the developed HLES prototype Based on the presented approach to develop a cross-docking HLES, a research prototype is implemented. This prototype is tested on a simulation platform to show that the implemented system works as intended and to demonstrate the value added of the proposed approach. The cooperation between the HLES and the scheduling systems is also evaluated in several simulation experiments.

Chapter 2

Holonic Logistics Execution System foundations

This chapter describes the basic principles of the Holonic Logistics Execution System (HLES). The development of this system does not start from scratch, but can build on previously developed concepts and technology. In order to make this thesis self-contained, the foundations on which the Holonic Logistics Execution System relies are described in this chapter. Several of these concepts and ideas are previously applied to the manufacturing domain, but will be explained here from a logistic point of view. Starting from the explanation of a Manufacturing Execution System (MES), Section 2.1 describes what is understood in this thesis by an LES or Logistics Execution System. In Section 2.2, the main aspects of the software architecture of the HLES are described, which combines the 'Product-Resource-Order-Staff Architecture' (PROSA) and the 'delegate Multi-Agent System' pattern (delegate MAS or D-MAS). This section also introduces holonic systems and multi-agent systems. Next, Section 2.3 presents the intelligent product concept and its relations with the HLES technology. The applicability of the technology and a more general description of PROSA are discussed in Section 2.4. To develop an HLES, models are required of the relevant real-world entities. A multimodel formalism is used to model these entities. The basic ideas of this formalism are explained in Section 2.5. Finally, Section 2.6 concludes this chapter.

2.1 Logistics Execution System

Manufacturing can be described as the process that takes place in a factory and in which finished goods are produced out of raw materials and components by using machines and labor. To organize the different manufacturing activities, control actions have to be taken at various levels. At the lowest level(s), control actions deal with the control of the available hardware (machines, automated devices, etc.) and personnel, for instance to ensure the hardware is functioning within its normal operating range. This includes sensing and handling of the products. At higher levels, planning and scheduling methods can be applied in order to make use of the available resources in an efficient way. *Manufacturing Execution Systems* are situated in between these levels and are responsible for manufacturing control. *Manufacturing control* can be defined as "the decision making activity concerned with the production planning and control, including the short-term and detailed assignment of operations to production resources" [78]. Routing of the products through the production system, resource allocation, disturbance handling and tracking and tracing of the products are all tasks of an MES. Due to the nonlinear nature of the production environment, uncertainties (e.g. variable processing times) and unexpected events (e.g. late deliveries or machine breakdowns) in the production processes and the combinatorial growth of the decision space, this is a daunting task [193]. Moreover, several (possibly conflicting) objectives have to be taken into account and as manufacturing control is a going concern, 'one-shot optimization' is not sufficient [202].

It is also said that Manufacturing Execution Systems have to fill in the 'gap' which exists between Enterprise Resource Planning (ERP) systems and the physical processes [83, 176]. According to the ANSI/ISA-95 standard, MES systems are situated on the 'Manufacturing Operations and Control' level (level 3) of the functional hierarchy model [58]. This level is also called the 'Manufacturing Operations Management' (MOM) level [59]. Figure 2.1 gives an overview of the various functional levels. The levels below correspond to the low-level control of equipment. Level 0 corresponds to the actual physical processes. The activities involved in sensing and manipulating the production processes are situated on level 1. Level 2 includes the monitoring and controlling of the physical processes (e.g. by programmable logic controllers (PLC) and supervisory control and data acquisition (SCADA) systems). The fourth level is the planning at office level and includes business-related activities. ERP systems are for instance situated on this level. Level 3 is then concerned with production operations management, maintenance operations management, quality operations management and inventory operations management [59]. Most Manufacturing Execution Systems (including the HMES) focus on production control related issues. Production operations management takes

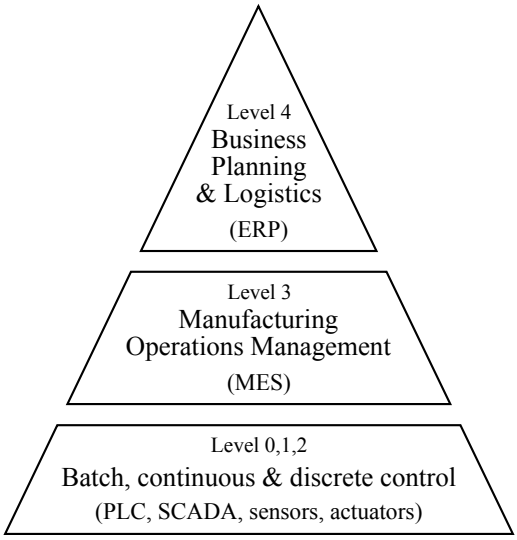


Figure 2.1: Functional hierarchy model (adapted from [83, 164]).

care of the necessary activities to elaborate products with the required costs, quality and timeliness, and includes detailed production scheduling, production dispatching and production execution management. Detailed production scheduling makes use of the (middle/long-term) 'production schedule' established at level 4 and takes the local situations and resource availabilities into account to obtain a 'detailed production schedule' (e.g. finite capacity planning). Next, this 'detailed production schedule' is used by the production dispatching activity to dispatch production to equipment and personnel by means of 'production dispatch lists'. These lists are then used by the production execution management activity to steer the production by sending operational commands to the lower levels. Note that the delimitation between these levels and their activities is not absolute and will differ in different companies. More information about manufacturing execution systems and their roles can for instance be found in [46, 83, 154, 164].

Manufacturing activities are only one aspect of the supply chain, which includes also *logistics*. The Council of Supply Chain Management Professionals (CSCMP) defines logistics as "the process of planning, implementing, and controlling procedures for the efficient and effective transportation and storage of goods including services, and related information from the point of origin to the point of consumption for the purpose of conforming to customer requirements" [220]. The management of this process is involved with operational, tactical

and strategic decisions and usually includes activities as fleet management, warehousing, materials handling, logistic network design, etc. [220]. The logistics domain is characterized by the following properties[1]:

Large decision space In most logistic problems, many different decision variables need to be taken into account [153]. For instance, to which truck will an order be assigned in combination with which driver(s), is it possible to combine orders in one cargo, can multimodal transport be used, etc.

Highly constrained decision space The decision space is also highly constrained [153]. These constraints include physical constraints (e.g. vehicle capacity, available storage space, allowable driving time, etc.) and business objectives (e.g. on time delivery, efficient use of vehicles, etc.). Note that these constraints can be conflicting.

Large scale Logistic systems usually have a large scale, both in terms of physical distribution and number of orders and logistic resources [88].

Nonlinearity In the logistics domain, effects are not proportional to causes. For instance, multiple trucks can be delayed while waiting for one belated truck, causing many tardy deliveries.

Dynamics The logistics domain is a dynamic domain [88]. New orders arrive during execution and have to be incorporated in existing plans. Disturbances and unexpected events (for instance road accidents) can occur, invalidating the current plans and requiring decisions to adapt to the new situation. In order to be able to react quickly, real-time data has to be collected and processed. This also implies that logistic problems consist of going concerns and 'one-shot optimization' is not sufficient [202].

Uncertainty A related characteristic is uncertainty [88, 153]. Decisions have to be taken based on uncertain and incomplete (or even wrong) information. For instance, due to the ever-increasing traffic, travel times can be very variable. The properties of the goods (e.g. weight) are - although indicated by the customer - usually also uncertain.

Decision making across company borders Logistic operations are nowadays controlled by multiple decision makers across company borders. This requires a good cooperation between the involved partners. For instance, the carrier should agree upon a pick-up time with the shipper and a delivery time with the consignee. Note that the different decision makers can have conflicting interests.

[1] Most of these properties are also characteristics of the manufacturing domain.

Severe competition Logistic firms face a fierce competition, which is typical for a commoditized and labor-intensive (instead of knowledge-intensive) sector [145].

If the main focus of the level 3 control system is not about production activities, but more about logistic operations, the term *Logistics Execution System* or LES is used. The Material Handling Institute (MHI) employs the following definition: "Logistics Execution Systems manage inventory, space, material handling equipment, labor and transportation resources to assure timely, error-free fulfillment and visibility of order status throughout the supply chain" [69]. While in general an LES considers also maintenance and quality, most Logistics Execution Systems (including the HLES considered in this thesis) focus on the organization of the logistic operations. These operations include all kinds of material handling operations at logistic nodes (e.g. distribution centers or cross-docks) and the transportation of goods between these nodes.

Logistics Execution Systems are less frequently used than Manufacturing Execution Systems. First, this can be (partially) attributed to the fact that logistic operations are less automated than manufacturing operations, which makes it more difficult to control and to adjust the operations by a computer system. Secondly, the various resources to perform the logistic operations are not grouped together at one location (like at a factory for manufacturing operations), but are spread over various logistic nodes and the links between these nodes. This also complicates the control of these logistic resources. However, improvements in information and communications technology make this less of a problem. For instance, GPS technology makes it easier to track goods and resources and on-board computers (with GPRS connection) allow for data exchange (e.g. status information, such as fuel level, or instructions) between the various logistic resources in transit and the back office. Thirdly, the organization of logistic operations involves cooperation between different partners (suppliers, carriers and customers), which makes this task a job more suited for human operators. However, ICT developments (like advance shipping notice (ASN) via electronic data interchange (EDI), RFID scanning, etc.) make an electronic information flow between the partners possible and this allows the logistic operations to be organized by a computer system.

In the context of this thesis, the task of a Logistics Execution System is to coordinate and control the behavior of the various logistic resources. This task consists of the following activities:

Operation determination For all requests from the planning at office level (level 4), the LES has to decide which operations are required to fulfill these requests. For instance, for a transportation request, the LES has to

determine if orders be directly transported from origin to destination or have to be transported via one or more intermediate nodes.

Resource allocation All (transportation and material handling) operations have to be assigned to available resources, i.e. the LES has to determine which resources and when these resources will perform the necessary operations (eventually in accordance with a provided schedule). This allocation is constrained by technological limitations and capacity constraints.

Execution supervision This activity involves handling of disturbances and unforeseen events (e.g. traffic jams or truck breakdowns) by reassigning operations. This reassignment can be automatic (the LES takes the initiative) or on request. In order to be able to react automatically, the LES collects information about the current state of the logistic system.

Tracking & tracing The LES is responsible for tracking and tracing all goods and the various mobile resources (trucks, trains, ships, ...). All relevant data are monitored and temporarily or permanently stored.

Based on the aforecited characteristics of the logistics domain, the following quality requirements[2] suggested for MESs [52, 125, 197, 218] are also of interest for Logistics Execution Systems:

Robustness The LES should be able to deal with disturbances (e.g. truck breakdowns) and uncertainty (e.g. variable processing times). Robustness[3] means the persistence of the characteristic behavior (and performance) of a system under perturbations or conditions of uncertainty [181]. The LES is robust against disturbances if the degradation in performance caused by disturbing events in the logistic system is as small as possible.

Reconfigurability This is the ability of the LES to dynamically change its configuration in a timely and cost-effective manner, usually to respond to dynamic changes. This includes the need to keep all related information sufficiently consistent, coherent and correct.

[2] According to Weyns [225]: "Quality is the degree to which a system meets the nonfunctional requirements in the context of the required functionality. Quality attributes are nonfunctional properties of a software system such as performance, usability, and modifiability."

[3] Although most people have an intuitive understanding of robustness, the concept is difficult to define and many definitions are provided in the literature. The definition given here is generally formulated and relates robustness to performance, which is application-specific. In the context of scheduling, robustness is associated with robust pro-active scheduling or simply robust scheduling, which is explained in Section 5.4.3.

Flexibility Flexibility indicates whether new elements may be easily added to augment the existing level of functionality and whether changes to existing elements can be made.

Responsiveness or reactivity This is the ability of the LES to respond rapidly to disturbances and unexpected events that impact upon the performance. Appropriate adaptations need to be made in order to reduce this impact. This is a possible way to make the LES robust.

Proactiveness Another possibility to increase the robustness is proactiveness, meaning that the LES is able to anticipate. Without this ability, the control system is myopic and has limited possibilities to foresee the consequences of its actions. However, the unpredictability and nonlinearity of the logistics domain limit the possibilities of the LES to rely on its proactive abilities.

Performant control This refers to the influence the control system has on the performance of the logistic system. The LES has never full control and the performance is affected by uncontrollable events as traffic jams. However, by managing the logistic operations, the control system has a direct impact on the performance.

Nowadays, different partners of the supply chain enhance their cooperation by abandoning the companies' barriers that strictly separate the entities involved in a partnership, and by extending integration beyond the borders of a single company [112]. This implies that, on the MOM level (level 3), MES and LES systems have to cooperate.

2.2 Architectural design

This section describes the main aspects of the software architecture of the proposed Holonic Logistics Execution System. The architecture is the same as the architecture of the HMES. A general introduction to the HMES is given by Valckenaers and Van Brussel [193] and its architecture is described in detail by Saint Germain and Verstraete [165, 216, 218]. This architecture combines the PROSA reference architecture and the 'delegate MAS' pattern into a working control system. As PROSA is developed in accordance with the holonic manufacturing paradigm, some explanation about holonic systems will be provided in the next section. Section 2.2.2 will then give some background information about multi-agent systems, as the HLES is implemented with multi-agent technology. The PROSA reference architecture is described in detail in Section 2.2.3, while Section 2.2.4 explains the delegate MAS pattern.

2.2.1 Holonic systems

The term *'holon'* was introduced by Koestler in *The Ghost in the Machine* [99]. Two observations of how social and biological systems are organized motivated Koestler to propose this concept. The first observation was that complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms present. This observation was influenced by Simon's parable of the two watchmakers [177]. The second observation was that, although it is easy to identify sub-wholes or parts, 'wholes' and 'parts' in an absolute sense do not exist anywhere. The term 'holon' was then proposed to describe the hybrid nature of sub-wholes/parts in real-life systems. Holons simultaneously are self-contained wholes to their subordinated parts, and dependent parts when seen from the inverse direction. Or put more simply: a holon is something that is whole in itself as well as part of a greater whole.

Koestler also points out that holons are autonomous self-reliant units, which have a degree of independence and handle contingencies without asking higher authorities for instructions. At the same time, these holons are subject to control from higher authorities. The first property emphasizes that holons are stable forms and can cope with disturbances. The second property highlights that holons are intermediate forms, providing the proper functionality for the larger whole.

According to Koestler, a holarchy is then a hierarchy of self-regulating holons which function:

- as autonomous wholes in supra-ordination to their parts;
- as dependent parts in subordination to control on higher levels;
- in coordination with their local environment.

Based on these concepts, a new form of manufacturing control systems was proposed: *holonic manufacturing systems*. This new paradigm had to provide an answer to shortcomings of earlier control systems. Previously, the control architectures[4] of these systems had evolved from centralized via hierarchical to heterarchical architectures [52].

Centralized control *Centralized control architectures* [52] are characterized by a central computer that performs all planning and information processing and registers the activities of the whole manufacturing system. In this way, overall system status information can be easily retrieved from a

[4] A control architecture determines the interrelationships between the various control components and allocates the different decision making responsibilities (e.g. part routing and resource allocation) to specific control components [52].

single source. This ability to access complete global information also makes optimization a more realistic expectation. However, centralized architectures tend to have a poor responsiveness, reliability, modifiability and extensibility.

Hierarchical control The shortcomings of the centralized control architectures resulted in the development of *hierarchical control architectures* [24, 52, 210]. These architectures introduce 'levels' of control that have a specific functionality and are organized in a top-down approach. There are strict master-slave relationships between the levels. Control decisions are operated top-down, while status reporting operates bottom-up. The benefits of these architectures include fast response times, gradual implementation, redundancy and limited complexity of individual control modules. Despite these advantages, there are also many disadvantages. The rigid structure makes it very difficult to make unforeseen modifications and the increased coupling between the modules adversely affects modifiability, extensibility and fault-tolerance. As low-level modules have to consult higher levels in the hierarchy in case of a disturbance, the system's reactivity to disturbances is weak. Moreover, global decision making is often based on obsolete information.

Heterarchical control *Heterarchical control architectures* [24, 52, 210] are characterized by a flat structure. These architectures consist of distributed locally autonomous entities that cooperate with each other (without the master-slave relationship from hierarchical architectures) to make global decisions. This local autonomy requires that global information is minimized or eliminated. Advantages are enhanced modularity, reduced coupling between the modules and increased robustness against disturbances. A main disadvantage is the low predictability of heterarchical architectures, as it is difficult to operate according to a predefined plan. Also, there is no global optimization possible and consequently, a high performance cannot be guaranteed.

Holonic manufacturing systems [24, 210] were put forward as neither centralized, hierarchical nor heterarchical control systems could face the challenges the manufacturing world was confronted with. Holonic control systems try to combine the high and predictable performance promised by hierarchical systems with the robustness against disturbances and the agility of heterarchical systems by having characteristics of both architectures. To avoid the rigid structure of hierarchical systems, holonic manufacturing systems provide autonomy to the individual holons. This allows the control system to respond quickly to disturbances and to reconfigure itself to face new requirements. In order not to ban all hierarchy, which is essential to master complexity, holons work together

in 'loose' hierarchies. Such a hierarchy is different from a traditional hierarchy in that:

- holons can belong to various hierarchies;
- holons can form temporary hierarchies;
- holons do not rely on the correct functioning of the other holons in order to perform their tasks.

The relationship between different levels is not a master-slave relationship, but an advisory relationship.

To develop a holonic manufacturing system, the concepts developed by Koestler were translated into a set of appropriate concepts for manufacturing [13, 192]. More information on the holonic manufacturing paradigm can for instance be found in [13, 109, 125, 212].

As a holon is an autonomous entity that cooperates with other holons to achieve its goals, multi-agent systems seem very appropriate to implement holonic manufacturing systems. There are however two differences between 'holons' and 'agents' [13]. Firstly, a holon can contain one or more other holons, while an agent is not composed of other agents. Secondly, while agents are pure software entities, holons can include both hardware and software parts. Still, multi-agent systems are the most natural choice to implement holonic systems and will be discussed in the next section.

2.2.2 Multi-agent systems

In computer science, an *agent* can be seen as a computer system that is capable of independent action on behalf of its owner. To satisfy its design objectives, an agent does not have to be told explicitly what to do at any given moment, but can figure that out for itself [229]. There is however no universally accepted definition of the term agent in the multi-agent community. Multi-agent research is applied in many research areas (e.g. artificial intelligence, manufacturing and robotics) which all have their own focus, resulting in different perspectives on the concept of an agent. While there is a general consensus that *autonomy* is an important capability of an agent, there is little agreement beyond this. A possible definition is given by Wooldridge [229]: *"An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives"*. Autonomy means that agents can operate without the direct intervention of humans or others, and have some kind of control over their actions and their internal state [230]. In most applications, an agent will not have complete control over its

environment. It will have at best partial control, and can try to influence its environment.

In the multi-agent community, various attributes or properties are assigned to agents (e.g. mobility, benevolence, rationality, reactivity) [73, 230]. Based on these attributes, several types of agents are distinguished. Although there is no consensus on how to classify agents, two common types of agents that are distinguished are intelligent agents and situated agents. These two types will be shortly discussed in the following two paragraphs.

Intelligent agents

An *intelligent agent* (also called an autonomous intelligent agent) is an agent that is able to reason about a symbolic representation of its application domain. To be intelligent, an agent has some extra capabilities next to autonomy. Wooldridge suggests the following capabilities [229, 230]:

Reactivity Intelligent agents are able to perceive their environment, and respond on time to changes that occur in this environment in order to satisfy their design objectives.

Proactiveness Intelligent agents do not simply act in response to their environment, but are able to exhibit goal-directed behavior by taking the initiative in order to satisfy their design objectives.

Social ability Intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

In the intelligent agents approach, the environment mainly contains the other agents. Intelligent agents do not share information about the environment with each other.

A well-known model to design an intelligent agent is the belief-desire-intention (BDI) model proposed by Rao and Georgeff [155, 156]. According to this model, an agent has three mental attitudes. First, the agent has *beliefs* about its environment (itself and other agents). This represents the informational state of the agent. Note that the term belief is used rather than knowledge as what an agent believes may not necessarily be true. Using the possible-world semantics, these beliefs can be modeled as a set of belief-accessible 'worlds' (i.e. possible future states of the environment). This set contains all worlds that the agent believes to be possible from the current point in time (by making decisions about which action to perform). Second, *desires* represent the motivational state of the agent, i.e. the objectives the agent would like to accomplish. They

determine the goal-accessible worlds as a subset of the belief-accessible worlds. Third, *intentions* represent the deliberative state of the agent and are desires the agent has committed to achieve. Similarly, the intention-accessible worlds are a subset of the goal-accessible worlds.

Situated agents

The concept of a *situated agent* was introduced based on [229]:

- the rejection of symbolic representations and of reasoning about these representations;
- the idea that intelligent behavior is linked to the environment and is a product of the interaction between the agent and its environment;
- the idea that intelligent behavior emerges from the interaction of multiple simpler behaviors.

The term situated is used in order to stress that an agent is actually situated in a (software) environment with which it interacts. In early research, these agents are also called reactive agents, because they can be perceived as 'simply' reacting to an environment, without reasoning about it.

A situated agent has an explicit position in the environment and has local access to the environment. Each agent is placed in a local context which it can perceive and in which it can act and interact with other agents. The functionality of the multi-agent system then results from the agents' interactions in the environment, rather than from their individual capabilities.

A well-known example of a reactive architecture is the subsumption architecture developed by Brooks [31]. According to this architecture, an agent's decision making is realized through a set of task-achieving behaviors. Each behavior continuously perceives the environment and reacts to it by performing certain actions. These behaviors are implemented using finite-state machines. No symbolic representation or reasoning is used. The different behaviors are organized as parallel working layers. To resolve conflicts between actuator commands from different layers, the priority of these layers increases from bottom to top. Higher levels can subsume the roles of lower levels by suppressing their outputs. However, lower levels continue to function as higher levels are added.

Multi-agent systems

A *multi-agent system* or MAS is then a system that consists of various agents, which interact with each other. In order to interact, these agents require the ability to cooperate, coordinate and negotiate with one another [229]. Two types of interaction can be distinguished. In *direct* interaction, the agents communicate with each other by exchanging messages. This type of interaction is most common in multi-agent systems. In *indirect* interaction on the other hand, the agents communicate with each other through the environment. The agents exploit the environment in order to share information and to coordinate their behavior. This type of interaction is typically used in situated multi-agent systems.

Indirect interaction is also used in so-called 'stigmergic' multi-agent systems. Grassé [74] introduced the term stigmergy to explain nest construction in termite colonies. The concept indicates that individual entities interact indirectly through a shared environment. Individuals respond to the modifications made to the environment (by other individual entities) by also modifying the environment. In sign-based stigmergy, these modifications imply the deposition (and adaptation or removal) of marks to the environment [33]. A well-known example of such a mark is the pheromone trail created by foraging ants between a discovered food source and the colony's nest [147]. In multi-agent systems, two popular types of marks are (synthetic) pheromones [33] and computational fields (Co-Fields) [124]. Example applications of stigmergy are ant colony optimization (ACO) [55, 56], routing in telecommunication networks [22, 51] and path planning [171].

To develop a multi-agent system, two different aspects have to be considered [229].

Agent design This aspect focuses on the design of (the internal model of) the individual agent. The agents should be capable of independent, autonomous action to successfully carry out the tasks delegated to them. So, this aspect includes the design of the reasoning and reactive behavior of the agent. Reasoning refers to the capability of the agent to interpret symbolic representations of an application domain. Reactive behavior is the capability to respond in time to inputs from the environment or other agents.

Society design The focus of this aspect is on the interaction between the different agents. The agents should be capable of correctly interacting in order to carry out the tasks delegated to them.

The aforecited BDI model [155, 156] and subsumption architecture [31] are both concerned with agent design, as they offer guidelines for the design of an individual agent. The PROSA reference architecture (discussed in the next section) on the other hand can be considered as involved with society design. Although PROSA is a holonic architecture and describes the responsibilities of various holons and their interactions, PROSA implementations - including the proposed HLES - make use of multi-agent technology. In this way, PROSA determines how the different agents have to interact in order to accomplish their tasks.

The agents employed in the multi-agent implementation of the HLES can be considered as situated agents. They are situated in an environment and can perform actions in this environment, but these actions are not guaranteed to succeed. The functionality of the system results from the agents' mutual interactions and the interactions with the environment. Details about the design of the individual agents can be found in [165, 216, 218]. The environment also allows the agents to interact indirectly with each other. This indirect interaction is stigmergic: pheromones, which evaporate over time, can be deposited on (virtual) blackboards to make information locally available [193, 196]. Next to this indirect interaction, also direct interaction via message passing is employed.

The fact that these agents are situated agents does not imply they cannot reason. What is important however is that these agents do not need their own representation of the world to perform this reasoning. Each relevant physical entity will have a software counterpart in the environment that will represent this entity. This *single-source-of-truth* design makes it easier to maintain a correct and up-to-date representation and does not result in inconsistencies and incompatibilities between representations of different agents.

A nice overview of multi-agent systems and of applications in several manufacturing domains is given by Monostori et al. [136] and by Caridi and Cavalieri [36]. Examples of multi-agent applications for transportation are the 'Mars' approach [62] and the 'TeleTruck' approach [34]. A survey of multi-agent-based approaches to transportation and traffic management is for instance presented by Burmeister et al. [35] and by Davidsson et al. [49].

2.2.3 PROSA reference architecture

The proposed Holonic Logistics Execution System is based on the PROSA reference architecture [212, 231]. A *reference architecture* describes the mapping from various functionalities (which cooperatively solve the problem) onto software components and the data flows between these components [18]. A reference architecture is not an architecture itself, but can be used as the basis

for designing the system architecture for a particular system. For instance, the ADAptive holonic COntrol aRchitecture (ADACOR) [110, 111] can be considered as an instantiation of the PROSA reference architecture. Reference architectures are used in a specific (mature) domain and arise from experience [18, 231]. PROSA was originally developed for the manufacturing domain and based on experience in this domain, special attention was paid to [231]:

- separate the necessary elements, which are generic, from the optional elements, which can be domain specific;
- separate the structural aspects from the algorithmic aspects for resource allocation and process planning;
- separate resource allocation aspects and process specific aspects;
- enable the incorporation of legacy systems, or the introduction of new technology.

The PROSA reference architecture is developed in accordance with the holonic manufacturing paradigm. The basic components are holons and the architecture describes the responsibilities of the various holons and their interactions. The acronym PROSA stands for *Product-Resource-Order-Staff Architecture* and refers to the different types of holons. Three basic types of holons can be distinguished: product holons, resource holons and order holons. Each of them represents a separate concern in the application domain: process planning, resource allocation and logistics management. These basic holons can be aggregated into larger holons and specialization can be used to structure them. Staff holons are optional and can be added to provide the other holons with expert knowledge or to incorporate legacy systems. Figure 2.2 shows a module decomposition view[5] of the holonic reference architecture [218]. The depends-on relationships between the holons indicate that the various holons share data with each other.

The following paragraphs elaborate on the four different holon types.

Resource holon

A *resource holon* corresponds to a resource in the underlying domain (equipment, infrastructure elements and personnel). In a logistic context, this means that for instance all transport means (trucks, freight trains, cargo aircraft, ...) and material handling equipment (forklift trucks, conveyors, automated guided

[5] A module is an implementation unit of software that provides a coherent set of responsibilities. A module decomposition view describes the organization of the software as modules and submodules and shows how responsibilities are divided across these modules [41].

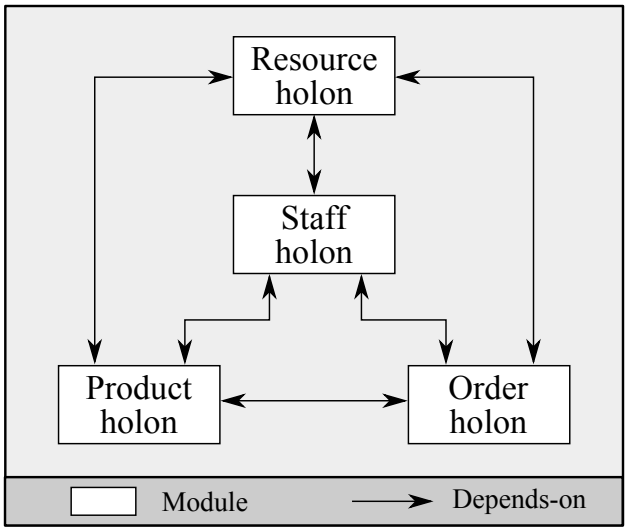


Figure 2.2: PROSA: module decomposition view.

vehicles, ...) will be represented by a resource holon. There will also be resource holons for other entities that are scarce and have to be shared (e.g. dock doors, pallet racks, floor space, etc.).

Each resource holon comprises the physical resource, together with a software part that controls this resource. It offers knowledge about processing capacity and processing functionality to the other holons and organizes and controls the usage of the physical resource. More concretely, a resource holon has the following responsibilities:

Reflection of reality A resource holon reflects its corresponding physical resource, i.e. contains information about the current state of the resource and expected future states. It should keep the reflection of the resource state synchronized with the actual resource state. Moreover, the holon has knowledge about the dynamic behavior of the physical resource and can answer what-if questions (e.g. what is the arrival time of a truck if it departs at a certain time).

Information provision A resource holon should be able to provide resource-related information to the other holons. This includes process information (e.g. possible operations), information about the local topology (to which other resource holons this holon is logically connected, see Section 2.2.4)

and about possible constraints (e.g. truck capacity, maximum cargo weight, etc.).

Maintaining a local schedule Each resource holon owns an agenda in which its future tasks/operations are recorded, based on requests from order holons. This can be seen as a reservation service which keeps track of the availability of the resource over time. Each operation to be processed by the physical resource needs to be reserved beforehand in this local schedule. To cooperate with the delegate MAS pattern (see Section 2.2.4), the local schedule is implemented as a (virtual) blackboard structure and applies an 'evaporation-refresh' mechanism. The reservations are placed on the blackboard as digital pheromones, which disappear (evaporate) over time. The reservations need to be regularly confirmed (refreshed) in order to keep them valid. This is a generic mechanism to handle changes: any outdated information simply disappears when it becomes too old.

Managing its local schedule The resource holons have local authority on how they organize (sequence or schedule) the various operations (from order holon requests), for instance by applying priority or batching rules. This local decision making is resource-specific and mainly depends on the performance settings of the resource.

Virtual execution This responsibility is a service for the order holons who can request information on the virtual outcome of an operation (e.g. quality and end time). Based on the local schedule (to decide how the operation can be fitted in between the already reserved operations) and its what-if functionality (to virtually execute the operation), the resource holon is able to provide accurate information.

Controlling the resource A resource holon controls the real-world resource by starting and stopping the (scheduled) operations and by monitoring the execution.

Several resource holons can be clustered together to form a bigger resource holon with its own identity. An example of such an *aggregated resource holon* is shown in Figure 2.3. A cross-dock holon consists of a temporary storage holon, one or more forklift holons, and one or more dock doors holon. The granularity of the aggregation will depend on the application and on the need of explicitly allocating these resources. For instance, it can be required to explicitly consider the forklift driver, and to see the forklift holon as an aggregate of a forklift truck holon and a forklift driver holon.

Specialization can be used to differentiate between the different kinds of resource holons. Figure 2.4 shows an example of such a specialization. Transportation

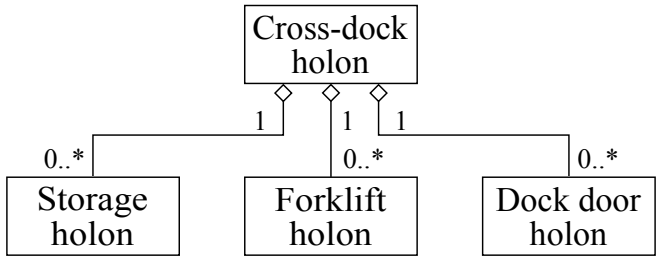


Figure 2.3: Example of an aggregated resource holon.

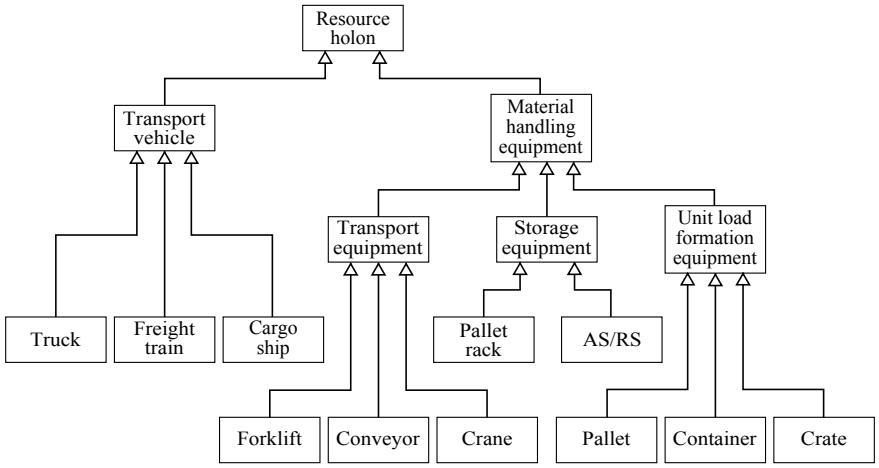


Figure 2.4: Example of specialization of resource holons.

vehicle and material handling equipment are both resource holons. Transport equipment, storage equipment and unit load formation equipment[6] are all kinds of material handling equipment. Pallet racks and automated storage and retrieval systems (AS/RS) are then examples of storage equipment.

[6]Unit load formation equipment is (reusable) equipment to contain materials in a unit load during transportation or storage.

Product holon

A *product holon* corresponds to a task type or order type[7]. To accomplish the task or to fulfill the order, a process (a sequence of (logistic) operations) has to be executed. The product holon then contains the knowledge on how instances of a specific task type (represented by order holons) can be executed by the resources, i.e. which operations are required to accomplish the task correctly and qualitatively. For instance, to deliver a package, the product holon knows that this package has to be picked up, transferred to the cross-dock, consolidated and eventually brought to its destination. The product holon also has information about constraints on or process parameters of these operations. For instance, if the package contains refrigerated products, the holon knows the allowable temperature range to which the package can be exposed during transportation. Note that a product holon only holds information about its order type, and not about individual order instances. The main responsibilities of a product holon are:

Maintaining process knowledge The product holons hold the necessary process knowledge to realize instances of their type. This includes, amongst others, process plans, process parameters and quality requirements. The product holons are responsible for keeping this information consistent and up-to-date, for instance if new operations are offered by (new) resources.

Determination of operation options A product holon informs the order holons about all possibilities for their next operation. Indeed, after completion of an operation, the order holon needs information about its next operation in order to accomplish its task. In its simplest form, the process plan is linear and the product holon supplies the order holon with only one possibility. In general, multiple options are possible and these alternatives depend on the current state of the order holon, i.e. based on the outcome of the previous operation. For instance, if the quality of an operation is insufficient, the product holon can decide the operation has to be redone. As another example, if a package with refrigerated products is exposed to high temperatures during its transportation, the holon can decide that the package should be disposed of instead of being delivered. The selection of one operation out of these options is the responsibility of the order holon itself.

Process information provision Just before a selected operation should start on a resource, the resource holon needs to know the desired process

[7]The term product holon originates from the manufacturing domain in which a task or order in general corresponds to the fabrication of a product. The terms task (type) and order (type) will be used interchangeably.

parameters (e.g. temperature during transportation in a refrigerator truck). The product holon is responsible for providing this process information to the resource holons. By providing this information just before the operation starts, the product holon is able to take the latest state of the order holon into account.

Similar to resource holons, product holons can be combined into an aggregated product holon which represents the combination of the corresponding process plans. For instance, the aforementioned product holon responsible for delivering a package could consist of three product holons: one to transport the package to the cross-dock, one to process the package inside the cross-dock and one to deliver the package to its destination. The aggregated holon delegates some of its responsibilities to these sub-holons, but is still in charge. This aggregation limits the complexity of the holons and allows an easy introduction of new product holons by combining other product holons.

Order holon

An *order holon* corresponds to a task (instance) or order (instance) that needs to be executed, e.g. the delivery of a package. Each order holon is closely linked to the product holon representing the corresponding task or order type. While a product holon can be linked to multiple order holons, each order holon will be associated with only one product holon. The order holon is responsible for handling the required resource allocations in order to accomplish the correct execution of its task. To this end, the order holon consults its corresponding product holon to find out which operations it needs to perform and searches for the proper resources and time slots to execute these operations.

In a logistic context, the order holons can often be associated with physical entities, i.e. the freight units that have to be transported (e.g. pallets). The order holon then consists of this real-world entity, together with a software part that controls the execution of the corresponding task. More concretely, an order holon has the following responsibilities:

Reflection of reality An order holon reflects the order instance, i.e. contains information about the current state of the order and the corresponding physical entity. This includes for instance the location of the order, the current operation being processed, the resource performing this operation, etc. The order holon is responsible for keeping the reflection of its state up-to-date with the actual state.

Searching solutions The order holons search for solutions[8] to execute their tasks. During their search, the order holons will consult their product holons to know the required operations and will virtually execute these operations (by using the virtual execution service of the resource holons) to check for resource availability.

Intention selection Each order holon evaluates the solutions it has found and chooses the most attractive solution (according to its performance measure) to become its intention.

Reserving its intention The order holon then informs the other holons about its intention by making the necessary reservations (future allocations) at the involved resource holons. As these reservations evaporate after a certain time, the holon has to confirm its reservation at regular time intervals.

Also order holons can be aggregated into an aggregated holon. For instance, several orders corresponding to freight that has to be transported can be aggregated into one batch in order to be transported together by a truck. Over time, an order can be part of multiple batches for different transport operations.

Staff holon

The three basic types of holons can be assisted by one or more staff holons. These holons can provide the other holons with expert knowledge about certain aspects of their decision making. For instance, a staff holon can give information on which containers can be batched on a freight train to the corresponding train holon and the concerned order holons. Note that the staff holons only provide advice and that the basic holons are still responsible for taking the final decisions. In this way, the concept of staff holons allows for the presence of centralized functionality in the architecture without introducing a hierarchical rigidity. This centralized functionality allows aiming for a good global performance, which is otherwise difficult to obtain as every holon tries to optimize its own (selfish) objective.

To obtain its advice, a staff holon may rely on centralized scheduling algorithms, human input, artificial intelligence methods, etc. Next to scheduling advice, the staff holon can for instance provide advice about route planning or the balanced loading of a cargo ship. In case of scheduling advice, the various order holons will attempt to execute (the relevant part of) the provided schedule. They will

[8] A solution corresponds to a sequence of resource allocations with corresponding start and end times.

deviate from the original schedule only if they find a significant better solution or the provided advice appears to be (or has become) infeasible. Detailed information on how the order holons can make use of a provided schedule can for instance be found in [203, 215, 216]. As a similar approach is used in this thesis, more information can also be found in Section 5.4.

Interactions between the holons

As indicated in Figure 2.2, the various holons interact and share data with each other. The main interactions between the basic holons are now shortly discussed.

Product-order The order holons interact with their corresponding product holon on how to accomplish the correct execution of their task by using certain resources. After (virtual) execution of an operation, the order holon passes information about the resulting state and about next possible resources to the product holon. Based on this information, the product holon provides the order holon with all possible next operations. For instance, after loading a container onto a trailer, the corresponding order holon consults its product holon to know the following operation that should be executed. Usually, multiple options are available, e.g. direct transportation to the final destination, transportation to an intermodal hub to be loaded onto a train or ship, etc.

Product-resource Product and resource holons share process related information. When generating a list of possible operations for an order holon, the product holon will interact with resource holons to know which operations the resources can perform. The other way around, the product holon provides the resource holon with technological aspects to correctly process an order, i.e. the necessary process parameters to perform an operation. For instance, if refrigerated products have to be transferred between two trucks in a non-cooled terminal, the product holon will indicate that this transfer should happen as fast as possible and impose a maximum transfer time.

Resource-order The resource and order holons mainly interact to reserve operations on the resources. To this end, the resource holons provide the order holons with the results of virtually executed operations and reserve capacity when requested. Once an operation is started, the resource holon also informs the order holons about the execution result and progress.

The desired coordination and control then emerges in a self-organizing[9] way from the interactions between the various holons.

Holon versus agent

As indicated higher, multi-agent systems seem very appropriate to implement holonic systems. So, the Holonic Logistics Execution System (HLES) is implemented with multi-agent technology. Although holons and agents are similar concepts, they are not equal. A holon is a more general concept, and an agent can be considered as one of the constituents of a holon.

A holon consists in general of a 'hardware' and a software part. This hardware part is the *physical entity* to which the holon corresponds. For instance, for all resource holons, their corresponding real-world resource is a part of the holon.

The software part can be divided into two software modules. A so-called *environment entity* module is responsible for reflecting the corresponding real-world entity. It mirrors the physical entity and can be considered as a 'software copy' of this entity. This module holds all relevant knowledge of its entity and keeps this information up-to-date. Importantly, this module does not include any decision making responsibilities.

All aspects related to decision making are part of the *agent* module[10]. Every holon takes decisions in order to achieve its goals. For instance, an order holon has to decide on what resources and when the necessary operations should be executed in order to accomplish its task.

The aforementioned responsibilities of the basic holons can be mapped onto these two software modules. For the resource holon, reflection of reality, information provision and maintaining a local schedule are responsibilities of the environment entity, the other responsibilities belong to the corresponding resource agent. Maintaining process knowledge for the product holon and reflection of reality for the order holon are responsibilities of the environment entity, the product and order agents take care of the other responsibilities.

Figure 2.5 shows the relations between the holon, agent and environment entity modules. A holon is the aggregate of an agent and an environment entity which share data with each other. The aggregate of all environment entities is the environment. As explained in Section 2.2.2, the various agents are situated in this environment and interact with it. The agents can perform actions in this

[9]Self-organization can be described as "the mechanism or the process enabling a system to change its organization without explicit external command during its execution time" [174].

[10]From here on, the terms resource agent, product agent and order agent will be used, referring to the agent module of respectively the resource, product and order holon.

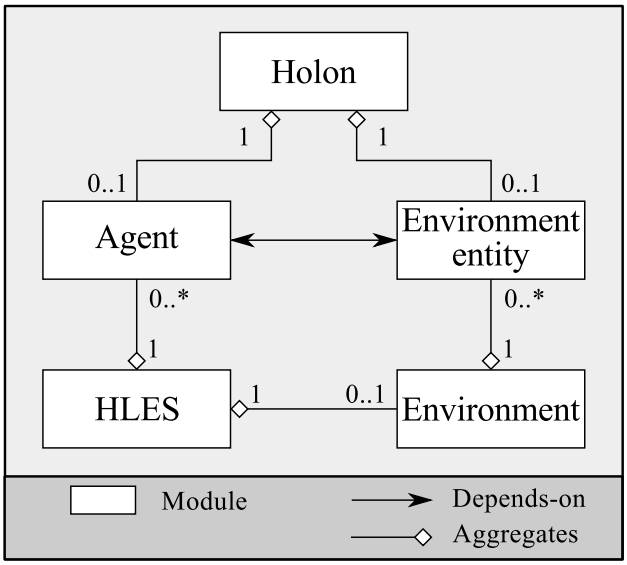


Figure 2.5: Holon versus agent: module view.

environment (e.g. starting a transport operation on a truck), but these actions are not guaranteed to succeed. The agents can also perceive the various entities in the environment. For instance, an order agent can observe information about the current state and capabilities of a resource. The environment also allows for communication between the agents. So, the environment contributes significantly in handling the complexity of the multi-agent system [87, 198, 217]. The HLES is then the aggregate of the environment and the various agents. Its functionality is the result of the agents' mutual interactions and the interactions with the environment.

Separation of concerns and single-source-of-truth

As indicated higher, during the development of the PROSA reference architecture, special attention was paid to *separation of concerns*. The three basic holons represent a separate concern of the underlying domain. An important characteristic of PROSA is for instance that resource allocation aspects and process specific aspects are separated. The issue of technical feasibility (executing a task in a technically correct and validated manner) is addressed by product and resource holons. Product holons are knowledgeable concerning the capabilities of resources that are relevant to them, but they

ignore resource capacity and availability aspects. The product holons inform the order holons about (all) technically correct manners to execute their task instance. Order and resource holons then address the issue of resource allocation. Within their solution space, identified by the product holons in cooperation with the resource holons, the order holons arrange the resource allocations and the execution of the required operations to accomplish the correct execution of their task.

Moreover, by dividing the software part of a holon into an agent and an environment entity, the concerns of decision making and reflection are also separated. This separates application-specific managing aspects from more generic and reusable domain models. Indeed, it is recognized that domains are more stable than requirements for applications in these domains [21, 86]. In a logistic context, the physical building blocks (pallets, containers, forklifts, conveyor belts, cargo aircraft, etc.) are highly invariant items from a coordination and control perspective. Generalizations (e.g. transportation vehicles and storage equipment) enjoy even more stability, while aggregation enables coping with larger subsystems (e.g. distribution centers).

By clearly separating several concerns, the easy reuse of software components in different applications is enabled [212]. This reduces the development time and costs of new applications. Moreover, this leads to higher-quality implementations of these components, resulting in increased reliability and performance and reduced maintenance costs.

As there is a choice-free entity for every entity in the world-of-interest[11] (i.e. the corresponding environment entity), this automatically leads to a *single-source-of-truth* design. The knowledge about a physical entity is maintained at one single instance, rather than spreading this knowledge across multiple components. This makes it easier to maintain correct and up-to-date information and simplifies software design. There will be no serious conflict amongst the several environment entities, as they reflect the real world which is integrated, consistent and coherent [201].

2.2.4 Delegate MAS

To obtain the desired functionality of the HLES, the PROSA agents have to coordinate their behaviors. The applied coordination mechanism is inspired by the food-foraging behavior of ant colonies and enables the agents not only to take the current situation into account, but also to consider the predicted

[11] The world-of-interest is the part of reality within a certain scope relevant for the application [217]. Scope in this context has two dimensions; it considers a certain *level of detail* and a certain *range*, i.e. a part of the world in which the system operates.

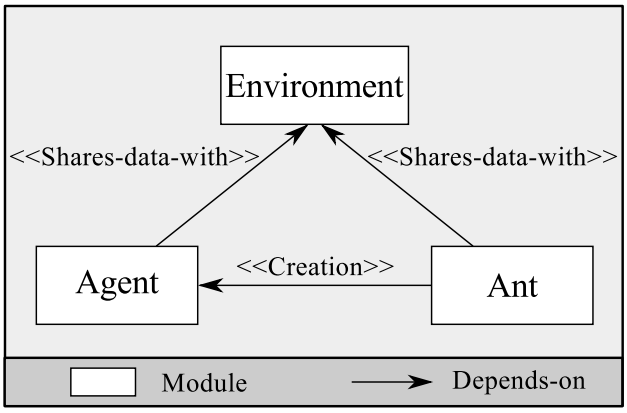


Figure 2.6: D-MAS: module decomposition view.

(short-term) future situation [81, 193, 196]. A more generic description of this approach is the delegate MAS architectural pattern.

An *architectural pattern* is a description of software components and their interactions, together with a set of constraints on these components and interactions which define a set of architectures that satisfy them [18]. *Delegate MAS* or *D-MAS* is an architectural pattern that allows an agent to delegate a responsibility to a swarm of lightweight agents. These agents perform particular activities to support the issuing agent in fulfilling its functions [87, 88]. An agent can simultaneously delegate multiple responsibilities, by applying the delegate MAS pattern for each of them. The agent may also use a combination of delegate MASs to handle just a single responsibility. Figure 2.6 shows a module decomposition view of the architectural pattern. There are three modules: the environment, the agent and the ant [218]. The depends-on relationships between the modules are refined. The agent and ant module share data with the environment module, while the relationship between the agent and ant module is a creation relationship.

The lightweight agents are called *ant agents* or simply *ants*, after their biological inspiration. These agents are lightweight in the sense that each ant may only perform a bounded computational effort within its bounded lifetime and has a bounded footprint (memory). They are responsible for executing a task that serves a responsibility of the issuing agent. Each ant is created and initialized by its issuing agent and travels autonomously through the (virtual) environment, starting from the location where the issuing agent resides.

Corresponding to the description used before, the *environment* is a software

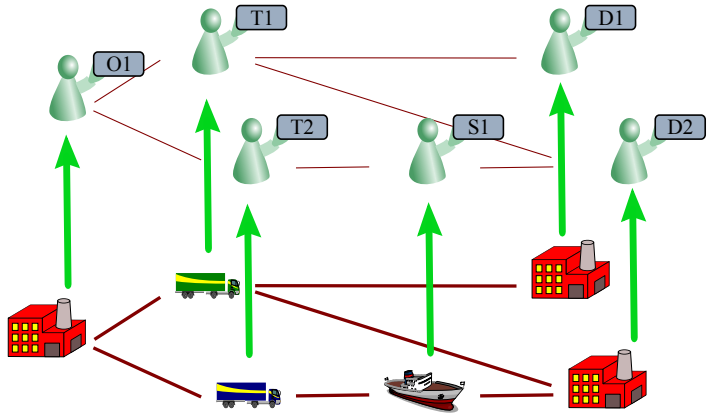


Figure 2.7: Example of a resource graph. The physical resources are shown together with their corresponding resource holons.

representation of the world-of-interest. To support navigation of the ants, the environment contains a directed graph constituted by the resource holons (nodes) and their logical connections (edges) [196]. A connection between two nodes indicates that an order can execute an operation on the second resource after completing an operation on the first one. Figure 2.7 shows a simplified example of such a resource graph indicating the options for the transportation of a container. There are resources corresponding to the origin (O1), destinations (D1 and D2) and transportation means (trucks T1 and T2 and cargo ship S1). As indicated by the connections, the container can be transported to D1 by truck T1 only. If D2 is the destination, the container can also be transported by T2 and cargo ship S1 as D2 is situated at a waterway. The topology may change over time and the resource holons are responsible for keeping the knowledge about their own connections up-to-date (see Section 2.2.3). The environment also enables indirect interaction based on stigmergy [79, 196]. The environment contains information spaces on which the ants can deposit, observe and modify information. These information spaces are local blackboards (which are part of the environment entities [218]) applying an 'evaporation-refresh' mechanism. The deposited pheromones have a limited lifespan and need to be refreshed regularly to prevent evaporation. The local schedules of the resource holons (see Section 2.2.3) are for instance stored on these blackboards.

An *agent* delegating a responsibility to a swarm of ants is responsible for maintaining the population size and the diversity of this swarm. The agent can choose the creation frequency and initialization for every ant type. The individual ants are not aware of these swarm properties. The agents can observe

and interpret the pheromones in the environment and adapt their behavior according to the results.

Three types of delegate MASs are distinguished in previous research: feasibility, exploring and intention delegate MAS[12] (see for instance [4, 50, 79, 152, 168, 193, 196, 235]). These types will also be employed in the Holonic Logistics Execution System and are discussed next.

Feasibility D-MAS

A resource holon delegates part of its 'information provision' responsibility to a swarm of so-called *feasibility ants*. These ants make global feasibility information (about the capabilities of the resource) locally available for the other holons. They put a kind of digital signposts on the blackboards enabling order holons to decide locally which routing options are available to them.

Feasibility ants are created by resource holons corresponding to an end point of the resource graph. The ants start at these nodes and traverse the graph upstream, i.e. in opposite direction of the orders. During their trip, they collect information on the capabilities (e.g. type of supported operations) of the visited resources. This information is also deposited on the local blackboards of the resource holons and merged with the already available information from other feasibility ants. In this way, every node contains information about which capabilities are reachable via its connected nodes (somewhat similar to routing tables in computer networks). This information permits order holons (or their corresponding product holons) to determine which part of their process plan can be executed downstream. This activity is performed at a regular frequency such that changes (both in resource capabilities and topology) become quickly visible throughout the system.

For instance, for the resource graph shown in Figure 2.7, feasibility ants are created by the resource holons corresponding to D1 and D2. They travel upstream towards O1 and collect information about the provided (transportation) activities. The ants deposit this information at the nodes they encounter. In this way, the information at O1 will for instance indicate that D1 can only be reached via truck T1.

Exploring D-MAS

The order holons are responsible for 'searching solutions' to execute their task. This responsibility is delegated to a swarm of *exploring ants* which will scout for

[12]The specific needs of an application determine which of these types are required.

possible feasible solutions (which account for resource availability and capacity).

Every order holon creates at regular time intervals exploring ants. These ants start at the current location of the order holon and travel downstream through the resource graph. During their journey, the task of the corresponding order is virtually accomplished by virtually executing the required operations on the resources. To this end, the exploring ants make use of the 'virtual execution' service of the resource holons to know the outcome of the operations. This service takes the expected resource availability into account. After the completion of each (virtual) operation, the ants consult the corresponding product holon to know their possible next operations ('selection of operation options' responsibility) and choose one of these operations. This operation is then virtually executed on the corresponding resource. Eventually, if its task is virtually completed, the exploring ant reports the discovered solution to the issuing order holon.

An example will further clarify this behavior. For instance, for the resource graph in Figure 2.7, a container has to be transported from O1 to D2. The order agent corresponding to this container will regularly send out exploring ants to search for solutions. An ant starts with retrieving the signposts at O1, placed there by the feasibility ants, and presents them to the associated product agent. In turn, the exploring ant receives the available routing options. For D2 as destination, both truck T1 and T2 are feasible options. The ant chooses one of these options, for instance T2, and starts its virtual journey. It queries the resource agent corresponding to T2 to virtually execute the required transport operation, starting from the container's release time at O1. The resource agent can consult the product agent to know the relevant process parameters and determines the virtual outcome of the operation by using its what-if functionality and its local schedule. The exploring ant updates its state (i.e. is virtually transported by T2), and in particular its virtual time, which is set to the estimated arrival time. Again, the ant collects the signpost information available at T2 and queries the product agent to know its possible next operations. There is only one option (being transported by S1) and the ant virtually executes this operation. Then, the exploring ant has virtually completed its task and reports the result (with the expected arrival time at D2) to its issuing order agent.

Intention D-MAS

By applying the exploring delegate MAS, the order holons have found solutions to accomplish their task. Each order holon evaluates its solutions and selects the most attractive solution to become its intention. The holon is then responsible

for 'reserving its intention' by informing the involved resource holons about its future visits. This responsibility is delegated to a swarm of *intention ants*.

When an order holon has selected an intention, it creates an intention ant. This ant behaves in the same manner as an exploring ant, except for two aspects. Firstly, intention ants do not have to consult the product holon for possible options and to choose between these options, but follow the route corresponding to the selected intention. Similar to the exploring ants, they will report the result of this trip to the order holon, while accounting for the consequences of changes in the system (e.g. truck breakdowns). Secondly, each intention ant informs the resource holons encountered on its journey about the order's intention. In other words, it makes the necessary reservations (future allocations) at the resources. As these reservations evaporate after a certain time, the order holon has to confirm its reservation at regular time intervals. To this end, the holon sends out intention ants at a regular frequency.

Resource intentions

The resource holons can also propagate intention information, i.e. information related to their own intentions (the reservations at their local schedule). Currently, no separate delegate MAS is used, but this propagation is also performed by the exploring and intention ants.

If an exploring or intention ant virtually executes an operation at a resource, the resource holon can indicate that this operation cannot be performed (at the requested time). In this case, the ant does not continue its journey, but returns on its steps and propagates information about the resource load backwards. This information can then guide other exploring ants (from the same order holon) to find solutions, for instance with a later release time. More concretely, the time period at which the requested operation cannot be performed according to the current load will be indicated at the local blackboards. During the backwards propagation, the indicated period is adapted at every resource by taking the execution time into account. In this way, the adapted period reflects the time interval that no execution should be performed at the current resource. If the execution time is not time-dependent, this can be simply done by subtracting the execution time from the start and end time of the interval.

To illustrate this, the previous example is continued (Figure 2.7). If an exploring ant fails to virtually execute the transport operation at S1, it will be informed about the time the container can be transported (for instance t_3). The exploring ant will then retrace its steps and travel back to O1 via T2. At both resources, the ant deposits information about this start time, but adapted based on the execution time of the resource (t_2 at T2 and t_1 at O1). If a next exploring

ant, departing from O1, uses this information by adjusting its virtual release time (to t_1) and follows the same path, it will arrive at S1 at the time it can immediately be transported (t_3).

Short-term forecasting

The combination of exploring and intention delegate MAS provides a view on the expected short-term future of the system. Both resource and order holons have short-term forecasts about their predicted execution. The order holons know the expected routings and resource allocations for their orders and the resource holons the predicted loads for the corresponding resources.

The resource holons receive the necessary information to calculate a short-term forecast of their utilization via the intention delegate MAS. Based on these forecasts (and their what-if functionality), they are able to give accurate answers to the queries from the exploring ants. This in turn allows the order holons to have a precise view on their short-term future. Note that the order holons create exploring and intention ants at regular time intervals, even after they have selected an intention. This allows them to react to disturbances and new opportunities and keeps the short-term forecasts up-to-date.

All short-term forecasts together can be seen as a 'dynamic[13] schedule' [81]. Figure 2.8 shows an example of such a schedule. In this way, these forecasts provide visibility of future actions, which is recognized as a valuable property [12]. This visibility allows for instance to identify potential capacity conflicts, permitting management to take action on time to avoid them. It also facilitates operators to see the bigger picture and to anticipate the impact of their decisions. Moreover, in the context of manufacturing, creating visibility on the shop floor is considered as one of the main goals of an MES [46].

These short-term forecasts can be used by the order and resource agents to take better (informed) decisions. For instance, as the order agent has an accurate view on its intention, it can make a well-considered decision whether or not to switch to one of its alternative solutions. The resource agents know their expected loads and can for instance decide to process a rush order or not based on how many reservations will be affected. As another example, the resource agent corresponding to a truck can, based on its expected load, decide when it is a good moment for maintenance or refueling.

For these forecasts to be valid and reliable, the order agents cannot continuously change their intentions. To this end, Hadeli [78] introduced socially-acceptable

[13] Dynamic in the sense that the schedule is regularly adapted according to the real-world situation.

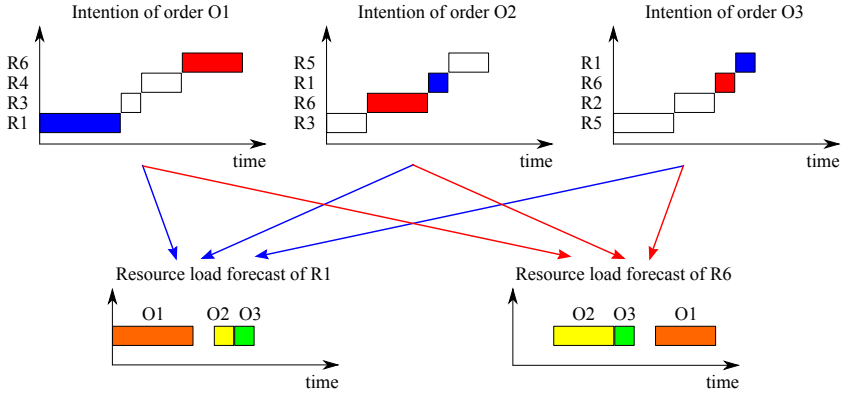


Figure 2.8: Example of generated short-term forecasts.

behaviors for the order agents. Firstly, order agents are only allowed to switch to a new intention if the performance increase is significant. Secondly, even if there is a significant gain in performance, the order agents adapt their intention with a given probability. Finally, the frequency at which order agents can change their intentions is limited. For more information, see for instance [78, 80, 82].

If the various agents (and their ants) belong to different organizations (for instance in a logistic context in which various companies are cooperating), it can be necessary to make use of trust mechanisms in order to obtain reliable short-term forecasts. Saint Germain [165] proposed a decision making pattern to integrate existing trust models, which are extensively studied, into the holonic control system. More information about this pattern can for instance be found in [165, 166, 168, 169].

2.3 Intelligent products

In manufacturing, resources have become more and more 'intelligent' over the years, for instance by the addition of microcontrollers, sensors and actuators. Moreover, they are usually connected to a local area network and have access to various data management services. Developments in automatic identification technologies (e.g. radio-frequency identification or RFID) can provide the means to also add intelligence to manufactured products, enabling truly intelligent manufacturing control systems [126]. This has led to the concept of *intelligent*

products. This section shortly discusses the intelligent product concept and how it relates to the described HMES and HLES technology.

McFarlane et al. [126] define an intelligent product as follows.

> An intelligent product is a physical and information-based representation of a product which:
>
> 1. possesses a unique identity;
> 2. is capable of communicating effectively with its environment;
> 3. can retain or store data about itself;
> 4. deploys a language to display its features, production requirements, etc.;
> 5. is capable of participating in or making decisions relevant to its own destiny.

Based on this definition, two levels of 'intelligence' can be distinguished [228]. The first level allows a product to communicate its status, i.e. information oriented. This corresponds to points 1 to 3 of the proposed definition. The second level allows a product to assess and influence its function in addition to communicating its status, i.e. decision oriented. This level covers points 1 to 5. An intelligent product consists of the physical product and an information-based counterpart (a software agent). Other definitions of an intelligent product are for instance given by Kärkkäinen et al. [95] and Ventä [214].

Meyer et al. [133] propose a classification of intelligent products based on three orthogonal dimensions (see Figure 2.9). First, intelligent products can be classified according to the *level of intelligence*. A first category of intelligent products is only capable of managing its own information. A second category can also notify its owner when there is a problem and a third category of intelligent products can completely manage its own life and is able to make all required decisions to accomplish this. A second dimension is the *location of intelligence*. The intelligence can be situated completely outside the physical product, at a different location, for instance at a server where a dedicated agent for the product is running. On the other hand, the intelligence can be embedded at the physical product itself. The third dimension is the *aggregation level of intelligence*. The intelligent product only manages information and decisions about itself, or is also aware of the components it is made of or contains. Meyer et al. [133] also discuss some technologies that (will) enable the intelligent product concept. Application domains are not only manufacturing, but for instance also supply chains, asset management and product life cycle management.

In a manufacturing context, the functioning of the HMES can also be explained as a result of the cooperation between intelligent products and intelligent

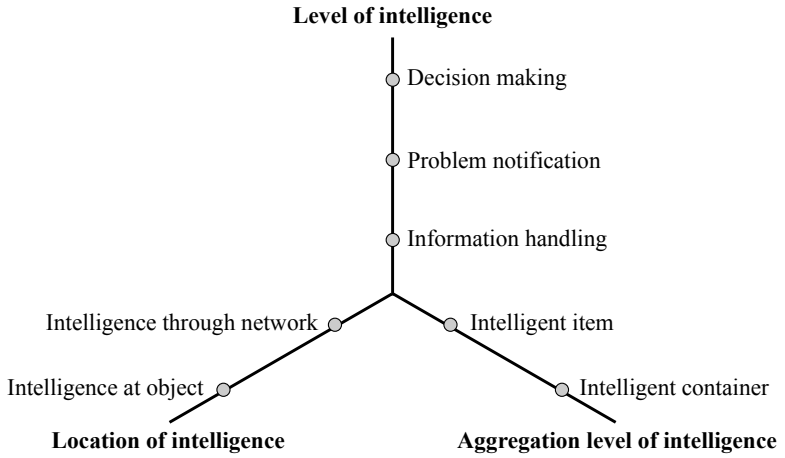


Figure 2.9: Classification of intelligent products (adopted from [133]).

resources. The intelligent products actively search for the operations needed to produce their corresponding products and have to decide which operations fit their needs best. These operations are offered by the *intelligent resources*, corresponding to the factory resources. In this way, an intelligent product corresponds to a PROSA order holon together with its corresponding product holon, and looks for the required operations to get its task fulfilled. An intelligent resource then corresponds to a resource holon. Figure 2.10 shows this correspondence. According to the classification of Meyer et al. [133], this kind of intelligent product has the following properties: intelligence through network, decision making and intelligent item.

Agents and beings

According to Valckenaers et al. [199], an intelligent product is the combination of an intelligent agent and an intelligent being. The *intelligent agent* is responsible for decision making and achieving objectives. On the other hand, the *intelligent being* reflects the corresponding real product and is restricted to provide functionality and services for which the corresponding reality provides adequate protection[14]. Any functionality or service that requires decision making, not imposed by reality, is delegated to the intelligent agent. Similarly, the intelligent resources consist of an intelligent agent and an intelligent being. This distinction between intelligent agent and being corresponds to the agent

[14] Adequate protection means that reflecting the physical world guarantees that the intelligent being is consistent and coherent.

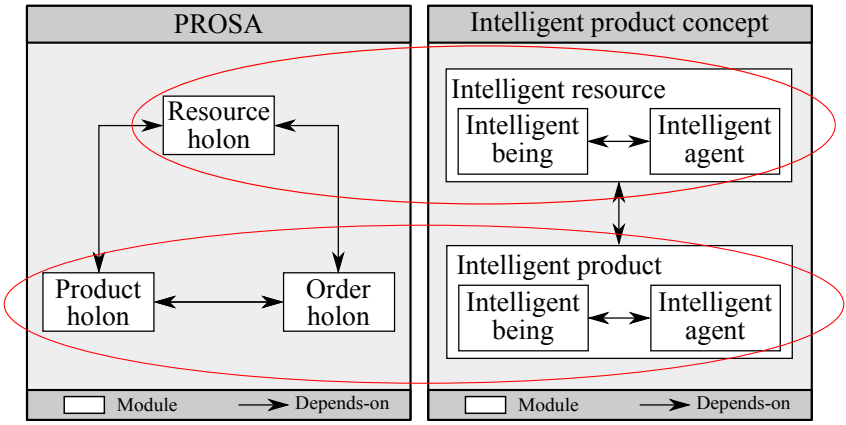


Figure 2.10: Correspondence between PROSA and the intelligent product concept.

and environment entity parts of the PROSA holons. More information on intelligent agents and beings can for instance be found in [194, 199, 200].

Intelligent cargo

In the context of transport logistics, the term '*intelligent cargo*' is also used (e.g. by the EURIDICE EU project [60]). Intelligent cargo is a concept that has the potential to offer high-quality logistic services. The FP7 project EURIDICE studies and develops this technology. The goal is to have a paradigm shift towards the intelligent cargo concept in the field of ICT applications for transport logistics. According to the project [145]: "Intelligent Cargo connects itself to logistic service providers, industrial users and authorities to exchange transport-related information and perform specific services whenever required along the transport chain". The vision of the project is the following: "In five years time, most of the goods flowing through European freight corridors will be 'intelligent', i.e.: self-aware, context-aware and connected through a global telecommunication network to support a wide range of information services for logistic operators, industrial users and public authorities"[15]. The EURIDICE project recognizes autonomous decisions as the most advanced level for intelligent cargo. Intelligent cargo will autonomously execute the necessary actions in function of the current circumstances and opportunities.

[15]While the proposed time period seems optimistic, goods are indeed becoming more and more 'intelligent' (e.g. by attached RFID tags). Currently, this intelligence is mainly used for improved tracking & tracing.

2.4 Generalization and applicability

During the development of PROSA and delegate MAS, the introduction of constraints on the applicability was minimized. Basically, the technology can be used for applications with the following characteristics [195, 205].

- Activities are executed on resources and are subject to technical constraints.
- Virtual execution of these activities in a digital mirror image of the world-of-interest is possible and can be much faster than in reality.
- The underlying world evolves several orders of magnitude slower than the control system.
- The underlying world and the control system do not compete for technical resources. For instance, the control of a telecommunication network fails to comply with this condition as both systems compete for the communication bandwidth of the network.
- The control system is able to observe and direct the entities in the underlying world.
- The socio-economic value of enhanced coordination more than compensates the cost and effort for the virtual execution.

Candidate application domains are, next to manufacturing and logistics, road construction, harvesting, power grids, traffic, etc. As already indicated, the terms product and order originate from the manufacturing domain and do not seem very suitable in other domains. A more general wording is task type and task (instance). Figure 2.11 shows a *rephrasing* of PROSA to account for a wider and more abstract or generic application domain. An order holon corresponds to a task that has to be executed, and does not have to represent a physical product instance. A product holon represents a task type. The interactions between the three components can be summarized as 'how', 'what' and 'when'. The task and task type components confer on *how* a task should be executed, the task and resource components determine *what* or which operations have to be executed and the task and resource components decide *when* these operations are executed.

The 'what' relationship between task type and resource (or between intelligent product and intelligent resource) can be seen as based on services. The intelligent products search for the services required to get their task fulfilled, while the intelligent resources offer these services. Services can be described by means of (complex) ontologies, but in many cases it suffices to describe services by their capabilities and constraints. For instance, the service offered by a truck can be described by the transport capability and by constraints like the maximum volume and weight of the load.

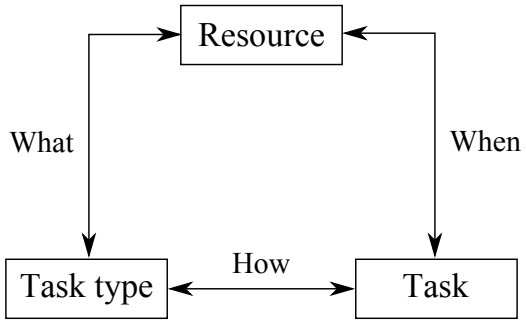


Figure 2.11: Rephrasing of PROSA for a generic application domain.

Because of this service-oriented approach, the concepts and principles of the holonic control system can be easily applied to other domains than manufacturing and logistics. The services offered by the resources form a decoupling point between the generic holonic system and application-specific elements. The behavior of the intelligent products is generally applicable (but with specific plug-ins for decision making), while the intelligent resources offer domain-specific services and contain a model of the domain-specific behavior of the corresponding resources. These models allow the intelligent products to see the effects of a service (e.g. the duration), without the intelligent products having to deal with domain-specific properties.

The proposed holonic control system has already been applied to various applications in domains as manufacturing control, open-air engineering, traffic and robotics. Examples of applications are the control of a car paint shop [150, 211] and a photographic foil facility [167, 217], the coordination of harvesting vehicles [4, 5], multi-robot coordination [151, 152] and the control of chain conveyor systems [206, 208]. A complete overview can be found in [205].

Supporting services

During operations, it might be necessary to introduce services to support core activities of the system. These *supporting services* have to be carried out by the resources during which these resources are not productive or available for normal tasks. Nevertheless, they are necessary to guarantee the correct functioning of the system. Examples are maintenance in a manufacturing context or the refueling of trucks in a logistic context.

These services can be easily integrated when they are offered by the intelligent resources (resource holons). As an example, consider machine maintenance in a

factory. In a more simplified setting, the machine itself offers a maintenance service, next to its other processing services. In a more advanced setting, this maintenance service is offered by another resource (a technician, consultant, operator, ...). In the latter case, two resources are needed at the same time to execute the service (the machine being maintained and e.g. the technician carrying out the maintenance), so the control system has to be able to deal with multi-resource allocation. In both settings however, the supporting services can be seamlessly integrated with the core activities. It suffices that for every supporting service that has to be executed, a corresponding intelligent product (order holon) is created. This intelligent product will search for the intelligent resources required to get its task fulfilled, just as the intelligent products representing the business activities. The intelligent resources do not make a distinction between both types of intelligent products, and will for all products try to allocate a time slot to execute the required service.

In manufacturing, several strategies can be used to create intelligent products in order to perform maintenance. If periodic maintenance is preferred for a machine, the control system can create intelligent products at regular time intervals. Another possibility is that the intelligent resource itself creates intelligent products to execute maintenance tasks, for instance when it notices a degraded performance (in process time or quality). Similarly, when a machine breaks down, the corresponding intelligent resource can create an intelligent product that is responsible for the repair of the machine. It is possible to give these maintenance and repair intelligent products a higher priority to ensure the maintenance or repair is done on time. Moreover, when maintenance is delayed or repair is needed, the intelligent resource will indicate the unavailability or degraded performance of its operations. This induces intelligent products, needing these operations, to wait until maintenance or repair has occurred. Other examples of supporting services in the context of manufacturing are the delivery of additional tools to machines, the delivery of empty containers or palettes to workstations, etc.

In logistic applications, an example of a supporting service is the refueling of trucks. In order to process a set of transportation tasks, a truck needs to refuel regularly. Similarly to maintenance, this refueling can be integrated with the transportation activities by creating an intelligent product that is responsible for performing the refueling task. This intelligent product will then try to allocate a time slot in which the refueling will happen. Again, it is possible to give the intelligent product a higher priority, although this is not necessary. When the refueling intelligent product is not able to schedule the refueling in time, the other intelligent products - representing freight - will notice that the truck will run out of fuel and is not able to reach its destination. Consequently, several intelligent products will look for another truck to reach their destination,

allowing the refueling to be scheduled in time.

2.5 Multimodels

Every holon consists of a decision making component and a component that reflects reality. The latter component - the environment entity or intelligent being - has to be able to answer what-if questions about (the dynamic behavior of) its corresponding real-world entity. Therefore, it contains a model that supports virtual execution. Currently, a multimodel formalism is used, which allows using different models with different formalisms and different levels of detail. This formalism is based on the finite state automaton (FSA) controlled multimodel described by Fishwick and Zeigler [63]. The basic ideas of the applied multimodels will be explained in this section, more information can be found in [165].

A model applies to a certain entity of the world-of-interest at a certain level of detail. To describe the status of an entity, the concept of the *state* of an entity is used. The state is the collection of variables necessary to describe the status of an entity at any given time [226]. For a truck for instance, the state can contain variables to represent its position and fuel content. The considered scope determines which variables are included in the state of the entity. The multimodel then acts on this state.

The *Petri net* formalism is used in this thesis for the top-level of the multimodels[16]. Petri nets are a graphical and mathematical tool for modeling and studying systems which are concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic [137]. As a graphical tool, Petri nets can serve as a visual-communication aid similar to block diagrams and flow charts. A Petri net consists of places and transitions (represented by bars and circles), interconnected by directed arcs, either from places to transitions or from transitions to places. Each arc has a weight (positive integer). If the weight is not indicated, a default weight of one is assumed. A transition has so-called input and output places. If an arc connects a place with the transition, that place is an input place of the transition. Similarly, if an arc connects the transition with a place, that place is an output place of the transition. Each place represents a condition of the modeled system while a transition is an event which causes a state change. The graph (formed by places, transitions and arcs) only defines the structure of the behavior of the system; the semantic meaning should be added to the places and transitions. Places may contain a discrete number of tokens. During execution, tokens travel through the Petri

[16] To be more specific, these models can be considered as colored, timed Petri nets [165].

net and their configuration represents the 'state' or 'marking' of the Petri net. A transition can only 'fire' if its input places contain the number of tokens indicated by the weight of the corresponding arcs. When this 'fire condition' is true, the transition is 'enabled'. An enabled transition may fire, i.e. the required tokens are removed from its input places and a number of tokens (indicated by the weight of the corresponding arcs) is created in its output places. This process is atomic, so even if multiple transitions are enabled simultaneously, only one transition can fire at a time.

In the applied *multimodel* formalism, a transition can represent a submodel and it can have an 'in' and an 'out' boundary condition. These boundary conditions define when the submodel is valid. They are conditions on some of the properties of the entity's state. An in boundary condition determines when the submodel is active and can be executed. The out boundary condition defines when the submodel is ready and becomes inactive again. A transition may fire when its fire condition and the in boundary condition are true. Then, the corresponding submodel is executed until the out boundary condition is true and the correct number of tokens is placed on all output places.

The submodels can again be Petri nets, or *continuous time models*. In these models, the state variables change continuously over time. Such a model can be any software module that complies to a specific interface [165]. These models are able to update the state of an entity until a certain boundary condition becomes true.

To illustrate the multimodel concept, the model of a diverter in a chain conveyor network will be described. A chain conveyor is a type of conveyor that consists of a continuously circulating transport chain. Carriers (e.g. roll containers) can be attached to the chain in order to be pushed forward. Multiple chains can be connected to each other by means of diverters which can switch a carrier from one chain to another. Figure 2.12 shows the top-level model of such a diverter. The model consists of four places and five transitions (four of them have a submodel). When this model is executed, transition *T1* can fire because the place *Start* contains a token and no in boundary condition is defined. The submodel *Waiting* is then executed until the *Enabled* boundary condition is true. This submodel represents the behavior of the diverter when it is idle. The out boundary condition is fulfilled when the diverter is 'enabled' and has to switch a carrier from its current chain to another one. Next, a token is placed in *Enabled* and transition *T2* can fire. The corresponding submodel - again *Waiting* - is executed until its out boundary condition is true. This condition is true when the diverter is 'disabled' or a carrier arrives at the diverter. If *Disabled* becomes true, transition *T3* fires and a token is placed on *Start*. If *CarrierArrived* is true, transition *T4* and *T5* are sequentially fired. First, *Disconnecting* is executed. This *Disconnecting* model represents the disconnecting of the carrier from its

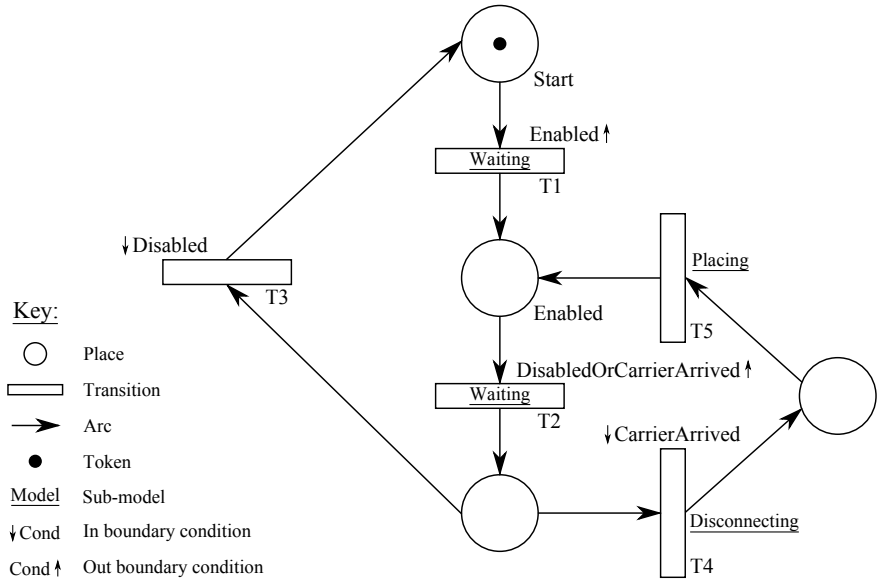


Figure 2.12: Top-level model of a diverter.

current chain. Subsequently, the carrier is placed on the other chain in the submodel *Placing*. Afterwards, a token is placed in *Enabled* and *Waiting* is again executed until the diverter is disabled or a next carrier arrives.

2.6 Conclusions

The aim of this chapter was to present existing research upon which the development of the Holonic Logistics Execution System is built. The chapter started with explaining what is understood with logistics and by describing several properties of the logistics domain. The role of an LES was also explained and related to the role of an MES. Next, holonic systems and multi-agent systems were introduced as a prerequisite to explain the software architecture of the HMES and HLES. This architecture combines the PROSA reference architecture, developed in accordance with the holonic manufacturing paradigm, and the delegate MAS architectural pattern. Note that this chapter made an abstraction of specific implementation aspects of the developed HLES implementation. These aspects are described in more detail in Chapter 5.

The chapter also shortly discusses the intelligent product concept, which adds

intelligence to manufactured products, and how this relates to the introduced holonic concepts. The applicability of these concepts is also discussed and to show that the concepts and principles can be easily applied to other domains than manufacturing and logistics, a more general description of PROSA is provided.

An important aspect for the easy development of new implementations is that the 'core' of the proposed coordination and control system can be reused. However, the necessary models and decision mechanisms have to be developed. The various decision mechanisms (e.g. the selection of an intention by the order agents or choosing the next operation by the exploring ants) are application-specific. But, because there is a clear separation between the decision making (agent) and reflection of reality (being or environment entity), the models can be seen as (domain-specific) building blocks and can be reused in every application that contains the corresponding real-world entities. So, merely the decision making is application-specific and can be considered as a plug-in to the system which can be easily replaced by another algorithm or rule. Moreover, the decisions can be based on the available short-term forecasts, which allows for better (informed) decision making.

As there is a model of every relevant entity in the world-of-interest, the proposed control system has a single-source-of-truth design. This makes it easier to maintain correct and up-to-date information. To model all relevant entities, the multimodel formalism is used, which is described in the last section of this chapter.

Chapter 3

Cross-docking

As the logistics domain is very broad, this thesis will focus on a specific logistic strategy: cross-docking. This chapter describes the cross-docking concept and gives an extensive overview of the available literature about this subject[1]*. First, Section 3.1 introduces the ideas and concepts of cross-docking. Then, Section 3.2 discusses in which situations cross-docking is a suitable strategy and deals with the requirements for a successful implementation. In Section 3.3, the characteristics are discussed that can be used to differentiate between alternative cross-docking systems. Next, Section 3.4 provides a review of the existing literature about cross-docking. The discussed papers are classified based on the problem type. These problems range from strategic or tactical to operational problems. The conclusions with opportunities to improve and to extend the current research are summarized in Section 3.5.*

3.1 Introduction

Cross-docking is a logistic strategy nowadays used by many companies in different industries (e.g. retail firms and less-than-truckload (LTL) logistics providers). The basic idea behind cross-docking is to transfer incoming shipments directly to outgoing vehicles without storing them in between. This practice can serve different goals: the consolidation of shipments, a shorter

[1]Most of the work described in this chapter is also published in J. Van Belle et al. "Cross-docking: State of the art". *Omega* 40(6). Special Issue on Forecasting in Management Science, pp. 827–846, 2012.

delivery lead time, the reduction of costs, etc. The role of cross-docking in industry even seems to increase [1, 2, 8, 26].

In a traditional distribution center, goods are first received and then stored, for instance in pallet racks. When a customer requests an item, workers pick it from the storage and ship it to the destination. From these four major functions of warehousing (receiving, storage, order picking and shipping), storage and order picking are usually the most costly. Storage is expensive because of the inventory holding costs, order picking because it is labor intensive. One approach to reduce costs could be to improve one or more of these functions or to improve how they interact. Cross-docking however is an approach that eliminates the two most expensive handling operations: storage and order picking [17, 67, 113, 172].

A definition of cross-docking provided by Kinnear [97] is: "receiving product from a supplier or manufacturer for several end destinations and consolidating this product with other suppliers' product for common final delivery destinations". In this definition, the focus is on the consolidation of shipments to achieve economies in transportation costs. The Material Handling Institute (MHI) defines cross-docking as "the process of moving merchandise from the receiving dock to shipping [dock] for shipping without placing it first into storage locations" [69]. The focus is now on transshipping, not holding stock. This requires a correct synchronization of incoming (inbound) and outgoing (outbound) vehicles. However, a perfect synchronization is difficult to achieve. Also, in practice, staging is required because many inbound shipments need to be sorted, consolidated and stored until the outbound shipment is complete. So, this strict constraint is relaxed by most authors. In this thesis, the following working definition is used: *cross-docking is the process of consolidating freight with the same destination (coming from one or more origins), with minimal handling and with little or no storage between unloading and loading of the goods.* If the goods are temporally stored, this should be only for a short period of a time. An exact limit is difficult to define, but many authors talk about 24 h (e.g. [17, 113, 191, 223]). If the goods are placed in a warehouse or on order picking shelves or if the staging takes several days or even weeks, it is not considered as cross-docking but as (traditional) warehousing. However, even if the products are staged for a longer time, some companies still consider it cross-docking, as long as the goods move from supplier to storage to customer virtually untouched except for truck loading [1, 227]. Many organizations use a mixture of warehousing and cross-docking to combine the benefits of both approaches [8].

A terminal dedicated for cross-docking is called a cross-dock. In practice, most cross-docks are long, narrow rectangles (I-shape), but other shapes are also used (L, T, X, ...) [17]. A cross-dock has multiple loading docks (or dock doors)

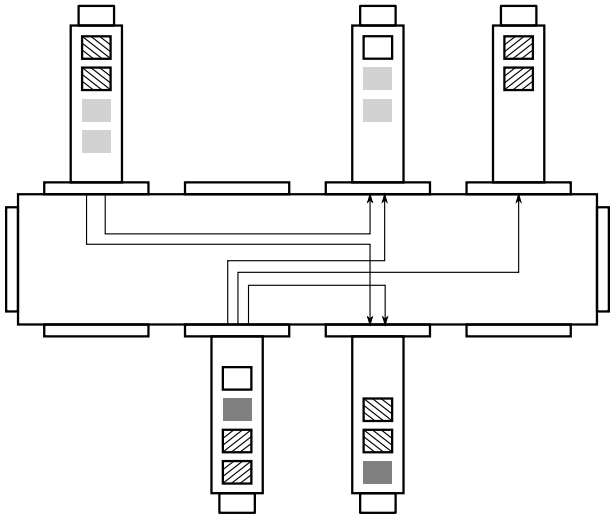


Figure 3.1: Material handling at a typical cross-dock.

where trucks can dock to be loaded or unloaded. Incoming trucks are assigned to a 'strip door' where the freight is unloaded. Then the goods are moved to its appropriate 'stack door' and loaded on an outbound truck. Mostly, there is no special infrastructure to stage freight. If goods have to be stored temporarily, they are placed on the floor of the cross-dock (e.g. in front of the dock door where the departing truck is or will be docked). However, it is possible that the cross-dock contains for instance a pallet storage, certainly if cross-docking is combined with warehousing.

Figure 3.1 presents a schematic representation of the material handling operations at an I-shaped cross-dock with 10 dock doors. Incoming trucks are either directly assigned to a strip door or have to wait in a queue until assignment. Once docked, the freight (e.g. pallets, packages or boxes) of the inbound truck is unloaded and the destination is identified (e.g. by scanning the barcodes attached to the goods). Then, the goods are transported to the designated stack door by some material handling device, such as a worker operating a forklift or a conveyor belt system. There, the goods are loaded onto an outbound truck that serves the dedicated destination. Once an inbound truck is completely unloaded or an outbound truck is completely loaded, the truck is replaced by another truck.

Cross-docking corresponds with the goals of lean supply chain management: smaller volumes of more visible inventories that are delivered faster and more

frequently [45]. In the literature, several other (possibly intertwined) advantages of cross-docking compared with employing traditional distribution centers and point-to-point deliveries are mentioned (e.g. [1, 26, 38, 67, 219]). Some advantages compared with traditional distribution centers are:

- cost reduction (warehousing costs, inventory-holding costs, handling costs, labor costs);
- shorter delivery lead time (from supplier to customer);
- improved customer service;
- reduction of storage space;
- faster inventory turnover;
- fewer overstocks;
- reduced risk for loss and damage.

Some advantages of cross-docking compared with point-to-point deliveries are:

- cost reduction (transportation costs, labor costs);
- consolidation of shipments;
- improved resource utilization (e.g. full truckloads);
- better match between shipment quantities and actual demand.

These advantages make cross-docking an interesting logistic strategy that can give companies considerable competitive advantages. Wal Mart is a well-known example [180], but also several other companies have reported the successful implementation of cross-docking (e.g. Eastman Kodak Co. [45], Goodyear GB Ltd. [97], Dots, LLC [140] and Toyota [227]).

Although cross-docking has already been applied in the 1980's (e.g. by Wal Mart), it has only attracted attention from academia much later and mostly during the recent years. For instance, more than 85 % of the academic papers found during the search process are published from 2004 on. During these years, a considerable amount of papers have been published and because of the growing interest from industry [1, 2, 8, 26], it is probable that still more research on this topic will be performed the coming years.

The term cross-docking usually refers to the situation in which trucks or trailers[2] are loaded and unloaded at a cross-docking terminal. However, the operations to handle freight at a harbor or airport are sometimes very similar. At a harbor for instance, containers are unloaded from a ship and temporarily placed onto the quay until they are loaded onto another ship or onto a truck. An airport can also be seen as a kind of cross-dock for transferring passengers and their baggage. In the literature, several papers can be found that deal with similar problems

[2]In this chapter, the terms truck, trailer and vehicle will be used interchangeably.

as encountered in cross-docking, but specific for harbors or airports (e.g. how to determine the layout of an airport terminal [11, 141], how to assign airplanes to gates [57], etc.). These papers are not taken into account for the literature review presented here. The chapter focuses on the typical cross-docking in which goods are transferred between trucks at a cross-dock. The specific application or industry (e.g. less-than-truckload (LTL) or courier, express and parcel (CEP) industry) is not important, as long as the applied material handling can be considered as cross-docking.

3.2 When and how to use cross-docking?

Although cross-docking is nowadays used by many companies, it is probably not the best strategy in every case and in all circumstances. This section briefly describes the existing literature that gives some guidelines for the successful use and implementation of cross-docking.

Apte and Viswanathan [8] discuss some factors that influence the suitability of cross-docking compared with traditional distribution[3]. A first important factor is the *product demand rate*. If there is an imbalance between the incoming load and the outgoing load, cross-docking will not work well. Hence, goods that are more suitable for cross-docking are the ones that have demand rates that are more or less stable (e.g. grocery and regularly consumed perishable food items). For these products, the warehousing and transportation requirements are much more predictable, and consequently the planning and implementation of cross-docking becomes easier. The *unit stock-out cost* is a second important factor. Because cross-docking minimizes the level of inventory at the warehouse, the probability of stock-out situations is higher. However, if the unit stock-out cost is low, the benefits of cross-docking can outweigh the increased stock-out cost, and so cross-docking can still be the preferred strategy. As shown in Figure 3.2, cross-docking is therefore preferred for products with a stable demand rate and low unit stock-out cost. The traditional warehousing is still preferable for the opposite situation with an unstable demand and high unit stock-out costs. For the two other cases, cross-docking can still be used when proper systems and planning tools are in place to keep the number of stock-outs to a reasonable level.

Some other factors that can influence the suitability of cross-docking are the distance to suppliers and customers (higher distances increase the benefits of consolidation), the product value and life cycle (a larger reduction in inventory

[3]It is assumed that the demand quantities are small, otherwise point-to-point deliveries are more suited.

Unit stock-out costs	*Product demand rate*: Stable and constant	Unstable or fluctuating
High	Cross-docking can be implemented with proper systems and planning tools	Traditional distribution preferred
Low	Cross-docking preferred	Cross-docking can be implemented with proper systems and planning tools

Figure 3.2: Suitability of cross-docking (adapted from [8]).

costs for products with a higher value and shorter life cycle), the demand quantity (a larger reduction in inventory space and costs for products with a higher demand), the timeliness of supplier shipments (to ensure a correct synchronization of inbound and outbound trucks), etc. [8, 115, 157].

Some authors use a more quantitative approach to study the suitability of cross-docking. For instance, Galbreth et al. [67] compare the transportation and handling costs between a situation in which a supplier has to ship goods to several customers with only direct shipments and a situation in which also indirect shipments via a cross-dock are possible. For the second situation, a mixed integer programming (MIP) model is proposed to determine which goods should go directly from supplier to customer and which goods should be shipped via a cross-dock to meet the (known) demands. The transportation costs are modeled in a realistic way: fixed for truckload shipping, while the less-than-truckload shipping costs are modeled using a modified all-unit discount (MAUD) cost function. The holding costs at the customers are proportional to the quantity and the holding time between arrival time and due date. The costs for the two situations are compared under varying operating conditions. The authors conclude that cross-docking is more valuable when demands are less variable and when unit holding costs at customer locations are higher. On the other hand, it is less valuable when the average demands are close to truck load capacity.

Other quantitative approaches make a comparison between a situation with a cross-dock and a situation with a traditional distribution center. For instance, Kreng and Chen [101] compare the operational costs. Besides the transportation and holding costs, the production costs (more specific the set-up costs) of the goods at the supplier are taken into account. When a cross-dock is used, more frequent deliveries to the cross-dock are required and the batch size needs to be smaller, which causes higher set-up costs. Waller et al. [221] look to both situations from an inventory reduction perspective.

Schaffer [172] discusses the successful implementation of cross-docking. When a company wants to introduce cross-docking, the introduction should be prepared very well. If the necessary equipment is already available and because cross-docking seems simple, one easily assumes that cross-docking can be implemented without much effort. However, cross-docking itself is quite complex and requires a high degree of coordination between the supply chain members (e.g. the timing of arrival and departure). So, the requirements for successful cross-docking should be understood thoroughly and the implementation should be planned carefully. In [172], Schaffer elaborates on six categories of requirements for a successful implementation.

According to Witt [227] and to Yu and Egbelu [234], software to plan and control the cross-docking operations (e.g. a warehouse management system or WMS) plays an important role in the successful implementation of cross-docking. The required (automated) hardware for a cross-docking system (material handling devices, sorting systems, etc.) might come off the shelf and is easily available today. But the software needs to be tailored to the specific requirements and is in general relatively less developed, although it is as important as hardware to cross-docking success. This is also confirmed by a survey among professionals who are involved in cross-docking and who denote IT system support as a key barrier to effective cross-docking [1, 2]. Hence, the system requirements need to be carefully defined and studied in order to prevent installing the physical system to discover afterwards there is no information and communication system in place for successful operation.

This software system can only work correctly if it is fed with accurate and timely information. Compared with regular distribution, the information flow to support cross-docking is significantly more important [157]. For instance, to coordinate the inbound and outbound trucks to the appropriate docks, the arriving time and the destination of the freight need to be known before the physical arrival of the goods (e.g. via advance shipping notice (ASN)). Several information technology tools are available to realize this information flow, e.g. electronic data interchange (EDI), shipping container marking (SCM), bar-coding and scanning of products using universal product code (UPC) [8]. Regardless of which technology is chosen, the supply chain partners must be able and willing to deliver the required information via this technology. A good cooperation across the supply chain can make or break the cross-docking implementation [157, 172, 227].

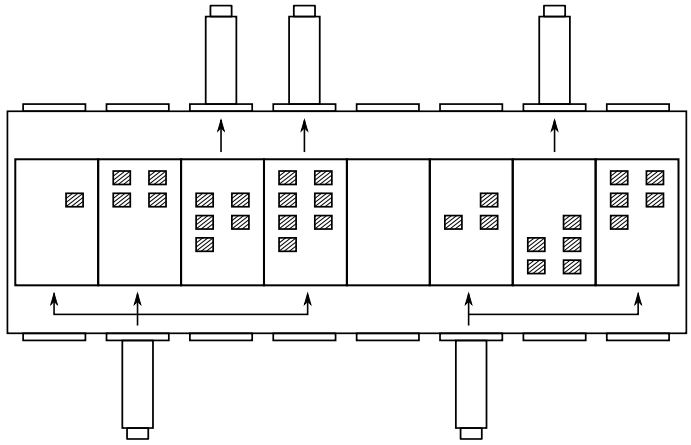


Figure 3.3: A single-stage cross-dock in which the products are staged in zones corresponding to the stack doors (adapted from [76]).

3.3 Cross-dock characteristics

Several characteristics can be considered to distinguish between various types of cross-docks (and cross-docking). A common distinction made in the literature is based on the number of touches [1] or stages [76]. In *one-touch* cross-docking, products are touched only once, as they are received and loaded directly in an outbound truck. This is also called *pure* cross-docking [8, 148]. In a *two-touch* or *single-stage* cross-dock, products are received and staged on the dock until they are loaded for outbound transportation. Usually, the goods are put into zones corresponding to their strip or stack door (see Figure 3.3). In the case of a *multiple-touch* or *two-stage* cross-dock, products are received and staged on the dock, then they are reconfigured for shipment and are loaded in outbound trucks. In a typical configuration, the incoming freight is first put in zones corresponding to the strip doors. The goods are then sorted to the zones corresponding to the stack doors (see Figure 3.4).

Another distinction can be made according to when the customer is assigned to the individual products [232]. In *pre-distribution* cross-docking, the customer is assigned before the shipment leaves the supplier who takes care of preparation (e.g. labeling and pricing) and sorting. This allows faster handling at the cross-dock. On the other hand, in *post-distribution* cross-docking, the allocation of goods to customers is done at the cross-dock.

Still some other distinctions are possible. The German supermarket retailer

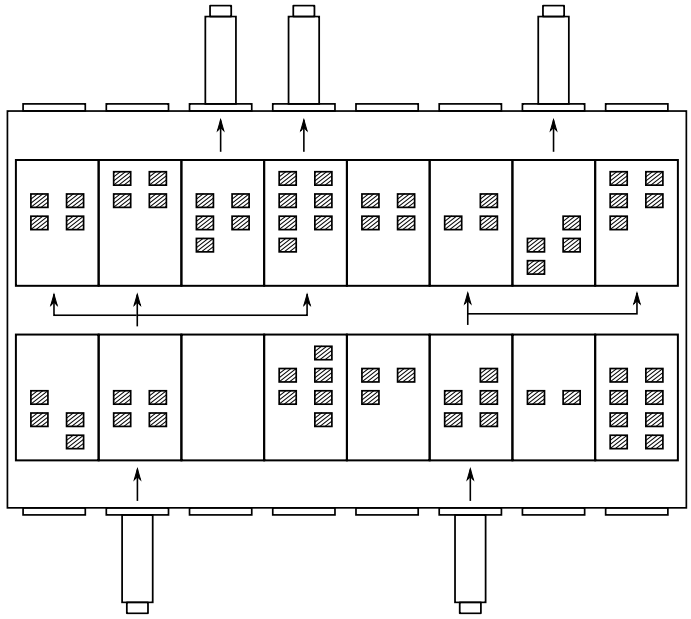


Figure 3.4: A two-stage cross-dock in which the products are staged in zones corresponding to the strip and stack doors and are sorted in between (adapted from [76]).

Metro-AG for instance distinguishes *source-oriented* and *target-oriented* cross-docking based on the location of the cross-docking terminals relative to suppliers and customers [102]. Napolitano [139] distinguishes several types of cross-docking based on the intended use and in [148], eight different cross-docking techniques are listed.

In this section, several characteristics are described that can be used to distinguish between different cross-dock types[4]. Note that real world characteristics of the cross-dock are considered, and not the properties from a specific decision problem related to cross-docking. For the papers included in the literature review (Section 3.4), the characteristics of the considered cross-docks will be listed in tables according to the characteristics described here[5]. However,

[4]Some of the characteristics described here are similar to the characteristics used by Boysen and Fliedner [26] to make a classification of truck scheduling problems. However, Boysen and Fliedner consider not only real world characteristics, but also characteristics of the (mathematical) models.

[5]At least for the papers in which these characteristics are described, i.e., in which real world details of the cross-dock are considered (Sections 3.4.5, 3.4.6, 3.4.7 and 3.4.8).

the structure of Section 3.4 is not based on these characteristics, but on the considered problem type.

The characteristics can be divided into three groups: physical characteristics, operational characteristics and characteristics about the flow of goods[6]. In the next sections, these groups will be described in more detail.

3.3.1 Physical characteristics

The physical characteristics are characteristics of the cross-dock that are supposed to be fixed (for a rather long time). The following physical characteristics are considered.

Shape Cross-docks can have a large variety of shapes. The shape can be described by the letter corresponding to the shape: *I, L, U, T, H, E, ...*

Number of dock doors A cross-dock is also characterized by the number of dock doors it has. In practice, cross-docks range in size from 6 to 8 doors to more than 200 doors, and even a cross-dock with more than 500 doors exists [75]. In the literature, sometimes the number of dock doors is limited to only 1 or 2. In these cases, the idea is not to model a realistic cross-dock, but to gain some insight by studying a simplified model.

Internal transportation The transportation inside the cross-dock can be executed *manually* (e.g. by workers using forklifts) or there can be an *automated* system in place (e.g. a network of conveyor belts). The available infrastructure will of course be dependent on the type of freight that is handled in the cross-dock. For instance, LTL carriers handle mostly palletized freight and so make use of forklifts. Conveyor systems on the other hand are among others used by parcel carriers, as they deal with many (small) packages. A *combination* of both transportation modes is also possible.

[6]This classification is rather vague. For some characteristics, it is not clear in which group they fit best or they can be assigned to multiple groups. For instance, temporary storage is considered as a flow characteristic. However, temporary storage can also be seen as a physical characteristic (storage is not possible because of space constraints) or operational characteristic (it can be an operational decision that storage is not allowed, e.g. to avoid congestion inside the cross-dock).

3.3.2 Operational characteristics

Some operational decisions can influence the functioning of the cross-dock. These operational constraints lead to the following characteristics.

Service mode According to Boysen and Fliedner [26], the service mode of a cross-dock determines the degrees of freedom in assigning inbound and outbound trucks to dock doors. In an *exclusive* mode of service, each dock door is either exclusively dedicated to inbound or outbound trucks. If this service mode is used, mostly one side of the cross-docking terminal is assigned to inbound trucks and the other side to outbound trucks. A second mode is *mixed* mode. In this mode, inbound and outbound trucks can be processed at all doors. These two modes can also be combined. In this *combination* mode, a subset of doors is operated in exclusive mode while the rest of the doors is operated in mixed mode.

Pre-emption If pre-emption is allowed, the loading or unloading of a truck can be interrupted. This truck is then removed from the dock and another truck takes its place. The unfinished truck has to be docked later on to finish the loading or unloading.

3.3.3 Flow characteristics

The characteristics of the flow of goods that have to be processed by a cross-dock can be very different. The following characteristics are distinguished.

Arrival pattern The arrival times of the goods are determined by the arrival times of the inbound trucks. The arrival pattern can be *concentrated* at one or more periods if the inbound trucks arrive together at (more or less) the same times. For instance, a cross-dock in the LTL industry serving a certain geographical area usually receives freight at two periods. Goods that have to be transported from inside that area to another area are picked up during the day and all pickup trucks arrive in the evening at the cross-dock. The goods are then sorted during the night and the outbound trucks leave in the morning. To simplify the problem, several papers assume that the inbound trucks arrive together (at the beginning of the time horizon). On the other hand, freight from outside the region but destined for that area arrives in the early morning and is then distributed during the day. Another possibility is that the arrival pattern is *scattered* and the inbound trucks arrive at different times during the

day. The arrival pattern has an influence on the congestion of the cross-dock and on the scheduling of workers and resources.

Departure time The departure times of the trucks can be restricted or not. In many cases there are *no* restrictions and the trucks leave the cross-dock after all freight is loaded or unloaded. However, it is also possible that the trucks have to depart before a certain point in time, for instance in order to be on time for a next transportation task. In this case, there can be restrictions imposed on the departure times of the *inbound* trucks only, so that these trucks have to be unloaded on time. In a similar way, it is possible that only the *outbound* trucks have to leave the cross-dock before a certain moment[7]. For instance, in the parcel delivery sector, the outbound trucks usually leave at a fixed point in time. Parcels arriving late have to wait until another truck departs for the same destination. It is also possible that *both* inbound and outbound trucks have restricted departure times.

Product interchangeability The freight handled at a cross-dock is in general not interchangeable. In this case, all products are dedicated to a specific *destination*[8] or a specific outbound *truck* (pre-distribution). Information about the destination or the dedicated truck is normally known before the products arrive at the cross-dock. It is however also possible that interchangeability of products is *allowed* (post-distribution). In this situation, only the type of products to be loaded on the outbound trucks and the corresponding quantity is known[8]. When the products are interchangeable, usually some value-added activities (e.g. labeling) need to be performed.

Temporary storage In pure cross-docking, the arriving freight is directly transported to outbound trucks, so no storage is needed. In practice however, this is rarely the case. In general, the goods are temporarily stored on the floor of the cross-docking terminal (e.g. in front of the stack doors) or even in a (small) warehouse. However, it is possible that goods are not allowed to be stored. For instance, if refrigerated products have to be cross-docked in a non-cooled terminal, these products have to be directly moved from a cooled inbound to a cooled outbound truck.

[7] This point in time can be dependent on the (due dates of the) actual load of the truck.

[8] The assignment of the products to a specific outbound truck is then an operational decision.

3.4 Literature review

Cross-docking practitioners have to deal with many decisions during the design and operational phase of cross-docks. These decisions can have a serious impact on the efficiency, so they have to be carefully taken. In the literature, several decision problems are studied. Some of these problems are more concerned about decisions with effects on a longer term (strategic or tactical), while others deal with short-term decisions (operational). This section gives a review of the existing literature about cross-docking problems. The literature review is structured according to the basic planning process a manager, wanting to start with cross-docking, is confronted with.

The first decisions that have to be taken during the planning process are strategic decisions: where will a cross-dock (or cross-docks) be located and what is the best layout of a cross-dock. Once the cross-dock is available, it will be part of a supply network (with one or more cross-docks). A tactical decision that has to be made then is how the goods will flow through the network to minimize the costs, while making supply meet demand. Next, the manager is faced with the operational decision (although it has also tactical aspects) of vehicle routing: before arriving at the cross-dock, freight has to be picked up at various locations, and the goods have to be delivered to multiple locations after consolidation at the cross-docking terminal. Other operational decisions deal with the assignment of trucks to dock doors or the scheduling of the trucks, and with the location where goods will be temporarily stored. Of course, the manager will also be confronted with problems that are not specific for cross-docking: the scheduling of the internal resources for the loading and unloading of the freight (e.g. the workforce), choosing the best staging strategy and determining an optimal truck packaging sequence.

The next sections describe the cross-docking problems dealt with in the literature. Only the problems that are specific for cross-docking are considered. First, the strategic decisions are discussed: the location of cross-docks and layout design. The tactical problem of cross-docking networks is described next. Further, the operational decisions are handled: vehicle routing, dock door assignment, truck scheduling and temporary storage. Finally, some papers that study other issues related to cross-docking are discussed.

3.4.1 Location of cross-docks

The determination of the location of one or more cross-docks is part of the design of a distribution network or supply chain. An important strategic decision that has to be made concerns the position of these cross-docks. This problem

cannot be handled isolated from the decisions that determine how the goods flow through this network. The determination of the flow of goods is discussed in Section 3.4.3, but problems that also involve a decision about the location are considered here. The problem where to locate facilities (e.g. distribution centers or plants) has attracted a considerable amount of attention[9]. The papers discussed in this section determine additionally the optimal flow of goods through the network. Moreover, they regard the facilities to be cross-docks because they explicitly take individual vehicles into account or because temporary storage is not allowed.

A first study about the location of cross-docks is performed by Sung and Song [184]. In the considered problem, goods have to be transported from supply to demand nodes via a cross-dock (direct shipments are not allowed). The cross-dock can be chosen from a set of possible cross-dock locations, each with an associated fixed cost. The demands are assumed to be known and there are two types of vehicles with a different capacity and cost. The aim is to find which cross-docks should be used and how many vehicles are needed on each link in order to minimize the total cost. This total cost consists of the fixed costs of the used cross-docks and the transportation costs. The authors present an integer programming model of the problem. This model is very similar to the model presented by Donaldson et al. [53] and Musa et al. [138] (discussed in Section 3.4.3) and similar simplifying assumptions are applied. Compared with these two papers however, the approach of Sung and Song does not consider direct shipments but does include the location decision. Because the problem is NP-hard, they propose a tabu search-based algorithm to solve the problem. The solutions determine how the goods flow through the network. Based on this flow, the number of vehicles can be derived by solving a subproblem. Some computational experiments are performed on generated test instances and indicate that the proposed algorithm finds good feasible solutions within a reasonable time.

Sung and Yang [185] extend this work and propose a small improvement to the tabu search algorithm. The authors also present a set-partitioning-based formulation of the problem and propose a branch-and-price algorithm based on this formulation to obtain exact solutions. The computational results show that this algorithm gives better results in terms of the number of (small-scale) problem instances solved and the required computation time compared with the results obtained by solving the integer programming model with the optimization software package CPLEX©.

Gümüş and Bookbinder [77] study a similar problem, but now direct shipments are allowed and multiple product types are considered (multicommodity). The

[9]Several references can be found in the papers discussed in this section.

facility cost for each cross-dock consists of a fixed cost and a throughput cost charged per unit load. The transportation cost also has two components: a fixed cost for each truck and a variable cost per unit load per unit distance. A last cost that is taken into account is the cost for in-transit inventory. In this approach, the synchronization of inbound and outbound trucks is not taken into account. The authors provide a mixed integer programming model of the problem. By solving several smaller problem instances optimally (with the optimization software packages LINGO© and CPLEX©), the influence of several cost parameters is studied. The authors conclude that the optimal number of cross-docks is an increasing function of the ratio between the (fixed) truck cost and the (fixed) facility cost.

A different approach is taken by Jayaraman and Ross [93]. They study a multi-echelon problem in which goods (from multiple product families) have to be transported from a central manufacturing plant to one or more distribution centers. From there, the goods are moved via cross-docks to the customers. The problem is tackled in two stages. In the first stage, a strategic model is used to select the best set of locations for the distribution centers and cross-docks. The authors provide an integer programming formulation that aims to minimize the fixed costs associated with operating open distribution centers and cross-docks and the various transportation costs. Demand splitting is not allowed: customers have to be assigned to single cross-docks while cross-docks have to be assigned to single distribution centers only. In the second stage, an operational model decides upon the quantities of each product type that need to be transported via distribution centers and cross-docks. The model tries to minimize the transportation costs while satisfying customer demand. This model is less restrictive than the first model (it relaxes for instance the demand splitting assumption) and can be executed once the open distribution centers and cross-docks are determined with the help of the first model. Both models are more simplified compared with the previous approaches. For instance, individual vehicles are not considered and the transportation cost is proportional to the quantity to ship. The authors propose a simulated annealing approach to solve larger problem instances. The computational experiments on generated problem instances indicate that the heuristic gives results with a deviation of about 4 % of the optimal solution (obtained with LINGO©), but 300–400 times faster.

In [162], the same authors present two other heuristics to tackle the problem. Both heuristics are based on simulated annealing but use an extra mechanism to avoid locally optimal solutions. The first heuristic makes use of a tabu list, the second heuristic allows a sudden re-scaling of the 'temperature'. For both heuristics, the solution quality and computational performance are tested for different 'cooling schedules'. The experimental results indicate that the simulated annealing heuristic combined with tabu search gives better solutions

in slightly more time.

Bachlaus et al. [14] also consider a multi-echelon supply chain network, including suppliers, plants, distribution centers, cross-docks and customers. The goal is to optimize the material flow throughout the supply chain and to identify the optimal number and location of suppliers, plants, distribution centers and cross-docks. The problem is formulated as a multi-objective optimization model that tries to minimize the total cost and to maximize the plant and volume flexibility. Because of the computational complexity of the problem, the authors propose a variant of particle swarm optimization (PSO) to design the supply chain. Some computational experiments are conducted and the results show that the proposed solution approach gives better results than a genetic algorithm and two other PSO variants.

3.4.2 Layout design

Once the location of a cross-dock is determined, another strategic decision that has to be made is to choose the layout of the cross-dock. The layout is interpreted as the dimension and shape of the cross-dock, as well as the dimension and shape of the internal cross-dock areas and their arrangement.

Bartholdi and Gue [17] focus on the shape of a cross-dock. Most existing cross-docks are long, narrow rectangles (I-shape), but there are also cross-docks shaped like an L, U, T, H or E. The cross-dock shape is sometimes determined by simple constraints (e.g. size and shape of the lot on which it will stand), but in this chapter the focus is on how the shape affects cross-dock performance. Several experiments are performed in which the labor costs (estimated by the total travel distance[10]) are measured for different shapes. The experiments suggest that an I-shape is the most efficient for smaller cross-docks (fewer than about 150 doors). For docks of intermediate size, a T-shape is best and for more than 200 doors (approximately) an X-shape is best. Cross-docks with a T or X-shape have a greater 'centrality'. However, they achieve this at the cost of additional corners which reduce the labor efficiency (two inside and two outside corners for T, four inside and four outside corners for X). An inside corner renders some doors unusable, while doors around an outside corner have less floor space available to stage freight. So, these additional corners are a fixed cost, which begins to pay off for larger docks. It is however not always easy to predict which shape is better, because this also depends on e.g. the freight flow pattern.

[10]Here and in the remainder of this chapter, the travel distance is the distance traveled (by workers, forklifts, ...) in order to transfer the goods internally from the inbound to the outbound truck.

Other papers deal with the design of the storage area where the freight can be temporarily staged (on the floor or in racks). In many cases, the freight is placed in several parallel rows and the workers can move between these rows. Vis and Roodbergen [219] deal with the operational decision where to temporarily store incoming freight (see Section 3.4.7). The proposed algorithm can also be used during the design phase to determine the optimal number of parallel storage rows and their lengths.

The (single-stage or two-stage) storage area can also be organized in parallel lanes directly next to each other which can only be accessed at both ends. Gue and Kang [76] make use of simulation to study the behavior of these so-called staging queues. The results suggest that, for a single-stage storage area, it is better to have more short lanes than fewer long ones, at least when the workers follow a rational approach. The results also indicate that two-stage cross-docking has a significantly lower throughput than single-stage cross-docking.

3.4.3 Cross-docking networks

Some authors do not study problems concerning a single cross-dock, but consider a network that contains one or more cross-docks. The aim is to determine the flow of goods through such a network in order to reduce costs, while making supply meet demand.

The research of Lim et al. [119] extends the traditional transshipment problem. The transshipment problem consists of a number of supply, transshipment and demand nodes. The arcs between these nodes have different capacity limits and costs. The objective is to find a minimum cost flow that meets all demands and the capacity constraints. In the extended transshipment problem, storage is allowed at the transshipment centers. These centers can be considered as cross-docks because the aim of the model is to minimize or eliminate holdover inventory. Moreover, this problem takes supplier and customer time windows into account and considers the capacity and holding costs of the cross-docks. All shipments have to pass via a cross-dock, so no direct shipments are considered. Similar to the original problem, the objective is to minimize the total cost (transportation costs and holding costs) while meeting demand and respecting the time windows and capacity constraints. If multiple departures and deliveries within a time window are allowed (multiple shipping–multiple delivery), the authors show that a time-expanded network can be used to formulate the problem as a minimum cost flow problem (MCFP) which can be solved in polynomial time. For other cases, the authors prove that the problem is NP-hard.

For the special case when only one delivery or departure is allowed within a time window and the departure and arrival times are fixed (single shipping–single delivery with fixed schedules), a genetic algorithm is developed by Miao et al. [134]. This heuristic gives better results (in terms of solution quality and computation time) than solving the integer programming formulation of the problem with CPLEX© (with a time limit).

Chen et al. [39] study a similar problem which they call the multiple cross-dock problem. The major differences are that supplies and demands are not-splittable and that different products can be considered (multicommodity flow problem). Also, transportation time is in this approach not taken into account. An integer programming formulation of the problem is provided, together with a proof of its NP-completeness. The authors propose three heuristics (simulated annealing, tabu search and a combination of both) to solve the problem. These heuristics provide better solutions than those obtained by solving the integer programming formulation with CPLEX©, within only less than 10 % the time used by CPLEX©. Among the three heuristics, tabu search seems to give the best results.

The previous studies represent the shipment of goods as flows. Individual transportation units are not considered and the transportation cost is proportional to the quantity to ship. However, to take advantage of consolidation, the vehicle transportation cost should be taken into account. A first approach that does consider the transportation vehicles explicitly (and this is why the authors regard it as cross-docking) is taken by Donaldson et al. [53]. In the considered problem, the goal is to determine whether to route freight directly from suppliers to customers or via a cross-dock and how many vehicles should be scheduled on each transportation link in order to minimize the transportation costs. Compared with the previous approaches however, this problem is more simplified, e.g. storage at the cross-docks is not considered and the synchronization of inbound and outbound trucks is left out of the problem. The authors eliminate links with a large transportation time in an attempt to consider time windows. However, when the due dates at the destination nodes can vary for the different goods, it is possible that the vehicle allocation of an obtained solution violates the due dates in practice. The authors present an integer programming model of the problem. Because the problem is difficult to solve with branch-and-bound (B&B) algorithms, an alternative approach is proposed. In this approach, an iterative procedure is used in which either the integrality restrictions on the links from origin nodes to the cross-docks or on the links from the cross-docks to the destination nodes are relaxed. This relaxation heuristic provides near optimal solutions in an acceptable time. The authors used this approach to compare several scenarios (with a different number of cross-docks at different places) for the network design of a postal service

company.

The same problem is also studied by Musa et al. [138]. They propose an ant colony optimization (ACO) heuristic to solve the problem and show that this heuristic gives in a short time slightly better results than a B&B approach (with the optimization software package LINDO©) that requires a much longer time.

The approach of Ma et al. [120] takes most of the aforecited concerns into account. The so-called *shipment consolidation problem* considers supplier and customer time windows and also the transportation times between the network nodes. Moreover, storage at the transshipment centers (cross-docks) is taken into account, shipments can be transported directly to their destination or via a cross-dock and the transportation cost accounts for the number of trucks. However, only one type of products is considered (single commodity). Again, the objective is to minimize the total cost (transportation and inventory cost) while satisfying the constraints imposed by the time windows. The authors present an integer programming model of the problem and show that it is NP-complete in the strong sense. Therefore, the authors propose a (two-stage) heuristic algorithm to solve the problem. The basic idea of the algorithm is to consider first trucks that can be fully loaded and then to find solutions that combine several smaller loads that are not considered yet. In the first stage, a full truckload plan (TL plan) and an initial less-than-truckload plan (LTL plan) are constructed. In the second stage, this initial LTL plan is improved iteratively by using a metaheuristic (squeaky wheel optimization or genetic algorithm). The computational experiments indicate that the proposed heuristic gives competitive results compared to CPLEX© (with a time limit) within a much shorter time.

3.4.4 Vehicle routing

Freight destined for a cross-dock needs in many cases to be picked up at various locations, and has to be delivered to multiple locations after consolidation at the cross-dock. Both the pickup and the delivery process can be seen as a vehicle routing problem and some studies consider cross-docking and vehicle routing simultaneously.

A first approach is taken by Lee et al. [108]. The aim is to find an optimal routing schedule for pickup and delivery (within the planning horizon) that minimizes the sum of transportation cost and fixed costs of the vehicles. It is assumed that split deliveries are not allowed and all pickup vehicles should arrive at the cross-dock simultaneously to prevent waiting times for the outbound trucks. While this can be a valid constraint for some cases (see Section 3.3.3), this is not generally true. The authors present an integer programming model

of the problem, which however seems unsatisfactory to solve the described problem. A tabu search algorithm is proposed to find solutions. This approach corresponds to the solving of two vehicle routing problems (one for pickup and one for delivery). The second routing problem can only start when the first one is finished and the complete process has to be finished within a certain planning horizon. Liao et al. [116] propose another tabu search algorithm to solve the same problem.

Wen et al. [223] study the so-called vehicle routing problem with cross-docking (VRPCD). In this problem, orders from suppliers have to be picked up by a homogeneous fleet of vehicles. These orders are then consolidated at a cross-dock and immediately delivered to customers by the same set of vehicles, without intermediate storage at the cross-dock. During the consolidation, goods are unloaded from the inbound vehicles and reloaded on outbound vehicles. The unloading must be completed before reloading starts. The authors assume that the duration of the unloading consists of a fixed time for preparation and a duration proportional to the load size. It is also assumed that if the delivery will be executed by the same vehicle as used for pickup, the unloading is not necessary (independent of the sequence in which the vehicle is loaded during the pickup tour). A time window is defined for all suppliers and customers and orders are not-splittable. In the case without consolidation, the solution of this problem can be found by solving two vehicle routing problems (one for pickup and one for delivery). Because of the consolidation however, the pickup and delivery routes are not independent. Only trying to minimize the distance of the pickup and delivery routes is not sufficient, the exchanges of orders at the cross-dock also have to be taken into account. These two aspects usually conflict with each other. The authors present a mixed integer programming formulation of the problem in which the objective is to minimize the total travel time of all vehicles. This formulation contains many variables and constraints, so the authors propose to use tabu search embedded within an adaptive memory procedure. This method is tested on realistic data involving up to 200 supplier-customer pairs. Experimental results show that the algorithm can produce solutions less than 1 % away from the optimum within short computing times (less than 5 s) for small problem instances. For larger instances, the gap with a lower bound is less than 5 % while the computation time stays below 5 min.

3.4.5 Dock door assignment

When an inbound or outbound truck arrives at the cross-dock, it has to be decided to which dock door the truck should be assigned. A good assignment can increase the productivity of the cross-dock and can decrease the (handling) costs. So, the *dock door assignment problem* tries to find the 'optimal' assignment of

inbound and outbound trucks to dock doors. It is assumed that there are at least as much dock doors as trucks, so each truck will be assigned to a different door and time aspects are not taken into account. If this condition is not fulfilled, the dock doors can be seen as (scarce) resources that have to be scheduled over time. This is the so-called *truck scheduling problem*. Both problems can be quite complex due to the number of doors and the dynamic nature of the problem. This section deals with the dock door assignment problem, while truck scheduling problems are discussed in Section 3.4.6.

The assignment of dock doors can be executed on a mid-term or short-term horizon [26]. Several papers solve the assignment problem on a *mid-term* horizon. Then, each dock door serves a specific inbound or outbound destination for a longer period of time (e.g. 6 months)[11]. All trucks coming from the same origin or having the same destination are assigned to the same dock. Such a fixed assignment is easier for workers because they know exactly to which dock door they need to ship each load, but it comes at the expense of a reduced flexibility. Even if a fixed assignment is used, it is important that the dock doors are reassigned when there is a significant change in the shipping pattern. When data about the inbound trucks are known far enough in advance, the assignment of the trucks can be solved on a *short-term* horizon. The trucks itself are assigned to the dock doors based on the actual freight flow. This 'floating dock' concept is put forward by Peck [149] who studied the material handling operations in an LTL terminal. Such an assignment implies that the workers are every day confronted with a different door for the same destination and have to take care that the freight is loaded into the correct truck. The use of modern information technology (e.g. bar code or RFID scanning together with a WMS) can be useful for this end. A combination of both is also possible. Several papers consider a cross-dock in which destinations are assigned to stack doors (so the outbound trucks are assigned on a mid-term horizon), while the assignment of the inbound trucks is done on a short-term horizon.

The characteristics of the cross-docks considered in the following papers are summarized in Table 3.1. As time aspects are neglected and there are enough available dock doors, the pre-emption, arrival pattern and departure time characteristic are not relevant here and are not shown.

In his dissertation, Peck [149] develops a detailed simulation model of an LTL terminal and tries to assign the trucks to dock doors in order to minimize the travel time[12] of the shipments. It is assumed that the travel time to transport the products between two trucks can be expressed as a function of the distance, based on the actual contents of the trucks and the required means of transport

[11] This implies that the cross-dock operates in exclusive service mode.

[12] Here and in the remainder of this chapter, the travel time is the time required to transfer the goods internally from the inbound to the outbound truck.

Table 3.1: Characteristics of the papers discussed in Section 3.4.5. A '$*$' indicates that not a single value of the characteristic is valid, but that all values can be used, 'ns' indicates that a characteristic is not specified.

Paper(s)	Shape	No. of doors	Internal transport	Service mode	Interchange-ability	Temporary storage
Peck [149]	I	$*$	manually	exclusive	truck	yes
Tsui and Chang [188]	I	$*$	manually	exclusive	destination	no
Bermúdez and Cole [19]	$*$	$*$	manually	exclusive	destination	ns
Cohen and Keren [44]	I	$*$	manually	exclusive	destination	ns
Oh et al. [143]	I	$*$	manually	exclusive	destination	ns
Bartholdi and Gue [16]	I	$*$	$*$	exclusive	destination	yes
Gue [75]	I	$*$	manually	exclusive	destination	ns
Brown [32] (semi-permanent)	$*$	$*$	manually	exclusive	destination	yes
(dynamic)	$*$	$*$	manually	mixed	truck	yes
Bozer and Carlo [28] (semi-permanent)	$*$	$*$	manually	exclusive	destination	ns
(dynamic)	$*$	$*$	manually	mixed	truck	ns
Yu et al. [233]	$*$	$*$	manually	exclusive	destination	ns

(2-wheeler, 4-wheeler or forklift). The designation of doors as either strip or stack doors is fixed beforehand. The problem is formulated as an integer programming model and because of the computational complexity, a heuristic (greedy balance algorithm) is provided to solve it. Simulation shows that his heuristic improves an assignment based on experience and intuition.

Another early study about the assignment of trucks to dock doors is performed by Tsui and Chang [188]. In this paper, a cross-dock is considered in which no storage is provided; all shipments go directly from inbound to outbound trucks. The problem is solved on a mid-term horizon, so the origins and destinations have to be assigned to dock doors, not the trucks itself. The designation of doors as strip or stack doors is fixed. The assignment problem is formulated as a bilinear programming problem that tries to minimize the travel distance of the forklifts (the number of forklift trips required to carry a certain load is assumed to be known). To solve it, the authors propose a simple heuristic method to find a local optimum. The authors do not provide test results, but conclude that the found solution can serve as a good starting point for the cross-dock manager.

There exist exact algorithms to solve bilinear optimization problems, but these are not very suited for this problem as the same authors mention in [189]. In this paper, a branch-and-bound algorithm is proposed to solve the dock door assignment problem exactly. The numerical tests show that this algorithm is however computational expensive.

Bermúdez and Cole [19] deal with a very similar problem, but now there is no fixed designation for the doors. All doors can have assigned either an origin or a destination. The mathematical model of Tsui and Chang [188] is adapted to take this into account. The objective function minimizes the total weighted travel distance instead of the real travel distance. A genetic algorithm (GA) is proposed to solve this problem. Based on data from an LTL carrier, the authors study the impact of different GA parameters on the solution and compare the results of the genetic algorithm with the results obtained with a pairwise exchange technique (2-opt). The genetic algorithm seems to give comparable or slightly better results.

Cohen and Keren [44] also extend the approach of Tsui and Chang [188]. The mathematical model is adapted to allow that freight for a certain destination can be split and delivered to multiple doors assigned to that destination (the capacity of the outbound trucks is taken into account). The proposed formulation is a non-linear MIP model that is impractical for real size problems. So, the authors propose a heuristic algorithm to solve it. Because of its simplicity, the heuristic can be easily recalculated to adapt to small changes in the freight flow pattern. It is however not clear how well this heuristic performs.

A different assignment problem is considered by Oh et al. [143]. This paper deals with cross-docking in a mail distribution center in which the different doors (and corresponding destinations) are clustered into groups. Each group has a shipping area located at the center of its stack doors. Arriving products are transported from the inbound trucks to these shipping areas, sorted according to their destination and loaded into outbound trucks. When a large amount of freight has to be shipped to a destination, this destination can be assigned to several stack doors. The objective is to find an assignment of destinations to stack doors and a clustering of destinations in groups that minimizes the total travel distance. So, the assignment of strip doors is not considered, and the assignment of stack door is solved on a mid-term horizon. The authors present a non-linear programming model of the problem and propose two heuristic methods to solve it: a decomposition heuristic and a genetic algorithm. Based on data obtained from the mail distribution center, the computational results indicate that both heuristics can reduce the travel distance compared with the current situation (about 13 % for the decomposition heuristic and about 9 % for the genetic algorithm).

Bartholdi and Gue [16] define the *layout* of a cross-dock as the specification of doors as either strip or stack doors and the assignment of destinations to stack doors. It is assumed that the assignment of inbound trucks to strip doors happens in real time by a dock supervisor using a first-come, first-served (FCFS) policy. The flow through each strip door then tends, over time, to resemble the aggregate flow through the terminal, so each inbound trailer is modeled as an 'average trailer'. The objective of this paper is to determine an 'optimal' layout. So, this paper deals with the mid-term assignment of outbound trucks, while the short-term assignment of inbound trucks is not considered and there is no fixed designation for the doors. In the previous approaches, the objective is the minimization of travel distance. According to the authors however, approaches to determine an optimal layout based on travel distance are inaccurate. The travel time should be taken into account, and the travel distance is not a good measure of travel time. The actual travel time also depends on the type of freight, the used material handling system and congestion. Minimizing the travel distance can even worsen congestion. In this paper, a (non-linear) mathematical model is described which can take different types of material handling systems into account and which uses models of different types of congestion. The model tries to minimize the total labor cost, which accounts for both travel costs (based on travel time) and congestion costs (based on waiting times due to congestion). The authors use a simulated annealing procedure that swaps pairs sequentially to solve the assignment problem. Based on results obtained with the developed model, the authors formulate some guidelines for efficient layouts. For instance, it is interesting to alternate high-flow stack doors with strip doors at the center of the cross-dock to reduce travel time and congestion. The

proposed method was used to improve the layout of an existing cross-dock and the authors report that labor productivity increased 11.7 % according to the company's measurements.

In the paper, it is assumed that the freight flows from strip doors to destinations are known and independent of the layout. This is modeled by placing an 'average trailer' at each strip door. However, when the cross-dock supervisor assigns incoming trailers to doors in real time based on the contents of the trailers and the location of the doors ('look-ahead scheduling' instead of FCFS), the material flows are altered and dependent on the layout. In [75], Gue examines the effect of look-ahead scheduling on the material flows and the layout of the cross-dock. To determine the layout with the lowest labor cost, the author proposes to search the solution space of all layouts with a local search algorithm (that swaps pairs of trucks). For a given layout, the labor costs can be determined if the resulting material flows are known (only travel costs are considered, no congestion costs). To model these flows, 'biased trailers' are constructed by solving a linear programming problem. Such a trailer contains freight that is biased toward the destinations that are closest to the strip doors to which it is assigned. The author proposes a specific look-ahead algorithm[13] to test the solutions using simulation. The simulation results indicate that it is possible to save 15 to 20 % in labor costs by using this look-ahead scheduling policy for the inbound trucks. Extra costs can be saved by constructing the layout of the terminal based on the altered flows (at least if the average number of destinations per inbound truck is low).

So, Gue determines the layout (that changes only periodically) and assumes that the inbound trucks will be scheduled using a real time policy. Another possibility is to determine the layout together with the short-term assignment of inbound trucks, i.e. the assignment of the inbound trucks itself and not the origins to dock doors. This is what Brown calls a *semi-permanent layout*. In her master thesis, Brown [32] studies the problem of assigning trucks to dock doors (trailer-to-door assignment problem or hub layout problem) and how to unload the inbound trucks (freight sequencing problem). For the *trailer-to-door assignment problem*, the objective is to minimize the total travel distance. A semi-permanent layout is constructed in two phases. The first phase allocates dock doors as strip or stack door and also assigns destinations to the stack doors. Similar to Bartholdi and Gue [16], 'average trailers' are used as inbound trailers. Starting from an initial assignment, a local search is performed with pairwise exchanges of trucks to generate a final solution. In the second phase, the inbound trucks are assigned to strip doors. Pairwise exchanges of inbound trucks are used to improve an initial assignment. Brown also considers a *dynamic layout* in which both the inbound and outbound trucks are assigned on a short-term

[13] Wang and Regan [222] propose two alternative scheduling policies (see Section 3.4.6).

horizon. Again, an initial assignment is improved by pairwise exchanges of trucks. The experimental results (based on actual shipment data) indicate that the dynamic layout reduces the total travel distance significantly compared with the semi-permanent layout.

Bozer and Carlo [28] also consider a semi-permanent and a dynamic layout. To determine the assignment of outbound trucks for the semi-permanent layout, the solution space of possible assignments is searched as done by Gue [75] and Brown [32], but simulated annealing is used instead of local search. Also different is that no 'average' or 'biased' inbound trucks are assumed, but actual data of several assignment periods are used. For a given outbound door assignment, the optimal inbound trailer-to-door assignment and the corresponding travel distance are then determined by solving a linear assignment problem (for which efficient algorithms exist). This is done multiple times, for each assignment period, and the sum of the resulting distances is taken. In this way, the variability in freight flow is taken into account. Once the optimal assignment of outbound trucks is determined, the short-term assignment of inbound trucks can be found by again solving the linear assignment problem. For the dynamic layout, the authors model the problem as a quadratic assignment problem (QAP) with rectilinear distances and present a MIP formulation. However, to solve large problem instances, the authors propose again to use simulated annealing, but now with the actual content of the inbound trucks. The proposed model tries to minimize travel distance, and congestion is not taken into account. The authors suggest that congestion can be avoided by not allowing solutions that have three outbound trucks assigned adjacently. The results of numerical experiments indicate that simulated annealing gives better results than a pairwise exchange steepest descent heuristic, which is known to perform well for solving a rectilinear QAP. Also, the dynamic layout seems to give slightly better results than the semi-permanent layout.

Yu et al. [233] also consider a semi-permanent layout. The objective is to minimize the total travel time. To determine the short-term assignment of inbound trucks, an online policy (different from FCFS) is proposed that assigns arriving inbound trucks on a real-time basis[14]. This policy is however myopic. It only guarantees to minimize the processing time of the considered inbound truck, but it may worsen the processing time of future arriving trucks. The designation of doors as strip or stack door and the mid-term assignment of the outbound trucks is found by solving the *destination-door allocation problem* (DDAP). The objective function of this problem is the expected value of the travel time with respect to several representative scenarios. A scenario describes the arrival times and the contents of the inbound trucks and is based on actual

[14]Because time aspects are taken into account, this can in fact be considered as (dynamic) scheduling of the inbound trucks.

data instead of averages. In this way, the variability in freight flow is taken into account in a similar way as by Bozer and Carlo [28]. The applied on-line policy is also taken into account by the objective function. Two heuristics are provided to solve the DDAP: a local search heuristic and a genetic algorithm. The authors performed a computational study using simulated data patterned after actual data. The results show that both heuristics can reduce the total travel time with about 20 % compared with current practice.

3.4.6 Truck scheduling

In the previous section, the assignment of trucks to dock doors was discussed. Temporal constraints were not taken into account; it was not possible to assign multiple trucks to the same door sequentially. The *truck scheduling problem* on the other hand considers the dock doors as resources (used by the trucks) that have to be scheduled over time. The problem decides on the succession of inbound and outbound trucks at the dock doors of a cross-dock: *where and when* should the trucks be processed.

In fact, the assignment problem is part of the truck scheduling problem. As mentioned in the previous section, this assignment can be executed on a short-term or mid-term horizon. Usually, the truck scheduling problem assigns the trucks to dock doors on a short-term horizon. In this case, trucks with the same origin or destination can be assigned to different dock doors. However, for the mid-term assignment, the origins and destinations of the trucks are assigned to doors instead of the trucks and the truck scheduling problem reduces to the sequencing of all trucks of equal origin or destination.

This section discusses papers that deal with the truck scheduling problem. The characteristics of the cross-docks considered in these papers are summarized in Table 3.2. The papers are also classified according to the classification scheme for deterministic truck scheduling problems proposed by Boysen and Fliedner [26] (see Table 3.3)[15]. The classification is based on three basic elements of any truck scheduling problem which are noted as a 'tuple': the 'door environment', operational characteristics and the objective. For each of these three main elements, several attributes are specified. For instance, some attributes of the operational characteristics are pre-emption (allowed or not), processing time to load or unload a truck (fixed or not for all trucks), intermediate storage (allowed or not), etc.

[15] Wang and Regan [222] and McWilliams [128] propose (dispatching) rules to dynamically assign trucks to dock doors, so these two papers cannot be classified according to the scheme of Boysen and Fliedner [26] and consequently, they are not included in Table 3.3.

Table 3.2: Characteristics of the papers discussed in Section 3.4.6. A '∗' indicates that not a single value of the characteristic is valid, but that all values can be used, 'n/a' or 'ns' indicate that a characteristic is not applicable or not specified.

Paper(s)	Shape	No. of doors	Internal transport	Service mode	Pre-emption	Arrival pattern	Departure time	Interchange-ability	Temporary storage
Chen and Lee [37]	n/a	2	ns	exclusive	no	concentrated	no	truck	yes
Chen and Song [38]	∗	∗	ns	exclusive	no	concentrated	no	truck	yes
Yu and Egbelu [234], Vahdani and Zandieh [191], Arabani et al. [10]	n/a	2	automated	exclusive	no	concentrated	no	allowed	yes
Boysen et al. [27]	n/a	2	∗	exclusive	no	concentrated	no	allowed	yes
Forouharfard and Zandieh [64]	n/a	2	ns	exclusive	no	concentrated	no	allowed	yes
Arabani et al. [9]	n/a	2	automated	exclusive	no	concentrated	outbound	allowed	yes
Vahdani et al. [190], Soltani and Sadjadi [178]	n/a	2	automated	exclusive	yes	concentrated	no	allowed	no
Sadykov [163]	n/a	2	ns	exclusive	no	concentrated	no	allowed	yes
Larbi et al. [106]	n/a	2	ns	exclusive	yes	scattered	no	destination	yes
Alpan et al. [6]	∗	∗	ns	exclusive	yes	scattered	no	destination	yes
Boysen and Fliedner [26]	∗	∗	ns	exclusive	no	concentrated	outbound	destination	yes
Rosales et al. [161]	∗	∗	manually	exclusive	no	concentrated	no	truck	yes
Wang and Regan [222]	∗	∗	manually	exclusive	no	scattered	no	destination	no
Acar [3]	∗	∗	manually	exclusive	no	scattered	no	destination	no
Konur and Golias [100]	∗	∗	n/a	exclusive	no	scattered	no	n/a	n/a
McWilliams et al. [131], McWilliams [128]	∗	∗	automated	exclusive	no	concentrated	no	destination	no
Chmielewski et al. [40]	∗	∗	manually	exclusive	no	concentrated	both	destination	yes
Lim et al. [117], Miao et al. [135]	∗	∗	manually	mixed	no	scattered	both	truck	yes
Boysen [25]	∗	∗	manually	exclusive	no	concentrated	no	truck	no
Shakeri et al. [175], Li et al. [114]	∗	∗	manually	mixed	no	concentrated	no	truck	yes

Table 3.3: Classification of most papers discussed in Section 3.4.6 according to the classification scheme proposed by Boysen and Fliedner [26]. When a certain attribute is not applicable, the default value is assumed to be valid.

Paper(s)	Notation
Chen and Lee [37]	$[E2\|t_j=0\|C_{max}]$
Chen and Song [38]	$[E\|t_j=0\|C_{max}]$
Yu and Egbelu [234], Vahdani and Zandieh [191], Arabani et al. [10]	$[E2\|\text{change}\|C_{max}]$
Boysen et al. [27]	$[E2\|p_j=p,\text{change}\|C_{max}]$
Forouharfard and Zandieh [64]	$[E2\|\text{change}\|\sum S_p]$
Arabani et al. [9]	$[E2\|\text{change}\|*]$
Vahdani et al. [190], Soltani and Sadjadi [178]	$[E2\|\text{pmtn,no-wait,change}\|C_{max}]$
Sadykov [163]	$[E2\|\text{limit,change}\|\sum S_p]$
Larbi et al. [106]	$[E2\|\text{pmtn}\|*]$
Alpan et al. [6]	$[E\|\text{pmtn}\|*]$
Boysen and Fliedner [26]	$[E\|t_{io},\text{fix}\|\sum w_sU_s]$
Rosales et al. [161]	$[E\|t_{io}\|*]$
Acar [3]	$[E\|r_j,\text{no-wait}\|*]$
Konur and Golias [100]	$[E\|r_j\|*]$
McWilliams et al. [131]	$[E\|p_j=p,\text{no-wait},t_{io}\|C_{max}]$
McWilliams et al. [132]	$[E\|\text{no-wait},t_{io}\|C_{max}]$
McWilliams [129]	$[E\|p_j=p,\text{no-wait}\|*]$
McWilliams [130]	$[E\|\text{no-wait}\|*]$
Chmielewski et al. [40]	$[E\|r_j,\tilde{d}_j,\text{limit},t_{io}\|*]$
Lim et al. [118]	$[M\|r_j,\tilde{d}_j,\text{limit},t_j=0\|*]$
Lim et al. [117], Miao et al. [135]	$[M\|r_j,\tilde{d}_j,\text{limit},t_{io}\|*]$
Boysen [25]	$[E\|p_j=p,\text{no-wait},t_j=0\|\sum C_o]$ $[E\|p_j=p,\text{no-wait},t_j=0\|*]$ $[E\|p_j=p,\text{no-wait},t_j=0\|\sum T_o]$
Shakeri et al. [175], Li et al. [114]	$[M\|t_{io}\|C_{max}]$

Single strip and stack door

Several authors consider a simplified cross-dock with a single strip and a single stack door to study the truck scheduling problem. Truck scheduling reduces in this case to the sequencing of the inbound and outbound trucks.

Chen and Lee [37] consider the so-called *two-machine cross-docking flow shop problem*. The objective is to sequence the inbound and outbound trucks in order to minimize the makespan, i.e. the time span from the start of the unloading of the first inbound truck until the end of the loading of the last outbound truck. The problem is modeled as a two-machine flow shop problem, but with additional precedence constraints to make sure that an outbound truck cannot be processed (on the second machine) before all its predecessor tasks have been completed (on the first machine). The load and unload times can be different for each truck (e.g. based on the actual content) and can possibly include the travel time. Pre-emption is not allowed and it is assumed that all trucks are available at the beginning of the planning horizon. Unloaded products can be temporarily put in storage (with infinite capacity) until the appropriate outbound truck is docked. The authors prove that this problem is strongly NP-hard and present a heuristic approach based on Johnson's rule (which solves the two-machine flow shop problem). A branch-and-bound algorithm to solve the problem optimally is also provided. Computational results show that the B&B algorithm can solve problems with up to 60 trucks in a reasonable amount of time.

This problem is extended by Chen and Song [38] to the *two-stage hybrid cross-docking scheduling problem*. Now multiple trucks can be loaded or unloaded at the same time by considering parallel machines at the inbound and outbound 'stage'. The travel time between the inbound and outbound docks is not taken into account. The authors provide a mixed integer programming model of this problem and propose several heuristics based on Johnson's rule to solve it.

Yu and Egbelu [234] also study a cross-dock with a single strip and a single stack door. Similar to the two-machine approach, the objective is to minimize the makespan, but now products are assumed to be interchangeable. So, the product assignments from the inbound trucks to the outbound trucks have to be determined additionally. Also different is that a truck changeover time is considered and the travel time between the strip and stack doors has been fixed. It is assumed that the inbound trucks can be unloaded in any sequence. The problem is formulated as a mixed integer programming model. To solve large problem instances, a heuristic algorithm is proposed. The heuristic method is tested on several small problem instances and the results indicate that the solutions are close to the optimal solutions obtained by complete enumeration

(percentage deviation between 0 and 11.13 %).

Vahdani and Zandieh [191] elaborate further on this problem and apply five metaheuristic algorithms to solve it: a genetic algorithm, tabu search, simulated annealing, an electromagnetism-like algorithm and variable neighborhood search. For these five metaheuristics, the solution obtained with the heuristic developed by Yu and Egbelu [234] is used as an initial solution or as a member of the initial population. The computational experiments show that these metaheuristics can improve the solutions obtained by the heuristic of Yu and Egbelu [234] at the expense of a slightly higher computation time. Arabani et al. [10] also present five metaheuristics to tackle this problem: a genetic algorithm, tabu search, particle swarm optimization, ant colony optimization and differential evolution.

Boysen et al. [27] deal with a very similar problem, but on a more aggregate level. The time horizon is divided into discrete time slots and it is assumed that the trucks can be completely loaded or unloaded within such a time slot. The authors formulate the problem as an integer programming model and show that this problem is NP-hard in the strong sense. To solve it, a decomposition approach is proposed in which two subproblems are considered: given a fixed inbound sequence, determine the optimal outbound sequence and vice versa. By solving these two subproblems iteratively until a stopping criterion is met, a global solution is found. These subproblems can be solved suboptimally with a heuristic approach or exactly by a (bounded) dynamic programming approach.

Some other papers deal with very similar problems. Forouharfard and Zandieh [64] try to sequence the inbound and outbound trucks in order to minimize the number of products that pass through temporary storage. The authors propose an imperialist competitive algorithm (ICA) to solve the problem. Arabani et al. [9] consider still another objective function. It is assumed that the outbound trucks have a due date and the objective is to minimize the total (weighted) earliness and tardiness of these trucks. Three metaheuristics are proposed to solve this problem: a genetic algorithm, particle swarm optimization and differential evolution. Vahdani et al. [190] consider also a similar problem, but now temporary storage is not allowed. To make it possible that the freight is directly shipped from inbound to outbound truck, the loading and unloading of the trucks can be halted and continued at a later point in time (pre-emption). The authors formulate the problem as an integer programming model and propose two metaheuristics to solve it: a genetic algorithm and an electromagnetism-like algorithm. Soltani and Sadjadi [178] present two metaheuristics (hybrid simulated annealing and hybrid variable neighborhood search) to tackle the same problem.

The truck scheduling problem for a cross-dock with a single strip and a single stack door is studied from a more theoretical point of view by Sadykov [163].

The objective is the minimization of the number of products that are temporary stored. Absolute times are not considered, but based on the (relative) sequence of inbound and outbound trucks the number of orders that can be directly transferred or have to be stored can be determined. It is assumed that the capacity of the storage area is limited, that all trucks are available at the beginning of the planning horizon and that products are interchangeable. Pre-emption is not allowed. The author formulates the problem as a MIP model and proves that this problem is NP-hard in the strong sense. For a simplified case in which the sequences of incoming and outgoing trucks are fixed, a dynamic programming algorithm is provided that can solve this case in polynomial time.

Larbi et al. [106] consider only the scheduling of the outbound trucks in a cross-dock with a single strip and a single stack door. An arriving inbound truck is unloaded and the products with the destination of the current outbound truck are directly loaded. The other goods can be temporarily put in storage (with infinite capacity), or the outbound truck can be moved to a parking zone, liberating the stack door for another truck (pre-emption of the loading operation). It is assumed that the outbound trucks are available at any time and that the unloading can be done in any order. The loading, unloading and travel times are not considered. The objective is to find the best schedule of outbound trucks that minimizes the total cost (storage and pre-emption costs). The authors distinguish between three cases with different levels of information about the inbound trucks. In the first case, full information is assumed, i.e. the sequence of the inbound trucks and the content of all trucks are known. A graph based algorithm is proposed that can solve this case in polynomial time. In the second case, it is assumed that no information about the inbound trucks is available. Only the daily quantities to ship to each destination are known in advance. The content of an inbound truck and its arrival time is only known upon arrival. For this case, the authors propose a heuristic based on a probabilistic decision rule to determine which outbound truck should be loaded next. In the third case, partial information is available. When an inbound truck arrives, the content and the sequence of a certain number (Z) of inbound trucks that will arrive next is also revealed. Two heuristic methods are presented. For the first heuristic, the approach proposed for the full information case is adapted for a rolling horizon. The second heuristic combines the algorithms for the full information and the no information case. The first heuristic is recalculated every time a new truck arrives (so every piece of new information is taken into account), while the second heuristic only has to be recalculated when Z trucks have arrived. The performed numerical experiments indicate that the total cost increases significantly if no information is available. When only partial information is available, there is also an extra cost, but this extra cost quickly decreases as Z increases. The numerical results also suggest that in this case the second heuristic gives better results.

This problem is extended by Alpan et al. [6] to a cross-dock with multiple strip and stack doors (for the case with full information). To solve the problem optimally, the authors propose a graph based dynamic programming approach. Because the number of nodes increases exponentially with the problem size, two strategies are examined to limit the number of nodes generated at each stage of the dynamic programming model.

Scheduling of inbound trucks

Other papers consider a more realistic cross-dock with multiple strip and stack doors, but deal only with the scheduling of the inbound trucks. It is assumed that the outbound trucks are already scheduled or are assigned on a mid-term horizon (i.e. the destinations are assigned to stack doors).

In addition to a classification scheme, Boysen and Fliedner present in [26] an optimization model for the case in which a fixed outbound schedule is used. The outbound trucks depart at predefined points in time, regardless of the loaded freight. For instance postal services usually apply fixed schedules. All shipments that arrive before the departure of the truck are loaded, the other shipments are postponed until the next departure to the same destination. The objective is then to schedule the inbound trucks in order to minimize the (weighted) number of delayed shipments. The model takes the travel time between the assigned inbound and outbound doors into account. The authors prove that this model is NP-hard in the strong sense.

Rosales et al. [161] study the scheduling of inbound trucks at a large cross-dock facility in Georgetown. The scheduling is performed for the period of one shift. The objective is to minimize the operational cost and to provide a balanced workload to all workers. The operational cost consists of two parts: the travel cost (proportional to the travel distance) and the labor cost. Because one worker is assigned to work at each dock, minimizing labor cost amounts to minimizing the number of docks required to handle the freight. Overtime is allowed, but it comes at an extra cost. The travel times are dependent on the door assignment of the inbound trucks, while the unload times are estimated based on the composition and volume of the freight. It is assumed that all trucks are available at the beginning of the shift and that pre-emption is not allowed. Goods can be temporarily stored near to the (scheduled) stack doors. The authors formulate the problem as a mixed integer programming model that includes constraints to enforce workload balancing. Computational experiments show that CPLEX© is able to solve realistically sized problems in a reasonable time and outperforms the current (manual) approach. By explicitly including workload-balancing constraints, the number of used docks can be

reduced with only little impact on the travel distance. The proposed model is also implemented at the Georgetown cross-dock and leads to a cost reduction and a better balanced workload.

Wang and Regan [222] also consider the scheduling of inbound trucks and propose some (dispatching) rules that are applicable in a dynamic environment. When a strip door becomes available, and multiple inbound trucks are waiting to be unloaded, one of these trucks has to be handled first. Usually, the next truck is chosen based on the FCFS policy. This is a fair rule with respect to the waiting time of the inbound trucks, but it may not lead to the optimal result for the cross-dock as a whole. In a cross-dock, the travel time between the docks is usually small compared with the time the products have to wait inside the trucks or at the docks. So, the authors propose two time-based algorithms that are concerned with the impact of a new inbound truck on the total processing or total transfer time. The processing time of a product consists of the waiting time at the strip door (inside the truck), the travel time between strip and stack door and the waiting time inside the outbound truck. The transfer time considers also the waiting time before the inbound truck is docked. It is assumed that there is always an outbound truck available for each destination, so there is no temporary storage space needed. The unloading of the trucks cannot be interrupted (no pre-emption) and the arrival times are scattered throughout the day. The authors performed a simulation study to compare both algorithms with the FCFS rule and the look-ahead policy proposed by Gue [75]. They conclude that significant time savings can be obtained by using the proposed time-based rules, at least when the average number of waiting trucks is higher than 0.65.

Another approach to schedule only the incoming trucks is taken by Acar [3]. In his master thesis, the objective is not to minimize the travel distance inside the cross-dock, but to have an assignment that is robust against the variability in system parameters such as truck arrival times, service times (for loading, unloading and transferring freight) and the truck loads. The author formulates the problem as a mixed integer quadratic programming (MIQP) problem to minimize the variance associated with the distribution of the idle times of the docks. Indeed, an assignment with even distribution of idle times at the strip docks will tend to absorb the stochastic variability in the arrival and service times. It is assumed that there is always an outbound truck docked at each stack door, so there is no temporary storage space needed. The truck arrival times are taken into account and pre-emption is not allowed. For each inbound truck, there can be a different service time (to unload and move its content). Because of the computational complexity, a simple heuristic algorithm is proposed. Some experimental tests on small problem instances indicate that this heuristic gives results on average within 4.41 % of the optimal solution (but the maximum

deviation is about 16 %). The author also proposes a dynamic heuristic to assign the trucks to docks at real time.

Konur and Golias [100] also take the uncertainty of the truck arrivals into account. It is assumed that the arrival time windows of the incoming trucks are known, but not the exact arrival times. Only the unloading operations are considered, internal transport and loading are not taken into account. The unload times depend on the truck (e.g. based on the actual load) and the assigned dock door. Pre-emption of the unloading operations is not allowed. The operational cost that is considered consists of two components for each inbound truck: a processing cost depending on the inbound door the truck is assigned to and a cost associated with the truck's waiting time. The scheduling problem is then formulated as a bi-objective bi-level optimization problem that tries to minimize the average total cost and the cost range. The average cost is the arithmetic average of the possible maximum and minimum cost (for all feasible truck arrival times), while the cost range is the difference between these two costs. By explicitly considering the cost range, the aim is to obtain a cost-stable schedule, i.e. a truck schedule for which the variation in cost as a result of truck arrival uncertainty is small. The authors propose a genetic algorithm to find Pareto efficient solutions. This approach is compared with two FCFS policies in several numerical experiments. The results indicate that the proposed algorithm is efficient in determining schedules that have a low average cost with a low cost range.

McWilliams et al. [131] consider the truck scheduling of inbound trucks at a cross-dock used in the parcel delivery industry. In such a cross-dock, unloaded parcels are transported to outbound trucks by means of a fixed network of conveyors. Because of this stationary network, the designation of doors as either strip or stack doors is fixed and the route of a parcel is defined by its assigned strip and stack door. The travel time of a parcel is dependent on its route, but also on congestion of the conveyor network. The objective of this *parcel hub scheduling problem* (PHSP) is then to minimize the time interval from the unloading of the first parcel until the loading of the last parcel (makespan). It is assumed that all trucks are available at the beginning of the time horizon and that pre-emption is not allowed. As full outbound trucks are immediately replaced, goods do not have to be intermediately stored. In [131], it is assumed that the batch sizes (and the unload times) of the inbound trucks are equal, while this assumption is relaxed in [132]. Because a conveyor network is a queueing network, it is difficult to develop an analytical model of its behavior. So, the authors propose a simulation-based scheduling algorithm (SBSA) to solve the PHSP. This algorithm is a genetic algorithm that makes use of a detailed deterministic simulation model to evaluate the makespan for each candidate solution. Computational results show a significant reduction in the

makespan (between 4.2 and 35.8 %) compared with arbitrary scheduling (as a representation of current practice).

Simulation optimization is however computationally expensive and requires excessive computing time to obtain solutions for large-scale problems. So, McWilliams [129] proposes a decomposition approach to tackle the PHSP. A combination of time-based and resource-based decomposition is applied. The time horizon is divided into several smaller sub-periods (time buckets) and the focus is on the bottleneck resources (the final sorters of the conveyor network). The objective is to minimize the maximum workload at the final sorters over all time buckets. This workload balancing problem is formulated as a (NP-hard) minimax programming model. The time between unloading and arriving at the bottleneck is assumed to be independent of the used strip door. In [129], it is assumed that the batch sizes (and the unload times) of the inbound trucks are equal and a genetic algorithm is used to solve the problem. This assumption is relaxed in [130] and the problem is solved by applying a local search algorithm and simulated annealing. Computational results indicate that the genetic algorithm finds solutions with a significant lower makespan than the SBSA while the computation time is more than a factor 10 lower. The local search and simulated annealing in turn seem to improve the results of the genetic algorithm in a similar computation time.

In the previous approaches, the workload at the final sorters is balanced in a static way. In [128], McWilliams presents a dynamic load balancing algorithm (DLBA). Whenever an unload dock becomes idle, one of the waiting inbound trucks has to be assigned to the idle dock. The objective is to balance the flow of parcels through the conveyor network and to avoid flow congestion. The algorithm makes use of updated information on the availability of inbound trucks and the state of the cross-dock. Computational results show that the proposed algorithm (applied in a static context) gives, for large problem instances, significant better results than the static approach in [130], and this in a much shorter computation time.

Chmielewski et al. [40] also study the scheduling of inbound trucks, but the authors consider at the same time the assignment of the outbound trucks on a mid-term horizon (i.e. the destinations are assigned to stack doors). Unloaded goods are placed in a buffer area, from where they are transported to a buffer area for loading (at each stack door). The workers and resources needed to perform this transportation are limited, and also the size of the buffer areas is limited. Pre-emption is not allowed and an earliest arrival and latest departure time are defined for each truck. One objective is to find an optimal schedule that leads to minimal total distances and a minimal number of required resources. A second objective is the minimization of waiting times. Trucks should be allocated to a door as soon as possible after their arrival. The authors

propose two solution approaches. In a first approach, the problem is modeled as a time-discrete, multicommodity flow problem with side constraints. The objective is to minimize the total cost (based on the travel distance). The costs increase slightly with time in order to take also the second objective (to minimize the waiting time) into account. To solve this mixed integer problem, the authors propose a decomposition-and-column-generation approach. The second approach makes use of a multi-objective evolutionary algorithm (EA) that results in a set of Pareto optimal solutions. This is a real multi-criteria approach that tries to minimize the total travel distance and the total waiting time. Two variants are considered: $(1 + 1)$-EA with one offspring each iteration and $(\mu + \lambda)$-EA with multiple offspring each iteration. Computational results show that the decomposition-and-column-generation approach outperforms the standard algorithm for MIP (branch-and-bound with CPLEX©) in terms of lower objective function values and better feasible solutions. However, this approach can only be used for a limited number of discrete time periods because otherwise the flow network becomes much too large. The computation times of the EA algorithms are much lower, but at the expense of solution quality; the total distance of the solutions is much higher. However, the waiting times are much better, due to the multi-objective approach.

Scheduling of inbound and outbound trucks

The following papers deal with the scheduling of both inbound and outbound trucks.

Lim et al. [118] consider a truck scheduling problem in which it is assumed that the trucks are loaded or unloaded during a fixed time window. This means that the scheduling problem is reduced to determining at which dock door the trucks have to be processed. The length of these time windows can be interpreted as the time needed to load or unload a truck. The objective of this so-called *truck dock assignment problem* is to minimize the total travel distance. The trucks can be assigned to any door (mixed service mode) and the capacity of the cross-dock is limited. Pre-emption is not allowed and trucks that cannot be served are penalized by adding an extra distance. A shortcoming of this approach is that the time to transport freight between the dock doors is not taken into account. The authors formulate the problem as an integer programming model and because the problem is NP-hard, they propose a tabu search and a genetic algorithm approach to solve it.

The same authors extend this approach by taking the travel time between the docks into account [117, 135]. The objective is now to minimize the operational cost (based on travel time) and the cost of unfulfilled shipments. A similar tabu

search heuristic [135] and an adapted genetic algorithm [117, 135] to solve this truck scheduling problem are discussed. The experimental test results indicate that the genetic algorithm outperforms CPLEX© in terms of solution quality and computing time. The tabu search approach in turn seems to dominate the genetic algorithm.

Boysen [25] deals with truck scheduling for a cross-dock in which products are not allowed to be intermediately stored. Such a zero-inventory policy is for instance used when frozen goods are transported and the cross-docking terminal is not cooled. To make sure that the cooling chain is not broken, goods are not allowed to be intermediately stored. This policy can be applied in several industrial sectors, but the paper focuses on the food industry. As a result, products are dedicated to a specific outbound truck and are not interchangeable. In the food industry, standardized cargo carriers and trailers are used, so it is assumed that docking, unloading and undocking of trucks take a very similar amount of time. The author also assumes that the travel times of goods inside the cross-dock are negligible because of the small size of cross-docking terminals in the food industry. Each dock door is exclusively dedicated to inbound or outbound operations (exclusive service mode). The author presents a formalization of the truck scheduling problem that can take into account different operational objectives (minimization of flow time, processing time and tardiness of outbound trucks). To solve this problem optimally, a dynamic programming approach is proposed in which an acyclic directed graph is constructed. The shortest path in this graph then corresponds to the optimal solution. This approach can be extended by applying lower and upper bounds (bounded dynamic programming). The author also presents a simulated annealing procedure. A computational study shows that the (bounded) dynamic programming approach can be used to solve smaller problem instances (up to 25 inbound trucks) optimally within a few minutes. For realistic (larger) problem sizes, the simulated annealing approach is able to find near-optimal results in less than 1 s. The author also indicates how this method can be used as part of a rolling horizon approach.

Shakeri et al. [175] study the truck scheduling problem in a cross-dock where goods are exchanged between the trucks, i.e. each truck serves both as inbound and as outbound truck. The problem is modeled as a two-stage parallel-machine scheduling problem and the objective is to minimize the makespan. In the unloading stage, goods are unloaded and moved to the temporary storage (with infinite capacity) at the correct dock door. It is assumed that the different goods of a truck can be unloaded in parallel. The moving can only start after unloading and when the destination truck is docked. The travel time is based on the distance between the dock doors. In the loading stage, the goods are (sequentially) loaded into the trucks. The loading of a truck can only start if its own goods are unloaded and all products that have to be loaded are available in

the storage area. It is also assumed that all trucks are available at the beginning of the planning horizon and that pre-emption is not allowed. Between two consecutive trucks, a set-up time is taken into account. The authors provide a (non-linear) mixed integer programming formulation of the problem that can be used for small problem instances.

To solve larger problem instances, a heuristic method is presented by Li et al. [114]. This dependency ranking search (DRS) heuristic consists of two parts. The first part builds a feasible sequence of jobs with respect to the number of dock doors. In the second part, these jobs are assigned to doors based on the distance between the doors. Some computational experiments were performed and the results show that, for the small instances, the CPLEX© solver performs slightly better than the DRS heuristic. However, CPLEX© is much slower. For medium and large problem instances, CPLEX© is not able to find solutions (in a time limit of 2 h) for most cases, while the heuristic finds a good solution in more than 8 of the 10 instances (in a few minutes).

3.4.7 Temporary storage

Although the idea of cross-docking is to unload products from trucks and directly load the products into departing trucks, temporary storage is usually inevitable. Freight has to be staged because of the imperfect synchronization of inbound and outbound trucks and because the goods do not arrive in the sequence in which they must be loaded. The loading sequence is for instance determined by the need to build tightly packed loads or to place fragile products on top, or by the order in which the goods have to be delivered if there are multiple stops [17]. Usually, a dispatching rule is used to determine where the freight has to be staged, for instance in front of the dock door where the outbound truck is or will be docked. There are however papers that deal with the operational decision where to store incoming freight. The characteristics of the considered cross-docks are summarized in Table 3.4.

A first study is performed by Vis and Roodbergen [219]. In this paper, the aim is to determine temporary storage locations for incoming freight such that the total travel distance of the goods is minimized. It is assumed that the dock door assignment and the travel distances are known. The authors show that this problem can be modeled as a minimum cost flow problem (MCFP), for which several polynomial time algorithms exist. A storage location can however be used only once in this approach. Therefore, the authors propose to solve the problem multiple times, each time taking the freight for the corresponding period into account. As a result, storage locations can be used multiple times. Numerical experiments are performed to compare the proposed method with a

situation in which the workers choose the storage locations and which usually results in loads stored at available locations nearest to the origins of the loads. The results show that the proposed algorithm can reduce the total travel distance up to about 40 %.

Werners and Wülfing [224] try to optimize the temporary storage locations at a parcel sorting center of Deutsche Post World Net. Arriving freight is transported to the loading areas by means of conveyors. In these loading areas, the freight is temporary stored in 'endpoints', which are situated next to the conveyors. From there, the parcels are manually carried (with roll containers) to the dock doors for further transport. This manual transport is time consuming and costly. So, the objective of the paper is to minimize the travel distances between the endpoints and dock doors. It is assumed that the outbound trucks for the parcels are known. The authors formulate this problem as a linear assignment problem. The model determines where the freight has to be temporary stored (in which endpoint) and at which dock door the outbound trucks have to be assigned. The model ensures that trucks cannot be docked at the same door if their departure time is too close to each other and that the load is equally distributed over the four sections of the facility. To obtain a nearly optimal solution, the authors use a hierarchical decomposition approach. In order to deal with data uncertainty (fluctuations in the quantities of arriving freight), the authors also present an adaptation to find robust solutions. The experimental results show that the proposed approaches can improve the load balancing over the four sections compared with the current situation. Moreover, the proposed approaches offer a reduction in (weighted) travel distance of 37 to 39 % compared with the current situation.

In his master thesis, Sandal [170] uses simulation to compare several staging strategies in order to support the optimal loading of the outbound trucks. The author distinguishes three cases that determine which freight is staged: no freight is staged (pure cross-docking), all freight is staged and the loading only starts when all goods are stored, or the goods that will (seriously) violate the scheduled loading sequence are staged while the other freight is loaded directly. When the freight is staged, two strategies can be distinguished. In the first strategy, the storage area before each stack door is treated as a single FCFS queue. In the second strategy, these storage areas are divided into three equal zones and freight is placed in one of these zones based on its ranking in the scheduled loading sequence.

Table 3.4: Characteristics of the papers discussed in Section 3.4.7. A '∗' indicates that not a single value of the characteristic is valid, but that all values can be used, 'ns' indicates that a characteristic is not specified.

Paper(s)	Shape	No. of doors	Internal transport	Service mode	Pre-emption	Arrival pattern	Departure time	Interchange-ability	Temporary storage
Vis and Roodbergen [219]	∗	∗	manually	exclusive	ns	ns	ns	truck	yes
Werners and Wülfing [224]	U	∗	combination	exclusive	no	ns	outbound	destination	yes
Sandal [170]	I	6	manually	exclusive	no	concentrated	no	destination	yes

Table 3.5: Characteristics of some papers discussed in Section 3.4.8. A '∗' indicates that not a single value of the characteristic is valid, but that all values can be used, 'ns' indicates that a characteristic is not specified.

Paper(s)	Shape	No. of doors	Internal transport	Service mode	Pre-emption	Arrival pattern	Departure time	Interchange-ability	Temporary storage
Li et al. [113], Álvarez-Pérez et al. [7]	∗	∗	ns	ns	no	scattered	both	truck	yes
Stickel [182]	∗	∗	manually	exclusive	no	scattered	outbound	destination	yes
Magableh et al. [121]	∗	∗	manually	exclusive	no	scattered	outbound	destination	yes

3.4.8 Other issues

The following papers deal with still other cross-docking issues. Table 3.5 summarizes the characteristics of the considered cross-docks (for the papers in which the real world characteristics are described).

Li et al. [113] consider the scheduling of internal resources for the loading and unloading of freight. The loading and unloading process is usually accomplished by teams of workers and equipment. Since the number of available teams is limited[16], these teams have to be scheduled efficiently. The objective is to complete the processing of each truck as close as possible to its due date (just-in-time). The authors model this problem as a two-phase parallel machine scheduling problem with earliness and tardiness penalties and formulate it as an integer programming model. This scheduling problem is NP-hard, so two heuristic approaches are proposed. Both approaches use a genetic algorithm and try to improve the best solution of each generation. The first approach applies a local search heuristic (squeaky wheel optimization), while the second approach solves the integer programming subproblem that results when the assignment of teams to trucks is fixed and only the start and end time of the loading and unloading processes can change. Experimental results indicate that both approaches find near-optimal solutions in a much shorter time than CPLEX©. The second approach gives the best result, but at the expense of a longer computation time.

A different method to solve the same problem is proposed by Álvarez-Pérez et al. [7]. This method is a combination of Reactive GRASP (greedy randomized adaptive search procedure) and tabu search. The Reactive GRASP procedure is used to construct initial solutions which are improved by the tabu search algorithm. The numerical experiments suggest that this method performs similar or slightly better compared with the heuristics of Li et al. [113], but is in turn more time-consuming for the larger problem instances.

Stickel [182] deals with the problem in which not only the scheduling of the internal resources is considered, but also the vehicle routing problem that appears for the pick-up and delivery of goods and the scheduling of the inbound and outbound trucks. Compared with previous approaches, these three types of problems are integrated and solved simultaneously. The author proposes two solution approaches: a centralized-hierarchical and decentralized-heterarchical approach. The centralized-hierarchical approach corresponds to the situation in which a central instance (like a third-party logistics provider) has all relevant

[16]This depends on the applied human resource strategy. In fact, if the employment contract stipulates that workers can be sent home or called from home depending on the work load, the number of workers can be considered as unlimited.

information and can take the necessary decisions. The problem is formulated as a mixed integer problem. However, because of its complexity, only small problem instances can be solved by applying branch-and-bound (with CPLEX©). In the second approach, it is assumed that there is not a single entity that has all decision power, but several entities have to cooperate. The truck scheduling is interpreted as an interface between the vehicle routing and the scheduling of the internal resources and time slots at the dock doors are allocated among the cooperating entities by means of a combinatorial auction.

Yan and Tang [232] compare the costs of a traditional distribution center with the cost of pre-distribution and post-distribution cross-docking. In pre-distribution cross-docking, it is assumed that the goods are directly loaded into outbound trucks. The suppliers are responsible for the necessary preparation and sorting to facilitate immediate loading at the cross-dock. This requires that the suppliers know the order quantities for each destination. In post-distribution cross-docking, the preparation and sorting happens at the cross-dock itself. This incurs higher costs at the cross-dock, but allows assigning the goods to destinations upon arrival at the cross-dock. In this way, the influence of the fluctuating demand can be reduced by pooling the risk during the transportation period from the supplier to the cross-dock. The authors construct analytical models to perform a pair-wise comparison of the cost (including inventory, back order and operational costs) of the three systems. It is assumed that the demand is correlated between two adjacent periods, but independent between different destinations. The cost of the different systems depends on several parameters (e.g. delivery lead time, unit holding cost and variation in demand). Numerical experiments are performed to study the influence of these parameters on the preference of cross-docking. The results indicate that pre-distribution cross-docking is preferred when the demand is stable and the lead time between supplier and cross-dock is short. However, when the demand is uncertain and the lead time is long, the benefits of reallocating goods among stores outweigh the higher operational costs and so post-distribution cross-docking is preferred. Post-distribution also seems to be preferable if the number of destinations or the unit holding cost increases.

In [187], the same authors compare in a similar way pre-distribution and post-distribution cross-docking when transshipments are allowed; goods can be shipped from an overstocked destination to a nearby understocked destination in order to avoid back orders. Post-distribution cross-docking has higher operational costs than pre-distribution cross-docking, but will need less transshipments due to the pooled demand. It is assumed that the demand is independent in time and between different destinations. An analytical formulation of the costs (including transshipment costs) for both systems is provided and the cost sensitive factors are analyzed. The results of numerical

experiments suggest that a higher uncertainty of demand, a higher unit transshipment costs and a longer lead time from the supplier to the cross-dock make post-distribution cross-docking more preferred. Pre-distribution cross-docking is more preferable when the unit holding cost or unit back order cost is very high or very low.

Simulation is a general technique that also can be used to deal with several aspects of cross-docking. For instance, simulation allows comparing alternative cross-dock layouts or can be used to test various dock door assignment strategies, and this for one or more selected performance metrics. Some of the papers discussed above make use of simulation (e.g. [32, 76, 131, 149, 170, 222]).

In [159], Rohrer explains that simulation on the one hand is useful to determine whether all the equipment will function together properly and to test different design alternatives. On the other hand, simulation can also be used to test alternative control algorithms before the actual implementation. The author lists also some issues that have to be taken into account while modeling a cross-dock and provides some useful performance metrics.

Magableh et al. [121] present a simulation model that represents the operations within a cross-dock, specifically the processing of inbound and outbound shipments. The authors tried to make the model generic so that it can easily be expanded to model other cross-docking facilities. The presented model can for instance be used to analyze the effect of an increased demand or to compare the performance of different dispatching rules.

3.5 Conclusions and research opportunities

As can be noted, a considerable amount of papers about cross-docking have been published, certainly during the recent years. Several papers deal with cross-docking in a general way (e.g. suitability for cross-docking and the implementation of cross-docking), while other papers are concerned with a specific type of problem (on a strategic, tactical or operational level). Especially the problems of dock door assignment (Section 3.4.5) and truck scheduling (Section 3.4.6) have attracted the attention of many researchers. Despite this attention, there are still many opportunities to improve and extend the current research.

First of all, not all problems with which cross-docking practitioners are confronted are extensively discussed. For instance, only a few papers about cross-dock layout design (Section 3.4.2) are published. These papers deal with the shape of the cross-dock and the design of the storage area. Other aspects,

like the dimension of the cross-dock and the dimension, shape and arrangement of the internal cross-dock areas, are however not considered. There are also not many papers that deal with temporary storage (Section 3.4.7), while a good strategy can improve the cross-docking operations, for instance by avoiding excessive travel distances and congestion.

In the second place, not all types of cross-docks are considered. As can be seen in Tables 3.1, 3.2, 3.4 and 3.5, the same characteristics are appearing and some characteristics do almost not occur. For instance, only a few papers deal with a conveyor network for the internal transportation. While the use of forklifts may indeed be more common in industry, there are also many cross-docks that make use of a conveyor network (e.g. parcel carriers). As the choice for an automated system imposes some restrictions (exclusive service mode, fixed routes) and gives rise to some other issues (congestion), it would be interesting to specifically consider this type of cross-docking. Also, most papers study cross-docks with an exclusive mode of service and without pre-emption. While this can simplify the planning and execution of the daily operations, it limits the flexibility of the cross-dock. So, future research could be performed to determine how pre-emption and a mixed service mode can be correctly applied in order to improve the cross-docking operations. Moreover, not many papers take restrictions on departure times (deadlines) for the trucks into account. Also, only a few papers assume that goods are interchangeable, while this is not an exceptional situation (e.g. inbound trucks arriving at the cross-dock of a retailer containing only one product type destined for several branches, the distribution of newspapers, ...). So, it would be interesting if future research also considers restricted departure times and interchangeable products.

Thirdly, many of the presented papers make simplifying assumptions that limit the real-world applicability. For instance, it is usually assumed that the loading and unloading of a truck can be done in any order. Also, internal congestion is not taken into account in most papers (except in e.g. [16, 131, 132]), value added activities like repacking or labeling are usually not considered and trucks are inbound or outbound, but not both (except in [114, 175]). Table 3.6 gives some other examples for the various problem types discussed in Section 3.4. Future research should address these assumptions in order to make the proposed approaches more applicable in practice.

In the fourth place, also to improve the applicability, the approaches should be more robust and dynamic. In the presented papers, it is usually assumed that all necessary information, for instance about the incoming loads (e.g. the exact content and arrival time), is fixed and known in advance. However, discussions with cross-docking practitioners reveal that there are (serious) deviations between the predicted and actual information. For instance, it is not unusual that the weight of an arriving load is higher than indicated

Table 3.6: Illustrative list of simplifying assumptions.

Problem type	Example(s) of simplifying assumptions
Location of cross-docks	All trucks have same capacity and costs, transportation costs are proportional with shipping quantity (instead of number of required trucks)
Layout design	Travel distances do not account for temporary storage, congestion is not considered
Cross-docking networks	Time windows are neglected, storage time and costs are not taken into account, individual trucks are not considered
Vehicle routing	All trucks have same capacity and costs, loading and unloading sequence is neglected, loads are not-splittable
Dock door assignment	Infinite capacity for temporary storage, travel distance is used as a measure of travel time (neglecting e.g. congestion)
Truck scheduling	All trucks are available at beginning of time horizon, unloading can start immediately (required workforce and material is always available)
Temporary storage	Congestion is neglected, arrival and departure times are not considered

on the cargo documents, which can possibly cause an overloaded outbound truck. Some other examples are given in Table 3.7. So, robustness against these kinds of deviations is required in order to be applied in an industrial context. Only three of the discussed papers propose a robust approach. In [3], the objective is to obtain a schedule of the inbound trucks that is robust against variability in the arrival and service times. In [100], an approach is presented to obtain a cost-stable truck schedule under arrival time uncertainty and in [224], an assignment of goods to storage locations is determined that is robust against fluctuations in the arriving freight quantities. Moreover, most of the presented papers are not appropriate for a dynamic environment, as the considered (operational) problems are assumed to be static. Of course, this is a simplification of reality. The control of a cross-dock is a going concern and so these problems are inherently dynamic (trucks arrive early or late, equipment fails, ...). Consequently, real time decisions are necessary. A few papers propose a (simple) dynamic approach (e.g. [3, 128, 222, 233]) or explain how the proposed static approach can be applied as part of a rolling horizon approach

Table 3.7: Illustrative list of shortcomings regarding robustness and dynamics.

Problem type	Example(s) of shortcomings
Location of cross-docks	Demand pattern and costs are assumed to be known (and deterministic)
Layout design	Deterministic flow pattern (door assignment is assumed to be known)
Cross-docking networks	Transportation time, costs and quantities are assumed to be known (and deterministic)
Vehicle routing	Deterministic transportation and (un)load times
Dock door assignment	Freight flows from strip doors to destinations are known (and deterministic)
Truck scheduling	Deterministic travel times and costs
Temporary storage	Arrival times and loads are assumed to be known (and deterministic)

(e.g. [25]), but certainly more research in this direction is required.

Lastly, in practice, cross-dock practitioners have to deal with several problems together. While some of the presented papers tackle more than one problem (e.g. [32, 170, 182] and the papers discussed in Section 3.4.1), most papers are concerned with just one problem. Furthermore, as these problems are interdependent, improvements are expected when they can be solved together. So, future research is required that integrates several problems in one approach. For instance, it would be interesting to combine truck scheduling with the routing of the trucks. On the one hand, the routing schedules of the inbound trucks determine the arrival times at the cross-dock, which in turn influence the scheduling of the trucks. Also, the routings of the outbound trucks influence the truck scheduling by setting deadlines for the trucks. On the other hand, if the truck schedule also determines which goods have to be combined in one truck, this influences the routing of the outbound trucks. As both problems are interdependent, it makes sense to combine them. This provides more alternatives to the decision maker, which allows better solutions but also makes the decision making problem more difficult.
The truck scheduling and the scheduling of the resources inside the cross-dock are also interdependent problems that can be combined. The scheduling of the trucks heavily influences the workload for the internal resources. For instance, the assignment of the trucks to dock doors determines the travel distance for the workers. Also, by not correctly spreading the workload (in

space and time), congestion can occur inside the terminal. Conversely, the resource scheduling determines the time lag between the inbound and outbound operations and influences in this way the truck scheduling. So, solving both problems simultaneously can improve the cross-docking operations.
The scheduling of the trucks and the internal resources is also interdependent with the packing and unpacking of loads. The time at which a certain item can be unloaded is dependent on the way the inbound truck is packed, while the loading sequence of the outbound trucks determines if goods can be directly moved from inbound to outbound or have to be temporarily stored. Consequently, this influences the involved material handling and the time lag between the unloading and loading operations. There is a trade-off between the optimal packing of the trucks which involves more material handling, and less material handling but a worse packing of the trucks (and possibly more trucks are required to transport the same amount of freight). So, it could be interesting to combine the scheduling with the packing decisions. Moreover, the packing also influences the vehicle routing as it imposes restrictions on the sequence of customer visits.
Other problems that are interdependent and for which benefits can arise by solving or considering them together are the scheduling of trucks and the unloading strategy for the work force (e.g. workers unload a truck completely before unloading another truck (trailer-at-a-time [32]), workers can unload a certain number of trucks together, ...), the layout design and the temporary storage strategy, and the vehicle routing and the routing of goods through a cross-docking network.

This chapter introduced the cross-docking concept and described several cross-docking problems. The next chapter elaborates on two of these problems, particularly the problems of truck scheduling and vehicle routing. For both problems, a heuristic method is developed in order to find good solutions in an acceptable amount of time. These solutions then will support the cross-docking HLES (described in Chapter 5) to improve its performance.

Chapter 4

Cross-dock scheduling

As indicated in Chapter 2, a staff holon can provide the basic holons with expert knowledge, for instance from a centralized scheduling algorithm. The next chapter describes an HLES implementation to coordinate and control the cross-docking operations. This HLES can be supported in finding good global solutions by scheduling advice about cross-dock related problems. This chapter presents two such scheduling algorithms. Section 4.1 describes a truck scheduling algorithm to assign the inbound and outbound trucks to the various dock doors of a cross-dock[1]*. The resulting truck schedule can be used by the resource holon in charge of the cross-dock in order to define the details of the operations the orders have to execute on the cross-dock. Section 4.2 discusses a vehicle routing algorithm to assign the orders to the various trucks in order to be transported from and to the cross-dock. The order holons can use the resulting vehicle routing schedule as an advice. This chapter only describes the algorithms, how the HLES cooperate with these algorithms is explained in Chapter 5. The conclusions of this chapter are presented in Section 4.3.*

4.1 Truck scheduling problem

As explained in Chapter 3, the truck scheduling problem is concerned with the assignment of inbound and outbound trucks to the different dock doors of a cross-dock. The dock doors can be seen as resources that have to be

[1]This section is an edited version of J. Van Belle et al. "A Tabu Search Approach to the Truck Scheduling Problem with Multiple Docks and Time Windows". *Computers & Industrial Engineering*. Forthcoming.

scheduled over time. A solution of the truck scheduling problem defines *where* (at which dock door) and *when* a truck should be processed. A truck scheduling algorithm then has to find a solution that is 'optimal' with regard to a certain objective function (e.g. the minimization of the makespan or the total travel distance). In Chapter 3, three different categories of truck scheduling problems are distinguished. The first category considers a simplified cross-dock with a single strip and a single stack door and truck scheduling reduces to the sequencing of the inbound and outbound trucks. The problems in the second category consider cross-docks with multiple strip and stack doors, but deal only with the scheduling of the inbound trucks. The last category then considers the scheduling of both inbound and outbound trucks at multiple dock doors.

The truck scheduling problem considered here belongs to the third category. The following basic assumptions are made.

- An exclusive mode of service is considered, i.e. each dock door is either exclusively assigned to inbound or to outbound trucks (e.g. one side of the cross-dock is dedicated to inbound trucks and the other side to outbound trucks).
- Arriving goods are unloaded from the inbound trucks and transferred to the appropriate outbound dock where they are loaded into outbound trucks. Other internal operations - like sorting and labeling - are not considered.
- Sufficient personnel and equipment are assumed available for performing all loading, unloading and transferring operations.
- Pre-emption of loading or unloading a truck is not allowed. So, a docked truck has to be completely processed before it leaves the dock.
- For each truck, the (expected) arrival time is known.
- Departure times are defined for all trucks. The departure times are however not considered as hard constraints, but the tardiness of the trucks with respect to these times should be minimized.
- The transported freight is shipped in standardized cargo containers (e.g. pallets). As a consequence, the time required to load or unload one product unit is assumed to be fixed.
- The freight is loaded and unloaded sequentially, i.e. only one freight unit can be loaded or unloaded at the same time. So, the loading or unloading time of a truck is directly proportional to the number of freight units.
- The time needed to transfer goods from inbound to outbound trucks is directly proportional to the distance between the dock doors to which the trucks are assigned.
- Intermediate storage inside the cross-dock is allowed. This means that goods can be unloaded from an inbound truck before the appropriate outbound truck is available. The capacity of the storage area is infinite.

- The truck changeover time is fixed.
- Products are not interchangeable, i.e. any arriving product unit is dedicated to a specific outbound truck.
- The sequence in which goods are loaded or unloaded is not taken into account.

The objective is a weighted combination of two objectives. On the one hand, transferring all the goods from inbound to outbound trucks has to be optimized in order to minimize the total workload. On the other hand, the total tardiness of the trucks, with respect to the assigned departure times, has to be minimized. So, the considered objective function is the weighted sum of the total travel time and the total tardiness.

In accordance with the classification scheme proposed by Boysen and Fliedner [26] (see Chapter 3), the considered problem can be represented by $[E|r_j, t_{io}|*]$.

The truck scheduling problem considered here differs in several aspects from other truck scheduling problems in literature. First of all, it deals with the scheduling of both inbound and outbound trucks at multiple dock doors (category 3), while many authors consider problems of the first or second category. For instance, Yu and Egbelu [234] consider a truck scheduling problem of the first category. The objective function (minimization of makespan) is different as well and no arrival and departure times are considered. Also, the products are assumed to be interchangeable and consequently the product assignments from the inbound to the outbound trucks have to be determined. The same problem is also studied in other articles [10, 191]. As an example of the second category, McWilliams et al. [131] consider the scheduling of inbound trucks at a cross-dock used in the parcel delivery industry. In such a cross-docking terminal, unloaded parcels are transported to outbound trucks by a fixed network of conveyors. The congestion of the conveyor network is taken into account in the travel time of the parcels. So, the travel time is not only dependent on the assignment of trucks to dock doors as assumed here. Another difference is the objective function as the authors try to minimize the makespan. In [129, 130], a similar problem is considered, but with still another objective function. The objective is now to balance the workload. A dynamic version of this problem is also studied [128].

A few articles also deal with truck scheduling problems of the third category. Miao et al. [135] for instance study a simplified problem in which it is assumed that the trucks are loaded or unloaded during a predetermined time window. As a consequence, the scheduling problem is reduced to determining the assignment of trucks to dock doors. Other differences with the problem presented here are that the dock doors are not strictly divided into strip and stack doors and that the capacity of the cross-dock is limited. The objective is to minimize the

operational cost (based on travel time) and the cost of unexecuted shipments. Another truck scheduling problem belonging to the third category is examined by Boysen [25]. In contrast to the problem considered here, products are not allowed to be intermediately stored and the travel times of the products inside the cross-dock are assumed to be negligible. Three different objectives can be taken into account: minimization of flow time, processing time or tardiness of the outbound trucks.

To define the considered truck scheduling problem formally, a mathematical model is presented in the next section. Section 4.1.2 then presents a heuristic method to solve this problem. The results of experimental tests to validate this method are given in Section 4.1.3. Next, Section 4.1.4 describes possible extensions and limitations of the mathematical model and the solution approach.

4.1.1 Mathematical model

The problem is formulated as a mixed integer programming (MIP) model. The problem consists of n trucks (n_1 inbound trucks and n_2 outbound trucks) and m dock doors (m_1 strip doors and m_2 stack doors). The following parameters are used:

n_1	number of inbound trucks
n_2	number of outbound trucks
m_1	number of strip doors
m_2	number of stack doors
f_{ij}	number of product units that have to be transported from inbound truck i to outbound truck j
v_{ij}	1 if product units have to be transported from inbound truck i to outbound truck j, 0 otherwise
t_{kl}	travel time between strip door k and stack door l
a_i	arrival time of inbound truck i
b_j	arrival time of outbound truck j
c_i	departure time of inbound truck i
d_j	departure time of outbound truck j
L	time needed to load or unload one product unit
T	truck changeover time
w_1	weighting factor for the total travel time
w_2	weighting factor for the total tardiness
M	big number

The following continuous decision variables are defined:

r_i	start time of inbound truck i (time at which truck i enters the dock)
s_j	start time of outbound truck j (time at which truck j enters the dock)
e_i	end time of inbound truck i (time at which truck i leaves the dock)
g_j	end time of outbound truck j (time at which truck j leaves the dock)
h_i	tardiness of inbound truck i
u_j	tardiness of outbound truck j

Finally, the following binary decision variables are used[2]:

$$x_{ik} \begin{cases} 1 & \text{if inbound truck } i \text{ is assigned to strip door } k \\ 0 & \text{otherwise} \end{cases}$$

$$y_{jl} \begin{cases} 1 & \text{if outbound truck } j \text{ is assigned to stack door } l \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ijkl} \begin{cases} 1 & \text{if inbound truck } i \text{ is assigned to strip door } k, \text{ outbound truck } j \text{ is assigned to stack door } l \text{ and } v_{ij} = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} \begin{cases} 1 & \text{if inbound trucks } i \text{ and } j \text{ are assigned to the same strip door and truck } i \text{ is a predecessor of truck } j \\ 0 & \text{otherwise} \end{cases}$$

$$q_{ij} \begin{cases} 1 & \text{if outbound truck } i \text{ and } j \text{ are assigned to the same stack door and truck } i \text{ is a predecessor of truck } j \\ 0 & \text{otherwise} \end{cases}$$

The truck scheduling problem can then be formulated as follows:

$$\min \quad w_1 \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{m_1} \sum_{l=1}^{m_2} f_{ij}\, t_{kl}\, z_{ijkl} \;+\; w_2 \left(\sum_{i=1}^{n_1} h_i + \sum_{j=1}^{n_2} u_j \right)$$

[2]Note that the variables z_{ijkl} are required to make the formulation linear.

subject to

$$\sum_{k=1}^{m_1} x_{ik} = 1 \qquad \forall i = 1 \dots n_1 \quad (4.1)$$

$$\sum_{l=1}^{m_2} y_{jl} = 1 \qquad \forall j = 1 \dots n_2 \quad (4.2)$$

$$\sum_{k=1}^{m_1} \sum_{l=1}^{m_2} z_{ijkl} = v_{ij} \qquad \forall i = 1 \dots n_1, \forall j = 1 \dots n_2 \quad (4.3)$$

$$z_{ijkl} \leq x_{ik} \qquad \begin{array}{r} \forall i = 1 \dots n_1, \forall j = 1 \dots n_2, \\ \forall k = 1 \dots m_1, \forall l = 1 \dots m_2 \end{array} \quad (4.4)$$

$$z_{ijkl} \leq y_{jl} \qquad \begin{array}{r} \forall i = 1 \dots n_1, \forall j = 1 \dots n_2, \\ \forall k = 1 \dots m_1, \forall l = 1 \dots m_2 \end{array} \quad (4.5)$$

$$x_{ik} + x_{jk} - 1 \leq p_{ij} + p_{ji} \qquad \forall i, j = 1 \dots n_1, i \neq j, \forall k = 1 \dots m_1 \quad (4.6)$$

$$p_{ij} + p_{ji} \leq 1 \qquad \forall i, j = 1 \dots n_1 \quad (4.7)$$

$$y_{il} + y_{jl} - 1 \leq q_{ij} + q_{ji} \qquad \forall i, j = 1 \dots n_2, i \neq j, \forall l = 1 \dots m_2 \quad (4.8)$$

$$q_{ij} + q_{ji} \leq 1 \qquad \forall i, j = 1 \dots n_2 \quad (4.9)$$

$$r_j \geq a_j \qquad \forall j = 1 \dots n_1 \quad (4.10)$$

$$r_j \geq e_i + T - M(1 - p_{ij}) \qquad \forall i, j = 1 \dots n_1 \quad (4.11)$$

$$e_i \geq r_i + L \sum_{j=1}^{n_2} f_{ij} \qquad \forall i = 1 \dots n_1 \quad (4.12)$$

$$s_j \geq b_j \qquad \forall j = 1 \dots n_2 \quad (4.13)$$

$$s_j \geq f_i + T - M(1 - q_{ij}) \qquad \forall i, j = 1 \dots n_2 \quad (4.14)$$

$$g_j \geq s_j + L \sum_{i=1}^{n_1} f_{ij} \qquad \forall j = 1 \dots n_2 \quad (4.15)$$

$$g_j \geq e_i + \sum_{k=1}^{m_1} \sum_{l=1}^{m_2} t_{kl} z_{ijkl} + f_{ij} L - M(1 - v_{ij}) \qquad \forall i = 1 \ldots n_1, \forall j = 1 \ldots n_2 \quad (4.16)$$

$$h_i \geq e_i - c_i \qquad \forall i = 1 \ldots n_1 \quad (4.17)$$

$$u_j \geq g_j - d_j \qquad \forall j = 1 \ldots n_2 \quad (4.18)$$

$$r_i \geq 0, e_i \geq 0, h_i \geq 0 \qquad \forall i = 1 \ldots n_1 \quad (4.19)$$

$$s_j \geq 0, g_j \geq 0, u_j \geq 0 \qquad \forall j = 1 \ldots n_2 \quad (4.20)$$

The objective is to minimize the weighted sum of the total travel time and the total tardiness. Constraints (4.1) ensure that every inbound truck is assigned to a strip door and similarly, constraints (4.2) ensure that every outbound truck is assigned to a stack door. Constraints (4.3)–(4.5) define the correct relationship between the x_{ik}, y_{jl} and z_{ijkl} variables. The correct relationship between the x_{ik} and p_{ij} variables for the inbound trucks is expressed by constraints (4.6) and (4.7). Note that constraints (4.7) enforce that $p_{ii} = 0$. In a similar way, constraints (4.8) and (4.9) express the relationship between the y_{il} and q_{ij} variables for the outbound trucks. Constraints (4.7) and (4.9) are redundant and can be omitted, but including these constraints results in smaller computation times. Constraints (4.10) and (4.11) then determine the start time of each inbound truck as the maximum of the arrival time of the truck and the end time of its predecessor plus truck changeover time:

$$r_j = \max(a_j, \max_{i=1 \ldots n_1} p_{ij}(e_i + T)) \quad (4.21)$$

Due to the assumptions of sequential unloading and sufficient resource availability (personnel and equipment), the end time of each inbound truck is equal to its start time plus the time required to unload all products:

$$e_i = r_i + L \sum_{j=1}^{n_2} f_{ij} \quad (4.22)$$

This is expressed by constraints (4.12). For the outbound trucks, the start time can be defined in a similar way as for the inbound trucks:

$$s_j = \max(b_j, \max_{i=1 \ldots n_2} q_{ij}(f_i + T)) \quad (4.23)$$

This is enforced by constraints (4.13) and (4.14). The end time is at least as great as its start time plus the time required to load all products. However, the end time is also constrained by the end times[3] of the inbound trucks that ship products for the outbound truck. The end time has to be greater than or equal to the latest end time of these inbound trucks augmented with the time to transfer the appropriate product units to the outbound truck and the time to load these units. So, the end time of the outbound trucks can be expressed as follows:

$$g_j = \max(s_j + L\sum_{i=1}^{n_1} f_{ij}, \max_{i=1\ldots n_1} v_{ij}(e_i + \sum_{k=1}^{m_1}\sum_{l=1}^{m_2} t_{kl}z_{ijkl} + f_{ij}L)) \quad (4.24)$$

In this expression, $\sum_{k=1}^{m_1}\sum_{l=1}^{m_2} t_{kl}z_{ijkl}$ denotes the travel time from inbound truck i to outbound truck j. The expression is enforced by constraints (4.15) and (4.16). Note, however, that this does not completely prevent parallel loading (of goods from different inbound trucks) if multiple inbound trucks have similar values of end time plus transfer time. Therefore, the loading sequences of the outbound trucks also have to be determined. This requires introducing extra variables and constraints and will be omitted in order not to unnecessarily complicate the formulation. Therefore, equation (4.24) can be considered as a simplified formulation. For the tabu search approach however, an effective - yet simple - algorithm was developed for determining the correct end times (see Section 4.1.2). Constraints (4.17) and (4.19) then determine the tardiness of each inbound truck, which is defined as:

$$h_i = \max(e_i - c_i, 0) \quad (4.25)$$

Similarly, the tardiness of each outbound truck is given by:

$$u_j = \max(g_j - d_j, 0) \quad (4.26)$$

This is expressed by constraints (4.18) and (4.20). All continuous decision variables have to be greater than or equal to zero, which is expressed by constraints (4.19) and (4.20).

This problem formulation is quite large. For n trucks (n_1 inbound trucks and n_2 outbound trucks) and m dock doors (m_1 strip doors and m_2 outbound dock

[3]As the unloading sequence of the inbound trucks is unknown, it is assumed that goods from an inbound truck can only be transferred after complete unloading of the truck, i.e. after its end time.

doors), the total number of variables is given by:

$$\begin{aligned} &3n_1 + 3n_2 + n_1m_1 + n_2m_2 + n_1n_2m_1m_2 + n_1^2 + n_2^2 \\ &\quad = 3n + n_1(n_1 + m_1) + n_2(n_2 + m_2) + n_1n_2m_1m_2 \end{aligned}$$

The total number of constraints is:

$$\begin{aligned} &n_1 + n_2 + 3n_1n_2m_1m_2 + (n_1 - 1)n_1m_1 + n_1^2 + (n_2 - 1)n_2m_2 \\ &\quad + n_2^2 + n_1 + n_1^2 + n_1 + n_2 + n_2^2 + n_2 + n_1n_2 + n_1 + n_2 \\ &\quad\quad = 2n^2 + 4n + (n_1 - 1)n_1m_1 + (n_2 - 1)n_2m_2 + 3n_1n_2(m_1m_2 - 1) \end{aligned}$$

For instance, a moderate cross-dock with 5 strip doors, 5 outbound dock doors, 10 inbound trucks and 10 outbound trucks requires a MIP model with 2860 variables and 8980 constraints. In order to avoid long computation times (as reported in Section 4.1.3), a heuristic method is proposed to solve it in a reasonable amount of time. The next section describes the applied solution approach.

4.1.2 Tabu search approach

A tabu search (TS) approach [70, 71, 173] was developed for the truck scheduling problem. In the next paragraphs, the details of this approach are described.

Solution representation A solution of this truck scheduling problem can be represented by a sequence of pairs of dock doors and trucks (see Figure 4.1). For n trucks, the solution will have a length of n. The first n_1 pairs correspond to the strip doors (ID) and inbound trucks (IT), the next n_2 pairs correspond to the stack doors (OD) and outbound trucks (OT). Figure 4.2 shows an example solution. The first row corresponds to the dock doors, the second row is a permutation of the trucks. If multiple trucks are assigned to the same dock door, the sequence in which these trucks appear in the solution defines the sequence in which the trucks are assigned to the dock door. For instance, for the solution shown in Figure 4.2, the sequence of the inbound trucks at strip door 1 is 1–3–5. This sequence completely defines the truck schedule. The start times of the trucks can be determined by equations (4.21) and (4.23) and the end times of the inbound trucks by equations (4.22). The end times of the outbound trucks can be determined either by equation (4.24) or by Algorithm 4.1. In Algorithm 4.1, the exact end times of the outbound trucks are determined (together with the start times) by considering their optimal loading sequences with respect to the tardiness (i.e. based on the first-come, first-served policy).

Algorithm 4.1 Pseudocode of the algorithm to determine the exact end times of the outbound trucks of a solution of the truck scheduling problem.

```
function CALCULATEENDTIMES(ID, IT, OD, OT)
    for h ← 1 ... n2 do                    ▷ Iterate over the outbound trucks corresponding to their sequence in the solution.
        j ← OT_h   l ← OD_h
        s_j ← max(b_j, dt_l)               ▷ The start time of outbound truck j is the maximum of its arrival time and the time dock l is available (dt_l).
        g_j ← s_j
        for g ← 1 ... n1 do                ▷ Iterate over the inbound trucks corresponding to their sequence in the solution.
            i ← IT_g   k ← ID_g
            lt_i ← 0                       ▷ Determine the time at which goods from truck i can be loaded into truck j (lt_i). If truck i contains goods for truck j, lt_i is equal to the end time of truck i augmented with the travel time.
            if v_ij = 1 then
                lt_i ← e_i + t_kl
            end if
        end for
        ITS ← SORT(IT, lt)                 ▷ Sort the inbound trucks by lt_i (first-come, first-served).
        for g ← 1 ... n1 do                ▷ Iterate over the inbound trucks in order of their lt_i.
            i ← ITS_g
            g_j ← max(lt_i, g_j) + Lf_ij   ▷ The end time of loading from truck i into truck j is equal to the start time (the maximum of lt_i and the end time of loading from the predecessor of truck i) augmented with the loading time.
        end for
        dt_l ← g_j + T                     ▷ Adapt the time dock l is available.
    end for
end function
```

ID	OD
IT	OT

Figure 4.1: Solution representation for the truck scheduling problem.

1	2	1	2	1	2	1	2	1	2
1	2	3	4	5	1	2	3	4	5

Figure 4.2: Example solution representation for the truck scheduling problem ($n_1 = n_2 = 5$ and $m_1 = m_2 = 2$).

Initial solution The inbound and outbound trucks are sorted by arrival times. The strip and stack doors are sorted by their average distance to the stack and strip doors respectively. So, the most 'central' doors have the highest positions in the list. The sorted trucks are then assigned one by one to the next dock door in the sorted list. If a truck is assigned to the last dock door in the list, the next truck is again assigned to the first dock door of the sorted list. For instance, if the arrival sequence of both inbound and outbound trucks is 1–2–3–4–5, and the sorted lists of the dock doors are 1–2 and 2–1, the solution shown in Figure 4.2 would be the initial solution.

Neighborhood A composite neighborhood structure is used, consisting of two neighborhoods based on the following two moves:

- Swap move: two trucks are interchanged, i.e. the first truck is assigned to the dock door and the position in the sequence of the second truck, and vice versa (see Figure 4.3). If both trucks were assigned to the same dock door, only their position in the sequence would change.
- Insert move: a truck is assigned to another dock door (see Figure 4.4). The position of the truck at this dock door is determined by the position in the sequence of the solution.

Note that swap moves do not change the number of trucks assigned to the same dock door. Insert moves however allow that the trucks are redistributed over the dock doors. Both moves have to be applied to either the inbound part or the outbound part of the solution. The total number of different moves is then equal to:

$$n_1(n_1 - 1)/2 + n_1(m_1 - 1) + n_2(n_2 - 1)/2 + n_2(m_2 - 1)$$

1	2	1	2	1	2	1	2	1	2
1	**2**	3	4	**5**	1	2	3	4	5

↓

1	2	1	2	1	2	1	2	1	2
1	**5**	3	4	**2**	1	2	3	4	5

Figure 4.3: Example of a swap move for the truck scheduling problem: the inbound truck sequence at strip doors 1 and 2 changed from 1–3–5 and 2–4 to 1–3–2 and 5–4.

1	2	1	2	1	2	1	**2**	1	2
1	2	3	4	5	1	2	3	4	5

↓

1	2	1	2	1	2	1	**1**	1	2
1	2	3	4	5	1	2	3	4	5

Figure 4.4: Example of an insert move for the truck scheduling problem: the outbound truck sequence at stack doors 1 and 2 changed from 2–4 and 1–3–5 to 2–3–4 and 1–5.

As this number can become quite large, it is possible to limit the number of neighbors in order to allow more iterations in a certain time frame. This can be done by setting a lower value of the parameter q. The total number of moves is then limited to $4q$ (q swap moves in the inbound part, q insert moves in the inbound part, q swap moves in the outbound part and q insert moves in the outbound part). If the total number of moves of one type is smaller than q, all moves of that type are considered, otherwise q random moves of that type are generated.

Tabu list A tabu list with a fixed tenure t is used. The recent moves are stored in this list, not the recent solutions. The swap moves are characterized by the indices of the trucks that are interchanged (2 and 5 for the example in Figure 4.3), the insert moves by the index of the truck that is assigned to another dock door and that dock door (8 and 1 for the example in Figure 4.4). No aspiration criterion is used.

Termination criterion The tabu search is stopped when the current best solution has not improved during i consecutive iterations or when the maximum allowed calculation time c has elapsed.

The proposed tabu search algorithm is implemented based on the LORA framework developed by the CODeS research group[4]. To ensure that this local search approach is able to find acceptable solutions, the solutions have been compared with the exact solutions for several test scenarios. The results are described in the next section.

4.1.3 Experimental results

This section describes the experimental tests that were carried out to validate the tabu search approach. First, the experimental set-up is described and the test scenarios are defined. Next, the solutions generated by the tabu search approach are compared with the exact solutions (found by solving the MIP model with CPLEX©) and the results are discussed.

Experimental set-up

Several factors influence the 'hardness' of the truck scheduling problem instances:

1. The problem size, i.e. the number of trucks and dock doors.
2. The ratio of the number of trucks to the number of dock doors. If this ratio is low, the truck scheduling problem is almost reducible to the assignment of trucks to dock doors. In case of a high ratio, sequencing the trucks at the dock doors has to be taken into account as well.
3. The flow mix, i.e. the distribution of the arriving goods to the outbound trucks. This characteristic can be represented by the 'density' of the flow matrix (containing f_{ij}). If this matrix is sparse, the inbound trucks contain products for only one or a few outbound trucks. If the flow matrix is dense, the inbound trucks transport goods for (almost) all outbound trucks.
4. The simultaneousness of the truck arrivals. Sequencing the trucks becomes more difficult if the trucks arrive within narrow time frames. If the truck arrivals are better spread over time, the optimal sequence can be derived from the arrival sequence.
5. The tightness of the time windows (formed by the arrival and departure times).

For all experiments, the I-shaped cross-dock shown in Figure 4.5 is considered. The cross-dock has three strip doors and three stack doors and the (rectilinear) travel times between the strip and stack doors are indicated on the figure. The

[4]CODeS, KAHO St.-Lieven, KU Leuven, Gebroeders De Smetstraat 1, 9000 Gent, Belgium.

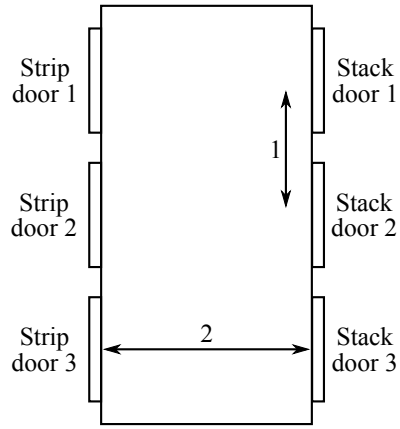


Figure 4.5: Schematic representation of the cross-dock considered for the experimental tests.

aforementioned factors have been reflected during the generation of different problem instances by varying three parameters:

Number of trucks By changing the number of trucks while the number of dock doors is fixed, variations can be obtained in both factors 1 and 2. This parameter can have one of the following three values:

- Low: 4 or 5 inbound trucks and 4 or 5 outbound trucks.
- Medium: 6 or 7 inbound trucks and 6 or 7 outbound trucks.
- High: 8 or 9 inbound trucks and 8 or 9 outbound trucks.

Flow mix This parameter determines the number of outbound trucks for which the inbound trucks contain products (factor 3). There are three values:

- Low: each inbound truck contains items for 25 to 50 % of the outbound trucks.
- Medium: each inbound truck contains items for 50 to 75 % of the outbound trucks.
- High: each inbound truck contains items for 75 to 100 % of the outbound trucks.

The flow matrix (containing f_{ij}) can be determined based on this value. First, the outbound trucks to which the items are transferred are randomly chosen for each inbound truck. The number of items to be transferred to these outbound trucks is randomly sampled between 1 and C/nbr, in which C is the capacity (the total number of items one truck can transport) and nbr is the number of outbound trucks for which the inbound truck

Table 4.1: Fixed parameter values for the truck scheduling problem instances.

Parameter	Value
L	2
T	3
w_1	1
w_2	2
C	33
q	250
t	16
i	10000
c	5 s

contains items. If this procedure leads to empty or overloaded trucks, the resulting flow matrix is discarded and the same procedure is repeated.

Time window This parameter determines how the arrival and departure times of the trucks are generated (and influences factors 4 and 5). For each truck, the arrival and departure times are randomly sampled in a certain time interval. The length t of this time interval is directly proportional to the number of trucks arriving or leaving in that interval: $t = a * nbr$ for the arrival times and $t = d * nbr$ for the departure times, in which nbr is the number of (arriving or leaving) trucks. The values of a and d are determined by one of the three values of this parameter:

- Low: $a = 30$ and $d = 15$.
- Medium: $a = 20$ and $d = 10$.
- High: $a = 10$ and $d = 5$.

For the inbound trucks, the arrival times are sampled from the interval $[0, t]$. If e is the average arrival time of the inbound trucks, the arrival times of the outbound trucks are then sampled from the interval $[e, e + t]$. The departure times of the trucks are sampled from the interval $[f, f + t]$, in which f is the arrival time of the truck.

Each combination of parameter values (27 in total) denotes a problem type. For each problem type 10 instances were generated. The parameters that are fixed are shown in Table 4.1.

Experimental results

All 270 problem instances have been solved with the proposed tabu search (TS) and with the CPLEX© solver[5]. All experiments were run on a PC with two 2.40 GHz hex-core processors and with 96 GB RAM. As it takes CPLEX© a long time to find the optimal solution for the larger problems, an upper bound of two hours (7200 s) is applied. If the optimal solution is not found within this time limit, the current best solution generated by the solver is reported. As explained in Section 4.1.1, the MIP model makes use of the simplified equation (4.24) to calculate the end times of the outbound trucks. For a fair comparison of the resulting objective values, the tabu search implementation makes also use of this simplified expression (TS1). For illustrative purposes, the results of applying the proposed tabu search with exact end times of the outbound trucks (computed by Algorithm 4.1) are also determined (TS2).

Table 4.2 summarizes the results. The first column shows the problem types, indicated by the value (L = low, M = medium, H = high) for its three parameters: number of trucks, flow mix and time window. For each problem type, the average objective value over its 10 instances is calculated. These values are shown in the next three columns for the tabu search approach with the simplified expression (TS1), the tabu search approach with the exact expression (TS2) and the MIP approach (using CPLEX©). The lowest average values for all problem types are indicated in bold. The fifth column indicates how many of the 10 problem instances have been proven optimal by CPLEX©. In order to compare the tabu search (TS1) and the CPLEX© approach, the best of both approaches is determined for each instance. Columns 6 and 7 then show how many times tabu search or CPLEX© approach was the winning approach (in case of a draw, both are counted). The solution of the tabu search approach is then compared with this best solution. If TS_i is the solution of problem instance i found by tabu search and C_i the solution found by CPLEX©, the average relative deviation (in percentage) is calculated as follows:

$$100 \left(\frac{TS_i - \min(TS_i, C_i)}{\min(TS_i, C_i)} \right)$$

The average value for each problem type is shown in the next column. Finally, the average calculation times of the three approaches are shown in the last three columns.

Table 4.2 shows that tabu search is able to find good results in a short time period. For the problems with the lower parameter values, tabu search (TS1)

[5]IBM ILOG CPLEX Optimization Studio 12.5.

Table 4.2: The experimental results for the 27 truck scheduling problem types (L = low, M = medium, H = high).

Problem type	Objective value			# MIP optimal	# TS1 best	# MIP best	Relative dev. (%)	Calculation time (s)		
	TS1	TS2	MIP					TS1	TS2	MIP
L L L	**518.300**	528.299	**518.300**	10	10	10	0.000	0.167	0.455	0.215
L L M	**592.072**	595.331	**592.072**	10	10	10	0.000	0.166	0.457	0.609
L L H	**837.478**	856.069	**837.478**	10	10	10	0.000	0.157	0.433	1.299
L M L	**526.391**	549.479	**526.391**	10	10	10	0.000	0.149	0.541	0.802
L M M	**654.059**	682.388	**654.059**	10	10	10	0.000	0.157	0.568	1.046
L M H	**830.229**	860.750	**830.229**	10	10	10	0.000	0.147	0.511	0.762
L H L	**684.285**	698.018	**684.285**	10	10	10	0.000	0.171	0.811	2.519
L H M	**601.945**	612.169	**601.945**	10	10	10	0.000	0.160	0.721	3.150
L H H	**957.740**	974.641	**957.740**	10	10	10	0.000	0.154	0.682	4.588
M L L	**695.459**	699.873	**695.459**	10	10	10	0.000	0.421	1.820	3.452
M L M	**962.622**	974.925	**962.622**	10	10	10	0.000	0.433	1.754	4.816
M L H	**1236.073**	1254.682	**1236.073**	10	10	10	0.000	0.419	1.751	4.254
M M L	**615.812**	624.649	**615.812**	10	10	10	0.000	0.445	2.747	2.236
M M M	**1004.442**	1021.004	**1004.442**	10	10	10	0.000	0.415	2.498	20.426
M M H	1443.384	1461.398	**1442.850**	10	8	10	0.030	0.445	3.042	79.618
M H L	**875.440**	884.003	**875.140**	10	9	10	0.035	0.430	3.359	12.897
M H M	1150.058	1170.172	**1149.558**	10	9	10	0.050	0.430	3.186	53.642
M H H	**1704.644**	1723.057	**1704.087**	9	9	9	0.065	0.514	3.929	1769.894
H L L	1033.019	1038.157	**1032.119**	10	8	10	0.148	0.902	4.519	12.263
H L M	**1197.026**	1207.316	**1197.013**	10	9	10	0.001	0.859	4.560	35.782
H L H	**1859.446**	1885.563	1861.349	5	9	7	0.025	1.056	4.921	4284.281
H M L	1037.082	1041.362	**1036.773**	10	9	10	0.024	1.027	5.002	704.793
H M M	**1151.924**	1155.059	**1151.924**	9	10	10	0.000	1.017	5.002	1305.639
H M H	1985.397	2006.045	**1982.719**	3	8	6	0.228	1.007	5.001	5218.501
H H L	1281.025	1292.770	**1279.053**	9	6	10	0.189	1.027	5.002	1419.368
H H M	**1712.346**	1729.040	**1712.254**	4	9	8	0.022	1.162	5.002	4897.159
H H H	**2437.626**	2466.332	2439.104	1	7	6	0.055	1.037	5.002	6516.383

was almost always able to find the optimal solution. For all problems, the average relative deviation compared to the best solution found is lower than 0.25 %. When looking at the results of the individual problem instances, the largest observed relative deviation is lower than 2 %. CPLEX© was not able to find the optimal solution for all problem instances within the time limit of two hours. For 30 of the 270 problem instances, the optimal solution was not found. Tabu search was able to find the same solution for 2 out of these 30 instances, and a better solution for 14 problem instances. In total, the tabu search approach found a solution at least as good as the CPLEX© result for 250 problem instances. Tabu search is also very efficient. The solutions of all problem instances were found in less than 2 s, and the average calculation time per problem type is less than 1.2 s.

To prevent loading the outbound trucks in parallel, the second tabu search implementation (TS2) makes use of the exact expression for the end times of the outbound trucks. This implies that the objective values obtained with TS2 are at least as large as the objective values of the MIP model. Comparing the results of TS1 and the MIP approach shows that the average relative deviation between the objective values of both is 1.5 %, with a maximum deviation of 12.5 %. This indicates that also TS2 is able to generate good quality results. The calculation of the exact end times is however computationally more expensive, which results in higher calculation times for TS2 than for TS1.

4.1.4 Extensions and limitations

Although various real-world details are taken into account, many others are not considered. For instance, congestion of the forklift trucks or capacity to temporary store goods are not considered. This section shortly describes some possible extensions and limitations of the proposed model and the solution approach.

Assignment restrictions A first extension could be the inclusion of assignment restrictions, if certain trucks can only be assigned to a subset of dock doors. Such a situation could for instance arise if trucks ship cooled products that can only be transferred inside a cooled area of the cross-dock. These restrictions can be easily included in the MIP model by setting $x_{ik} = 0$ and $y_{jl} = 0$ for all pairs of trucks and dock doors that are not allowed. The proposed tabu search can deal with these restrictions by prohibiting moves that would result in an infeasible assignment. Another possibility is to discard solutions with an infeasible assignment or by adding a penalty to the objective function value of these solutions.

Service mode An exclusive service mode was considered, i.e. each dock door is either exclusively used for inbound or for outbound trucks. A generalization to a mixed service mode is not obvious. The start times of the outbound trucks are dependent on the start times of the inbound trucks and the start times of the inbound trucks are dependent on the end time of its predecessor. As these predecessors can be outbound trucks in a mixed service mode, the determination of the correct start and end times of the trucks becomes much more complicated.

Earliness If next to tardiness also earliness has to be penalized (e.g. in a just-in-time strategy), the MIP model can be easily adapted to calculate the earliness ($\max(c_i - e_i, 0)$ for the inbound trucks and $\max(d_j - g_j, 0)$ for the outbound trucks). However, the proposed tabu search cannot be used as the sequence of the trucks at the dock doors does no longer define the truck schedule completely. In the proposed tabu search, the loading and unloading of the trucks starts and end as early as possible. But, as earliness is penalized, a better objective value can be obtained by starting and ending at a later point of time. So, only the sequence of the trucks at the dock doors is not sufficient to represent one valid solution and another solution representation is required which takes the start and end times of the trucks explicitly into account.

Interchangeable products If products are interchangeable, only the number and type of products to be loaded on the outbound trucks is defined. The exact outbound truck for each order still has to be determined. The mathematical model can be adapted to account for this by making parameters f_{ij} and v_{ij} variables and by introducing extra variables f_{ij}^p for each product type p. Also, constraints have to be added to ensure that the inbound trucks are unloaded and the required number of products is loaded into the outbound trucks. This is difficult to incorporate in the proposed tabu search approach as, for the current solution representation, the flow of goods should be known to determine the end times of the outbound trucks.

4.2 Vehicle routing problem with cross-docking

As explained in Chapter 3, vehicle routing is concerned with the assignment of orders to trucks. Pick-up of the orders at their origins and delivery to their destinations are considered. Both the pickup and the delivery process can be seen as a vehicle routing problem. These problems are however not independent, as the delivery can only take place after consolidation of the goods at the cross-dock.

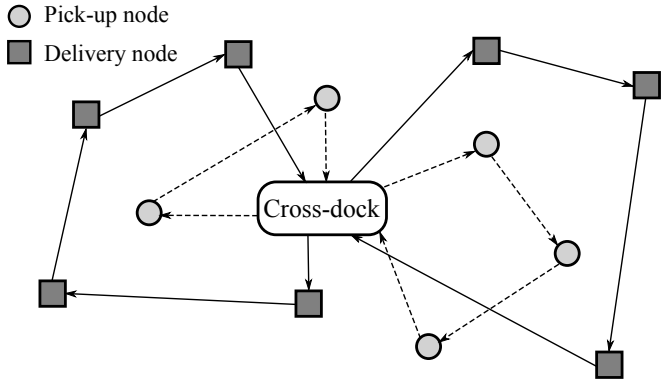


Figure 4.6: The vehicle routing problem with cross-docking.

The vehicle routing problem (VRP) was originally proposed by Dantzig and Ramser [48] and concerns the pick-up (or delivery) of multiple orders by a fleet of vehicles with a limited capacity while minimizing the total travel distance. In the meantime, this problem has been studied extensively (see e.g. [103, 104]) and several variations of the problem have been examined, for instance by considering time windows [29, 30, 72], time-dependent travel times [54, 89, 107, 122, 213], stochastic travel times [105, 107, 179] or stochastic demand [20, 68, 179, 186].

In the context of cross-docking, orders have to be picked up at their origins and transported to a cross-dock. There, they are consolidated and can be transported to their destinations (see Figure 4.6). Time windows are defined both for pick-up and delivery. This so-called vehicle routing problem with cross-docking (VRPCD) [223] can be seen as two coupled VRPs with time windows. The following basic assumptions are made.

- All trucks make at most one tour (a sequence of pick-ups or deliveries), starting and ending at the cross-dock.
- The fleet of trucks is homogeneous, i.e. all trucks have the same capacity.
- Orders are not-splittable, i.e. it is impossible that one truck takes half of the freight units of an order and another truck takes the other half.
- The travel times between the different nodes are known and assumed to be fixed.
- For each order, there is a release time and a due date. These due dates are however not considered as hard constraints, but are taken up in the objective function.

- The availability of personnel and equipment is not taken into account, i.e. there is always personnel and equipment available to perform loading, unloading or transferring.
- The transported freight is shipped in standardized cargo containers (e.g. pallets). As a consequence, the time required to load or unload one product unit is assumed to be fixed.
- The freight is loaded and unloaded sequentially, i.e. only one freight unit can be loaded or unloaded at the same time. So, the loading or unloading time of an order during pick-up, delivery or at the cross-dock is directly proportional to the number of freight units.
- The loading of a delivery truck can only start when all goods that have to be delivered by that truck are available at the cross-dock.
- The time needed to transfer goods from inbound to outbound truck at the cross-dock is assumed to be fixed.
- Intermediate storage inside the cross-dock is allowed. This means that goods can be unloaded from an inbound truck before the appropriate outbound truck is available. The capacity of the storage area is infinite.
- The sequence in which goods are loaded or unloaded is not taken into account.

The objective is the weighted combination of three objectives. Firstly, the total travel time needed to pick-up and deliver all orders has to be minimized. Secondly, as there are due dates defined for each order, the total tardiness of the orders has to be minimized. Thirdly, also the number of trucks (or tours) required to perform the pick-ups and deliveries has to be minimized. So, the considered objective function is the weighted sum of the total travel time, the total tardiness and the number of trucks.

The vehicle routing problem considered here differs in several aspects from other vehicle routing problems in literature. Compared to the problem studied by Lee et al. [108] and Liao et al. [116], the trucks are not required to arrive at the cross-dock simultaneously. Also, the problem presented in this chapter considers release times and due dates of the orders and the objective function includes the tardiness of the orders. In the problem considered by Wen et al. [223], there are also no release times and due dates, but time windows are defined in which the pick-up and delivery should take place. In this problem, the same vehicles are used for pick-up and delivery, which makes unloading and loading of some orders unnecessary. Another difference is that temporary storage is not allowed in this problem and the objective function only tries to minimize the total travel distance of the trucks.

To define the considered vehicle routing problem with cross-docking formally, a mathematical model is presented in the next section. Section 4.2.2 then

presents a heuristic method to solve this problem. Subsequently, the results of experimental tests to validate the proposed approach are given in Section 4.2.3. In Section 4.2.4, the adaptations are described which are made in order to be able to use the algorithm for rescheduling.

4.2.1 Mathematical model

The mathematical formulation presented here is based on the mixed integer programming (MIP) model described in [223][6]. There are n orders, m_1 pick-up trucks and m_2 delivery trucks. For ease of formulation, a dummy order (with index 0) is introduced. This dummy order (consisting of 0 items) is available for pick-up at the cross-dock at time 0 and has to be delivered to the cross-dock (without due date).

The following parameters are used:

n	number of orders
m_1	number of pick-up trucks
m_2	number of delivery trucks
q_i	number of items of order i (0 for $i = 0$)
r_i	release time of order i (0 for $i = 0$)
s_i	due date of order i (∞ for $i = 0$)
c_{ij}	travel time between the origin of order i and the origin of order j (index 0 represents the cross-dock)[7]
d_{ij}	travel time between the destination of order i and the destination of order j (index 0 represents the cross-dock)[7]
C	capacity of the trucks
L	time needed to load or unload one product unit
T	time needed to transfer one product unit from its strip door to the appropriate stack door
w_1	weighting factor for the total travel time
w_2	weighting factor for the total tardiness
w_3	weighting factor for the number of trucks
M	big number

The following continuous decision variables are defined:

u_k	unload time of pick-up truck k (time at which truck k is unloaded at the cross-dock)

[6]The model is however adapted to make sure that all orders are unloaded at the cross-dock from the pick-up trucks and are loaded in different trucks for delivery.

[7]The travel time is 0 if order i and order j have the same origin/destination.

p_k	ready time of delivery truck k (time at which truck k is ready to start its tour)
v_i	unload time of order i (time at which order i is unloaded at the cross-dock)
a_i^k	time at which pick-up truck k starts loading order i
b_i^k	time at which delivery truck k ends unloading order i
t_i	tardiness of order i

Finally, the following binary decision variables are used:

$$x_{ij}^k \begin{cases} 1 & \text{if pick-up truck } k \text{ consecutively picks up orders } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij}^k \begin{cases} 1 & \text{if delivery truck } k \text{ consecutively delivers orders } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

$$z_k^p \begin{cases} 1 & \text{if pick-up truck } k \text{ is used to pick-up orders} \\ 0 & \text{otherwise} \end{cases}$$

$$z_k^d \begin{cases} 1 & \text{if delivery truck } k \text{ is used to deliver orders} \\ 0 & \text{otherwise} \end{cases}$$

The vehicle routing problem with cross-docking can then be formulated as a MIP problem as follows:

$$\min \quad w_1 \left(\sum_{k=1}^{m_1} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij}\, x_{ij}^k + \sum_{k=1}^{m_2} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij}\, y_{ij}^k \right) + w_2 \sum_{i=1}^{n} t_i + w_3 \left(\sum_{k=1}^{m_1} z_k^p + \sum_{k=1}^{m_2} z_k^d \right)$$

subject to

$$\sum_{j=0}^{n} x_{0j}^k = z_k^p \qquad \forall k = 1 \ldots m_1 \quad (4.27)$$

$$\sum_{i=0}^{n} x_{i0}^k = z_k^p \qquad \forall k = 1 \ldots m_1 \quad (4.28)$$

$$\sum_{j=0}^{n} y_{0j}^k = z_k^d \qquad \forall k = 1 \ldots m_2 \quad (4.29)$$

$$\sum_{i=0}^{n} y_{i0}^{k} = z_{k}^{d} \qquad \forall k = 1 \dots m_2 \quad (4.30)$$

$$\sum_{k=1}^{m_1} \sum_{j=0}^{n} x_{ij}^{k} = 1 \qquad \forall i = 1 \dots n \quad (4.31)$$

$$\sum_{k=1}^{m_2} \sum_{j=0}^{n} y_{ij}^{k} = 1 \qquad \forall i = 1 \dots n \quad (4.32)$$

$$\sum_{i=0}^{n} x_{ih}^{k} - \sum_{j=0}^{n} x_{hj}^{k} = 0 \qquad \forall h = 1 \dots n, \forall k = 1 \dots m_1 \quad (4.33)$$

$$\sum_{i=0}^{n} y_{ih}^{k} - \sum_{j=0}^{n} y_{hj}^{k} = 0 \qquad \forall h = 1 \dots n, \forall k = 1 \dots m_2 \quad (4.34)$$

$$z_{k}^{p} \geq x_{ij}^{k} \qquad \begin{array}{r} \forall i = 0 \dots n, \forall j = 0 \dots n, \\ \forall k = 1 \dots m_1 \end{array} \quad (4.35)$$

$$z_{k}^{d} \geq y_{ij}^{k} \qquad \begin{array}{r} \forall i = 0 \dots n, \forall j = 0 \dots n, \\ \forall k = 1 \dots m_2 \end{array} \quad (4.36)$$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} q_i x_{ij}^{k} \leq C \qquad \forall k = 1 \dots m_1 \quad (4.37)$$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} q_i y_{ij}^{k} \leq C \qquad \forall k = 1 \dots m_2 \quad (4.38)$$

$$a_{j}^{k} \geq r_j \qquad \forall j = 0 \dots n, \forall k = 1 \dots m_1 \quad (4.39)$$

$$a_{j}^{k} \geq c_{0j} - M(1 - x_{0j}^{k}) \qquad \forall j = 0 \dots n, \forall k = 1 \dots m_1 \quad (4.40)$$

$$\begin{array}{l} a_{j}^{k} \geq a_{i}^{k} + L q_i + c_{ij} \\ \quad - M(1 - x_{ij}^{k}) \end{array} \qquad \begin{array}{r} \forall i = 1 \dots n, \forall j = 0 \dots n, \\ \forall k = 1 \dots m_1 \end{array} \quad (4.41)$$

$$u_k = a_{0}^{k} + L \sum_{i=0}^{n} \sum_{j=0}^{n} q_i x_{ij}^{k} \qquad \forall k = 1 \dots m_1 \quad (4.42)$$

$$v_i \geq u_k - M(1 - \sum_{j=0}^{n} x_{ij}^k) \qquad \forall i = 1 \ldots n, \forall k = 1 \ldots m_1 \quad (4.43)$$

$$p_k \geq v_h + T + L \sum_{i=1}^{n} \sum_{j=1}^{n} q_i y_{ij}^k - M(1 - \sum_{j=0}^{n} y_{hj}^k) \qquad \forall h = 1 \ldots n, \forall k = 1 \ldots m_2 \quad (4.44)$$

$$b_j^k \geq p_k + d_{0j} + Lq_j - M(1 - y_{0j}^k) \qquad \forall j = 0 \ldots n, \forall k = 1 \ldots m_2 \quad (4.45)$$

$$b_j^k \geq b_i^k + d_{ij} + Lq_j - M(1 - y_{ij}^k) \qquad \forall i = 1 \ldots n, \forall j = 0 \ldots n, \forall k = 1 \ldots m_2 \quad (4.46)$$

$$t_i \geq b_i^k - s_i - M(1 - \sum_{j=0}^{n} y_{ij}^k) \qquad \forall i = 1 \ldots n, \forall k = 1 \ldots m_2 \quad (4.47)$$

$$v_i \geq 0, t_i \geq 0 \qquad \forall i = 1 \ldots n \quad (4.48)$$

$$u_k \geq 0 \qquad \forall k = 1 \ldots m_1 \quad (4.49)$$

$$p_k \geq 0 \qquad \forall k = 1 \ldots m_2 \quad (4.50)$$

$$a_i^k \geq 0 \qquad \forall i = 0 \ldots n, \forall k = 1 \ldots m_1 \quad (4.51)$$

$$b_i^k \geq 0 \qquad \forall i = 0 \ldots n, \forall k = 1 \ldots m_2 \quad (4.52)$$

The objective is to minimize the weighted sum of the total travel time, the total tardiness and the number of required trucks. Constraints (4.27) and (4.28) ensure that every pick-up truck starts and ends at the cross-dock. Similarly, constraints (4.29) and (4.30) ensure that all delivery trucks start and end at the cross-dock. Constraints (4.31) express that every order is picked up by one of the pick-up trucks and constraints (4.32) that all orders are delivered by a delivery truck. The flow conservation constraints are expressed by constraints (4.33) and (4.34): if a truck visits a node to pick-up or deliver an order, this truck should travel to a next node (to pick-up or deliver another order) or to the cross-dock. Constraints (4.35) define the correct relationship between the x_{ij}^k and z_k^p variables. Similarly, the correct relationship between the y_{ij}^k and z_k^d

variables is expressed by constraints (4.36). Constraints (4.37) and (4.38) ensure that the vehicle capacity of the pick-up and delivery trucks is not exceeded. The arrival time of a pick-up truck k at the origin of order j is given by c_{0j} if this order is the first one to be picked up (if $x^k_{0j} = 1$), or by $a^k_i + Lq_i + c_{ij}$ if the truck picked up order i before j (if $x^k_{ij} = 1$). The time at which a pick-up truck can start loading an order is then given by the maximum of the release time of the order and the arrival time of the truck at the orders origin:

$$a^k_j = \max(r_j, c_{0j}x^k_{0j}, \max_{\substack{i=1\ldots n,\\ x^k_{ij}=1}} a^k_i + Lq_i + c_{ij}) \tag{4.53}$$

This is expressed by constraints (4.39)–(4.41). As it is assumed that the unloading is sequential and there is always personnel and equipment available, the unload time of each pick-up truck is equal to its arrival time at the cross-dock (given by the time at which truck k can start loading the dummy order) plus the time required to unload all products:

$$u_k = a^k_0 + L\sum_{i=0}^{n}\sum_{j=0}^{n} q_i x^k_{ij} \tag{4.54}$$

This is expressed by constraints (4.42). In these expressions, $L\sum_{i=0}^{n}\sum_{j=0}^{n} q_i x^k_{ij}$ denotes the total unload time of the pick-up truck. Constraints (4.43) then determine the unload times v_i of the orders. As the sequence in which goods are loaded or unloaded is not taken into account, the unload time of an order is set to the unload time of its pick-up truck. Note that in this expression $\sum_{j=0}^{n} x^k_{ij}$ is 1 if order i is picked up by truck k. The ready times of the delivery trucks are defined by constraints (4.44). These trucks can only start their tour when they are completely loaded. As it is assumed that the loading of a delivery truck can only start when all goods are available at the cross-dock, the ready time of a delivery truck is equal to the latest unload time of an order to be delivered by that truck augmented with the time to transfer one product unit and the time needed to load the truck:

$$p_k = \max_{\substack{h=1\ldots n,\\ f^k_h=1}} v_h + T + L\sum_{i=0}^{n}\sum_{j=0}^{n} q_i y^k_{ij} \tag{4.55}$$

In these expressions, $L\sum_{i=0}^{n}\sum_{j=0}^{n} q_i y^k_{ij}$ denotes the total load time of the delivery truck and $f^k_h = \sum_{j=0}^{n} y^k_{hj}$ is 1 if truck k has to deliver order h. The time at which a delivery truck k ends unloading at the destination of order j is

given by $p_k + d_{0j} + Lq_j$ if this order is the first one to be delivered (if $y_{0j}^k = 1$), or by $b_i^k + Lq_j + d_{ij}$ if the truck picked up order i before (if $y_{ij}^k = 1$):

$$b_j^k = \max((p_k + d_{0j} + Lq_j)y_{0j}^k, \max_{\substack{i=1\ldots n,\\ y_{ij}^k=1}} b_i^k + Lq_j + d_{ij}) \tag{4.56}$$

This is expressed by constraints (4.45) and (4.46). Constraints (4.47) and (4.48) determine the tardiness of each order, which is defined as:

$$t_i = \max(e_i - s_i, 0) \tag{4.57}$$

in which e_i is the time at which order i is unloaded at its destination. This time is equal to the variable b_i^k of the delivery truck k of order i (if $f_i^k = \sum_{j=0}^{n} y_{ij}^k$ is 1):

$$e_i = b_i^k f_i^k \tag{4.58}$$

All continuous decision variables have to greater than or equal to zero, which is expressed by constraints (4.48)–(4.52).

The problem formulation is quite large. For n orders and m trucks (m_1 pick-up trucks and m_2 delivery trucks), the total number of variables is given by:

$$\begin{aligned} &m_1 + m_2 + n + (n+1)m_1 + (n+1)m_2 + n + (n+1)^2 m_1 + (n+1)^2 m_2 + m_1 + m_2 \\ &\quad = 2n + 4m + 3nm + n^2 m \end{aligned}$$

The total number of constraints is:

$$\begin{aligned} &2m_1 + 2m_2 + 2n + nm_1 + nm_2 + (n+1)^2 m_1 + (n+1)^2 m_2 + m_1 + m_2 \\ &\quad + 2(n+1)m_1 + n(n+1)m_1 + m_1 + nm_1 + nm_2 + (n+1)m_2 + n(n+1)m_2 + nm_2 \\ &\quad = 2n + 5m + 2m_1 + 7nm + 2n^2 m \end{aligned}$$

For instance, a moderate problem with 20 orders, 5 pick-up trucks and 5 delivery trucks requires a MIP model with 4680 variables and 9500 constraints. In order to avoid long computation times (as reported in Section 4.2.3), a heuristic method is proposed to solve it in a reasonable amount of time. The next section describes the applied solution approach.

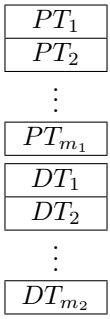


Figure 4.7: Solution representation for the vehicle routing problem with cross-docking.

4.2.2 Simulated annealing approach

To solve the VRPCD problem, a metaheuristic approach is applied. As experiments indicated that simulated annealing (SA) [98, 173] obtains better results than tabu search[8], a simulated annealing approach is used here. The next paragraphs explain the details of the applied simulated annealing.

Solution representation A solution of this vehicle routing problem with cross-docking can be represented by a collection of tours of the pick-up trucks and a collection of tours of the delivery trucks (see Figure 4.7). There are m_1 pick-up tours (PT) and m_2 delivery tours (DT). Each tour consists of a sequence of orders, representing the order in which the goods are picked up or delivered. An example solution is shown in Figure 4.8. The first two rows correspond to the pick-up tours, the second two rows to the delivery tours. For instance, the second pick-up truck visits the origins of 6 orders in the sequence 2–4–6–8–9–10. Similarly, the first delivery truck delivers 4 orders, in the sequence 1–3–5–7. These tours determine completely the solution of the VRPCD, as the pick-up, arrival, departure and delivery times of the orders can be calculated by equations (4.53) to (4.56). Algorithm 4.2 describes the calculation of the objective value of a solution.

Initial solution For the initial solution, the tours for pick-up and delivery are the same. To construct these tours, the orders are first sorted by their capacity requirements and the number of needed trucks based on these requirements

[8]Experimentation with a limited set of parameters indicated that the simulated annealing approach resulted in better objective values.

1	3	5	7		
2	4	6	8	9	10

1	3	5	7		
2	4	6	8	9	10

Figure 4.8: Example solution representation for the vehicle routing problem with cross-docking ($n = 10$ and $m_1 = m_2 = 2$).

is determined. Then, starting from the order with the largest capacity requirements, the orders are added one by one to the tour of the next truck that still has enough capacity. If an order is assigned to the last truck, the next order is again added to the tour of the first truck. If it is not possible to assign all order to the trucks in this way, the same procedure is applied, but now with one extra truck available. For instance, if the sorting of the orders results in the

Algorithm 4.2 Pseudocode of the algorithm to determine the objective value of a vehicle routing solution.

function CALCULATEOBJECTIVEVALUE(PT, DT)
  $travel \leftarrow 0 \quad tard \leftarrow 0$ ▷ Keep track of the total travel time, total tardiness and number of trucks.
  $nbr \leftarrow 0$
  for $k \leftarrow 1 \dots m_1$ **do** ▷ Iterate over the pick-up trucks and determine for each truck k its total load q.
    $t \leftarrow 0 \quad i \leftarrow 0 \quad q \leftarrow 0$
    $PO \leftarrow PT_k$
    for $h \leftarrow 1 \dots \text{size}(PO)$ **do**
      $j \leftarrow PO_h$ ▷ Calculate for each order j its pick-up time a_j.
      $a_j \leftarrow \max(t + c_{ij}, r_j)$
      $travel \leftarrow travel + c_{ij}$
      $t \leftarrow a_j + Lq_j$
      $q \leftarrow q + q_j$
      $i \leftarrow j$
    end for
    $u_k \leftarrow t + c_{j0} + Lq$ ▷ The unload time u_k of truck k is equal to its arrival time at the cross-dock plus the time to unload.
    $travel \leftarrow travel + c_{j0}$
    for $h \leftarrow 1 \dots size(PO)$ **do**
      $j \leftarrow PO_h \quad v_j \leftarrow u_k$ ▷ The unload time v_j of each order j is equal to the unload time of truck k.
    end for
    if $\text{size}(PO) > 0$ **then**
      $nbr \leftarrow nbr + 1$
    end if
  end for

Algorithm 4.2 (cont.) Pseudocode of the algorithm to determine the objective value of a vehicle routing solution.

```
    for k ← 1 ... m_2 do                    ▷ Iterate over the delivery trucks and de-
        p_k ← 0   q ← 0                       termine for each truck k its ready time
        DO ← DT_k                             p_k and total load q.
        for h ← 1 ... size(DO) do
            j ← DO_h
            p_k ← max(p_k, v_j + T)
            q ← q + q_j
        end for
        p_k ← p_k + Lq
        t ← p_k   i ← 0
        for h ← 1 ... size(DO) do
            j ← DO_h                        ▷ Calculate for each order j its delivery
            t ← t + d_ij + Lq_j               time b_j and tardiness t_j.
            travel ← travel + d_ij
            b_j ← t
            t_j ← max(b_j − s_j, 0)
            tard ← tard + t_j
            i ← j
        end for
        travel ← travel + d_j0
        if size(DO) > 0 then
            nbr ← nbr + 1
        end if
    end for
    return w_1 travel + w_2 tard + w_3 nbr
end function
```

sequence from 1 to 10, there are two trucks needed to transport all orders and the capacity limit of the first truck is reached when four orders are assigned to it, the solution shown in Figure 4.8 would be the initial solution.

Neighborhood A composite neighborhood structure is used, consisting of two neighborhoods based on the following two moves:

- Swap move: two orders are interchanged, i.e. the first order is assigned to the tour and the position in the tour at which the second order was assigned, and vice versa (see Figure 4.9). If both orders were assigned to the same truck, only their position in the tour has been changed.

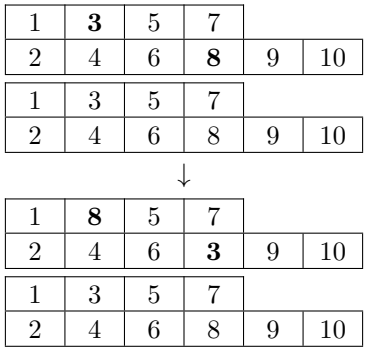


Figure 4.9: Example of a swap move for the vehicle routing problem with cross-docking: the tours of pick-up trucks 1 and 2 changed by interchanging orders 3 and 8.

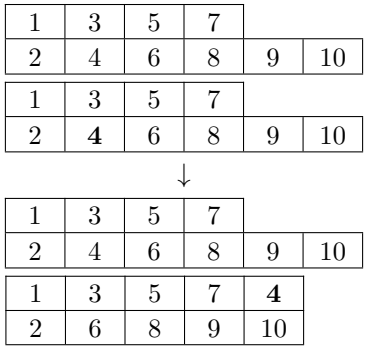


Figure 4.10: Example of an insert move for the vehicle routing problem with cross-docking: the tours of delivery trucks 1 and 2 changed by inserting order 4 at the last position of the tour of delivery truck 1.

- Insert move: an order is assigned to another truck (see Figure 4.10). The order is added to the last position of the tour of its new truck. It is possible that this truck is currently empty.

Note that swap moves do not change the number of orders assigned to the same truck. Insert moves however allow that the orders are redistributed over the pick-up and delivery trucks. Both moves have to be applied to either the pick-up part or the delivery part of the solution.

Acceptance probability In each iteration of the simulated annealing procedure, one random neighbor of the complete (composite) neighborhood is evaluated. The probability P to accept this move depends on the objective value v' of the considered neighbor, the current objective value v and the 'temperature' T_i, and is calculated as follows:

$$P = e^{-\frac{v'-v}{T_i}} \tag{4.59}$$

Cooling schedule An exponential cooling schedule is used, in which the temperature in each iteration is lowered by a constant factor α $(0 < \alpha < 1)$. Starting from an initial temperature T_0, the temperature at iteration i is then given by:

$$T_i = T_0 \alpha^i \tag{4.60}$$

Termination criterion The local search is terminated when the algorithm was not able to improve the current best solution during i consecutive iterations or when the maximum allowed calculation time c has elapsed.

The proposed simulated annealing is also implemented based on the LORA framework. To test whether this approach is able to find acceptable solutions, the solutions found by the simulated annealing are compared with the exact solutions for several test scenarios. The next section describes the results.

4.2.3 Experimental results

The experimental tests carried out to validate the simulated annealing approach are described in this section. First, the set-up of the experiments is described and the various test scenarios are defined. A comparison between the solutions generated by the SA approach and the exact solutions (found by solving the MIP model with CPLEX©) is presented next and the results are discussed.

Experimental set-up

The following factors influence the 'hardness' of the problem instances.

1. The problem size, i.e. the number of orders and trucks.

2. The ratio of the number of orders to the number of trucks[9]. If this ratio is low, the problem is almost reduced to the assignment of orders to trucks. The problem is more complicated in case of a high ratio, as the batching of orders and the routing of the trucks also have to be taken into account.
3. The distribution of the orders, i.e. the geographical distribution of the origins and destinations of the orders. If multiple orders have the same origin or destination, it is often efficient to serve these orders with the same truck[10], lowering the complexity of the problem.
4. The simultaneity of the due dates. The trade-off between the truck costs and the tardiness costs becomes more difficult if the due dates are in a short period of time.
5. The tightness of the time windows of the orders (formed by the release times and due dates).

For the experiments, it is assumed there are three pick-up trucks and three delivery trucks available. The other fixed parameters are shown in Table 4.3. The number of items of which an order consists is randomly sampled while making sure that the problem is feasible. The aforementioned factors have been reflected during the generation of different problem instances by varying three parameters:

Number of orders By changing the number of orders while the number of trucks is fixed, variations can be obtained in both factors 1 and 2. This parameter can have one of the following three values:

- Low: there are 4 orders.
- Medium: there are 5 orders.
- High: there are 6 orders.

Number of nodes This parameter determines the number of different nodes (relative to the number of orders), from which the origins and destinations of the orders are chosen randomly (factor 3). There are three values:

- Low: the number of nodes is equal to 2 times the number of orders.
- Medium: the number of nodes is equal to the number of orders.
- High: the number of nodes is equal to half of the number of orders.

Time window This parameter determines how the release times and due dates of the orders are generated. The release times and due dates are sampled in a certain time interval. The length t of this time interval is directly proportional to the number of orders: $t = a * nbr$, in which nbr is the

[9]This factor is influenced by the different costs of the objective function. If the truck cost is high compared to the other costs, the optimal solution tends to make use of few trucks, and so the 'real' ratio of the number of orders on the number of trucks tends to be high.

[10]This will also depend on the release times or due dates of these orders.

Table 4.3: Fixed parameter values for the vehicle routing problem instances.

Parameter	Value	
C	33	
L	2	
T	3	
w_1	1	
w_2	5	
w_3	1000	
T_0	500	
α	0.9999	
i	10000	
c	5	s

number of orders. The release times of the orders are sampled from the interval $[0, t]$ and the due dates from the interval $[e, e + t]$ in which $e = r + b * (n * TT + CL)$. In this expression, r is the release time of the order, n the average number of nodes that have to be visited by a pick-up truck, TT the average travel time between the nodes and CL the time to completely load or unload a truck. The values of a and b are determined by one of the three values of this parameter:

- Low: $a = 30$ and $b = 2$.
- Medium: $a = 20$ and $b = 1.5$.
- High: $a = 10$ and $b = 1$.

Each combination of values of these parameters (27 in total) denotes a problem type. For each problem type 10 instances were generated.

Experimental results

All 270 problem instances have been solved with the proposed simulated annealing approach and with the CPLEX© solver[11]. All experiments were run on a PC with two 2.40 GHz hex-core processors and with 96 GB RAM. As it takes CPLEX© a long time to find the optimal solution for the larger problems, an upper bound of two hours (7200 s) is applied. If the optimal solution is not found within this time limit, the current best solution generated by the solver is reported. Table 4.4 summarizes the results. The first column shows the problem types, indicated by the value (L = low, M = medium, H = high)

[11]IBM ILOG CPLEX Optimization Studio 12.5.

for its three parameters: number of orders, number of nodes and time window. For each problem type, the average objective value over its 10 instances is calculated. These values are shown in the next two columns for the SA and the MIP approach. The lowest average values for all problem types are indicated in bold. The fourth column indicates how many of the 10 problem instances have been proven optimal by CPLEX©. In order to compare the simulated annealing and the CPLEX© approach, the best of both approaches is determined for each instance. Columns 5 and 6 then show how many times simulated annealing or CPLEX© approach was the winning approach (in case of a draw, both are counted). The solution of the simulated annealing approach is then compared with this best solution. If SA_i is the solution of problem instance i found by simulated annealing and C_i the solution found by CPLEX©, the average relative deviation (in percentage) is calculated as follows:

$$100 \left(\frac{SA_i - \min(SA_i, C_i)}{\min(SA_i, C_i)} \right)$$

The average value for each problem type is shown in the next column. Finally, the average calculation times of the two approaches are shown in the last two columns.

Table 4.4 shows that simulated annealing is able to find good results in a short time period. For the problems with the lower parameter values, simulated annealing was almost always able to find the optimal solution. For all problems, the average relative deviation compared to the best solution found is lower than 0.4 %. When looking at the results of the individual problem instances, the largest observed relative deviation is lower than 3.5 %. CPLEX© was not able to find the optimal solution for all problem instances within the time limit of two hours. For 21 of the 270 problem instances, the optimal solution was not found. Simulated annealing found the same solution for 10 out of these 21 instances. In total, the simulated annealing approach found a solution as good as the CPLEX© result for 253 problem instances. Simulated annealing is also very efficient. The solutions of all problem instances were found in less than 0.1 s.

4.2.4 Rescheduling

It is possible to extend the proposed simulated annealing approach to be able to reschedule. In order to reschedule, the current situation has to be taken into account. For the considered VRPCD problem, this means that the positions of the trucks should be taken into account, and which orders are loaded or

Table 4.4: The experimental results for the 27 vehicle routing problem types (L = low, M = medium, H = high).

Problem type			Objective value		# MIP optimal	# SA best	# MIP best	Relative dev. (%)	Calculation time (s)	
			SA	MIP					SA	MIP
L	L	L	**5545.086**	**5545.086**	10	10	10	0.000	0.013	0.495
L	L	M	**6119.425**	**6119.425**	10	10	10	0.000	0.011	1.714
L	L	H	**7185.494**	**7185.494**	10	10	10	0.000	0.015	3.830
L	M	L	**4354.746**	**4354.746**	10	10	10	0.000	0.014	1.517
L	M	M	**6648.547**	**6648.547**	10	10	10	0.000	0.013	1.117
L	M	H	**9145.090**	**9145.090**	10	10	10	0.000	0.014	2.672
L	H	L	**5476.295**	**5476.295**	10	10	10	0.000	0.013	1.103
L	H	M	**7067.770**	**7067.770**	10	10	10	0.000	0.014	2.649
L	H	H	**7599.146**	**7599.146**	10	10	10	0.000	0.013	4.631
M	L	L	5059.405	**5058.023**	10	9	10	0.030	0.014	5.592
M	L	M	**5116.053**	**5116.053**	10	10	10	0.000	0.016	30.845
M	L	H	**8095.198**	**8095.198**	10	10	10	0.000	0.019	151.982
M	M	L	**5084.396**	**5084.396**	10	10	10	0.000	0.016	13.525
M	M	M	**5916.224**	**5916.224**	10	10	10	0.000	0.017	75.529
M	M	H	**8316.800**	**8316.800**	10	10	10	0.000	0.014	143.637
M	H	L	**5027.814**	**5027.814**	10	10	10	0.000	0.016	13.645
M	H	M	6420.112	**6415.873**	10	9	10	0.113	0.016	54.684
M	H	H	**8828.485**	**8828.485**	10	10	10	0.000	0.017	232.146
H	L	L	4863.596	**4856.076**	10	6	10	0.161	0.024	19.930
H	L	M	5630.384	**5609.614**	10	9	10	0.344	0.025	858.951
H	L	H	**8856.727**	**8856.727**	3	9	10	0.000	0.020	5370.240
H	M	L	**4746.094**	**4746.094**	10	10	10	0.000	0.021	126.188
H	M	M	5834.951	**5824.388**	9	8	10	0.215	0.019	2184.842
H	M	H	**8743.637**	**8743.637**	7	9	10	0.000	0.022	3398.164
H	H	L	5417.438	**5415.721**	10	9	10	0.038	0.019	544.975
H	H	M	6330.787	**6326.820**	8	7	10	0.095	0.019	3211.772
H	H	H	9025.867	**9017.867**	2	8	10	0.095	0.021	6193.772

already unloaded at their destinations. Concretely, the following information is required:

- the current time;
- the current positions of the trucks;
- if orders are already loaded in a truck, the truck in which they are loaded;
- if orders are already unloaded from a delivery truck, the truck from which they are unloaded;
- if a pick-up truck arrived at the cross-dock, its arrival time;
- if a delivery truck departed from the cross-dock, its departure time.

The following (extra) assumptions are made:

- If a pick-up truck has started its tour, extra orders can still be assigned, as long as it is not traveling to the cross-dock.
- No new (not loaded) orders can be assigned to a delivery truck, once it has departed from the cross-dock.
- The loading of a delivery truck can start (continue) when all (remaining) goods for that truck are available at the cross-dock.
- If there are not enough trucks available to pick up or deliver all orders, some orders will not be considered.

The solution representation is as before, but now the tours are divided in two parts (see Figure 4.11). The first part of the tours indicates the already executed part (dark grey). For the pick-up tours, the first part corresponds to the orders that are already loaded. The second part contains the orders that still have to be picked up. For instance, for the solution in Figure 4.11, the first pick-up truck has already picked up order 2 and still has to pick up orders 4, 8 and 9. The second pick-up truck has already picked up all remaining orders. For the delivery tours, the first part contains the orders that are already delivered at their destination (dark grey). The second part corresponds to the orders that still have to be delivered. Two types of orders can be distinguished in this second part: orders that are already loaded into the delivery truck (light gray) and orders that still have to be loaded. For the solution in Figure 4.11, orders 1 and 3 are already delivered at their destination by delivery truck 1, that also will deliver the already loaded orders 5 and 7. Delivery truck 2 has already two orders loaded, orders 6 and 10. The other orders still have to be loaded. Note that if the truck has not yet departed from the cross-dock, the first part is empty. If the truck has departed, all orders in the second part are loaded orders (as delivery truck 1 in Figure 4.11). The sequence of the orders in the first part of both the pick-up tours and delivery tours is not important anymore, as this is the already executed part of the solution and does not influence the objective value of the solution.

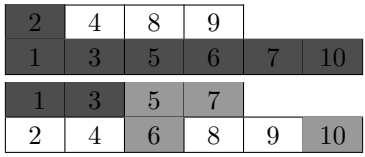


Figure 4.11: Example solution representation for the vehicle routing problem with cross-docking in case of rescheduling.

Three adaptations of the simulated annealing approach are then required. First, the calculation of the objective value should take the current situation into account. For the total travel time, the current position of the trucks should be taken into account, and not their initial position (at the cross-dock). Also, only the travel times of the not yet executed part of the tours should be be considered (part 2 of the tours). To calculate the delivery times of the orders, and hence the tardiness, the current time should be taken into account. Also, only the not yet executed part of the tour should be considered to determine the the pick-up, arrival, departure and delivery times of the orders. Algorithm 4.3 describes the adapted calculation of a solution's objective value if the current time, positions of the pick-up (PP) and delivery trucks (DP), arrival times of the pick-up trucks (PA) and departure times of the delivery trucks (DD) are known.

Second, the initial solution should correspond to the current situation. For the pick-up tours, this means that orders that are already loaded should be placed in the first part of the correct tour. Orders that still have to be picked up should be in the second part of a tour. For the delivery tours, the orders that are already delivered have to be placed in the first part and the orders that are loaded in the second part of the correct tour. The other orders should be in the second part of a delivery tour. If there are not enough trucks available to pick up or deliver all goods (for instance because all pick-up trucks are heading towards the cross-dock and some orders are not loaded yet), these orders will not be taken up in the initial solution.

A third adaptation is that not all moves are allowed anymore. The first part of the pick-up and delivery tours corresponds to orders that are already picked up or delivered, so this part of the solution cannot be changed. All moves that would change the first part of a tour are prohibited. For the pick-up tours, the orders of the second part can be mutually swapped (both within a tour and between tours) and they can be inserted at the tour of another truck, as long as that truck is not traveling to the cross-dock. Other moves are not allowed. For the delivery trucks, the orders of a truck that are not already delivered (part 2) can be swapped within that tour. The orders of the second part that are not

already loaded can also be swapped with orders of other tours. These orders can also be inserted at another truck's tour, if the truck has not departed from the cross-dock. Other moves are prohibited.

Algorithm 4.3 Pseudocode of the algorithm to determine the objective value of a vehicle routing solution in case of rescheduling.

```
function CALCULATEOBJECTIVEVALUE(PT, DT, time, PP, DP, PA, DD)
    travel ← 0   tard ← 0                    ▷ Keep track of the total travel time,
    nbr ← 0                                  total tardiness and number of
    for k ← 1 ... m_1 do                     trucks.
        q ← 0                                ▷ Iterate over the pick-up trucks and
        for h ← 1 ... size(PO) do            determine for each truck k its total
            j ← PO_h   q ← q + q_j           load q.
        end for
        if ARRIVED(k) then                   ▷ Determine the arrival time t of each
            t ← PA_k                         truck k.
        else
            t ← time   i ← PP_k
            PO ← PT_k
            for h ← 1 ... size(PO) do
                j ← PO_h
                if !PICKEDUP(j) then
                    a_j ← max(t + c_ij, r_j)  ▷ Calculate for each order j that is
                    travel ← travel + c_ij    not yet picked up its pick-up time
                    t ← a_j + Lq_j            a_j.
                    i ← j
                end if
            end for
            t ← t + c_j0
            travel ← travel + c_j0
        end if                               ▷ The unload time u_k of truck k is
        u_k ← t + Lq                         equal to its arrival time at the
        for h ← 1 ... size(PO) do            cross-dock plus the time to unload.
            j ← PO_h   v_j ← u_k             ▷ The unload time v_j of each order j
        end for                              is equal to the unload time of truck
        if size(PO) > 0 then                 k.
            nbr ← nbr + 1
        end if
    end for
```

Algorithm 4.3 (cont.) Pseudocode of the algorithm to determine the objective value of a vehicle routing solution in case of rescheduling.

```
    for k ← 1 ... m_2 do                          ▷ Iterate over the delivery trucks.
        DO ← DT_k
        if DEPARTED(k) then                       ▷ Determine the departure time t of
            t ← DD_k                                each truck k.
        else
            p_k ← time    q ← 0                   ▷ If truck k has not departed, deter-
            for h ← 1 ... size(DO) do               mine its ready time p_k.
                j ← DO_h
                if !LOADED(j) then
                    p_k ← max(p_k, v_j + T)
                    q ← q + q_j                   ▷ Determine the number of product
                end if                              units that have to be loaded (q).
            end for
            p_k ← p_k + Lq
            t ← p_k
        end if
        t ← max(t, time)    i ← DP_k
        for h ← 1 ... size(DO) do
            j ← DO_h
            if !DELIVERED(j) then
                t ← t + d_ij + Lq_j               ▷ Calculate for each order j that is
                travel ← travel + d_ij              not yet delivered its delivery time
                b_j ← t                             b_j and tardiness t_j.
                t_j ← max(b_j − s_j, 0)
                tard ← tard + t_j
                i ← j
            end if
        end for
        travel ← travel + d_j0
        if size(DO) > 0 then
            nbr ← nbr + 1
        end if
    end for
    return w_1 travel + w_2 tard + w_3 nbr
end function
```

This rescheduling approach takes several aspects of the current situation into account, but it is not easy to consider all relevant details. For instance, a drawback of the current approach is that no new transport operations can

be assigned to a truck that has completed its tour. This also signifies that newly available orders or orders not assigned to a truck cannot be included in the updated vehicle routing schedule once all trucks have departed from the cross-dock.

4.3 Conclusions

This chapter elaborates on two cross-dock related problems introduced in Chapter 3: the truck scheduling problem and the vehicle routing problem with cross-docking. For both problems, the real-world situation is as realistic as possible taken into account. As this includes extra aspects compared to similar problems considered in the literature, both problems are described in detail and mathematical models are provided to define them formally. Also, a heuristic approach is provided for the two problems in order to solve these problems in a reasonable amount of time. The experimental results confirm that both heuristics are able to find good results in a short time period.

Although both problems consider many realistic aspects, they still provide a simplified and approximated view of the real-world problem. Accounting for all relevant details is in many scheduling problems very difficult or even impossible and can make the scheduling intractable. For instance, the VRPCD problem does not consider the internal operations inside the cross-dock and the availability of the internal resources (e.g. forklifts). As indicated in Section 4.1.4, taking extra aspects into account is not always obvious and for instance a change in objective function can invalidate the proposed heuristic approach. Moreover, in a dynamic environment, operations deviate from schedule within the proverbial first minutes [146, 215]. So, there needs to be a system in place to execute the (static) schedules. The HLES can perform this schedule execution, while accounting for deviations and omitted details. On the other hand, the scheduling algorithms can support the HLES implementation in finding good global solutions. The cooperation between the presented scheduling algorithms and the Holonic Logistics Execution System is presented in Chapter 5.

For the vehicle routing problem with cross-docking, some adaptations to the simulated annealing approach are proposed in order to enable rescheduling. In general, rescheduling (or predictive–reactive scheduling, see Section 5.4.3) needs to address two main issues: when to react to real-time events and how to revise the existing schedule [12, 144]. Regarding the first issue, new schedules can be generated at regular intervals (periodic policy) or in response to an unforeseen event (event-driven policy). A combination of both policies is also possible (hybrid policy). The thesis does not elaborate on this issue, but a

good (application-specific) policy should be determined for a specific HLES implementation. Regarding the second issue, some local adjustments can be made to the current schedule (schedule repair), or a new schedule can be generated from scratch (complete rescheduling). The proposed rescheduling approach belongs to the second category and produces a new schedule while accounting for the current situation (e.g. truck positions). As this can result in instability and lack of continuity, extra mechanisms can be required in order to handle this nervousness. This thesis will also not deal in more detail with this issue.

Chapter 5

HLES for cross-docking

Chapter 2 described the basic principles of the Holonic Logistics Execution System (HLES). In Chapter 3, cross-docking was introduced as an emerging logistic strategy. This chapter explains how an HLES implementation to coordinate and control the cross-docking operations can be developed. First, the adaptations and extensions to the 'core' of the HLES (which provides the basic functionality) are described. Section 5.1 explains how the HLES offers support for mobile resources as trucks and trains. In Section 5.2, it is discussed how various orders can be processed together (batching). Section 5.3 explains how an order can plan the simultaneous use of multiple resources (multi-resource allocation). Second, the cross-docking HLES can be supported in finding good global solutions by scheduling advice. In Chapter 4, scheduling approaches for two cross-dock related problems were presented. Section 5.4 then describes how the HLES can cooperate with two scheduling systems based on these approaches. Next, during the development of an HLES implementation, domain-specific models and application-specific decision mechanisms are required for all relevant entities in the world-of-interest (products, resources and orders). The main aspects of these models and decision mechanisms are discussed in Section 5.5. Finally, Section 5.6 presents the conclusions of this chapter.

5.1 Mobile resources

In logistic applications, many resources do not have a fixed position in contrast to manufacturing resources (e.g. machines). Trucks and trains are examples of such mobile resources. Before these resources can perform an operation (e.g.

loading or unloading of freight), they have to move to the correct position. This section explains how the HLES offers support to include these operations for the mobile resources.

One of the responsibilities of a resource holon is 'maintaining a local schedule' (see Section 2.2.3). This local schedule keeps track of the future (reserved) operations of the resource. In previous HMES implementations, this local schedule only contained reservations of order holons. For the mobile resources of the HLES, the resource holon itself is now also able to reserve time slots for auxiliary drive operations (in which no orders are involved). The holon is then also responsible for ensuring that these extra operations are executed by the real-world resource.

The resource agent, responsible for the decision making aspects of the resource holon (i.e. 'managing its local schedule'), makes use of this extra ability of the local schedule. When an order holon requests information about the virtual outcome of an operation, the resource agent checks whether an auxiliary operation is needed in order to be able to perform the requested operation. For instance, if an order wants to be transported by a truck, the corresponding order holon will first try to virtually execute a loading operation. The truck agent will then examine the position of the order and determine whether a preceding drive operation (to the order's position) is needed. The resource agent has to decide how the requested operation and the extra drive operation can be fitted in between the already reserved operations and the order holon is informed about the result. If the order holon afterwards effectively makes a reservation for its requested operation, the resource agent will also reserve a time slot for the necessary drive operation.

The resource agent is responsible for keeping its local schedule, including the reservations of auxiliary drive operations, up-to-date. If an operation, for which also an auxiliary operation is reserved, is shifted in time, the time slot for the drive operation is adapted accordingly. When an operation is removed (e.g. by evaporation), the resource agent checks if the corresponding drive operation can also be removed or has to be shifted in front of the next reserved operation (for instance another load operation at the same position).

This mechanism is applied by all resource holons which represent a mobile resource. In the context of cross-docking, this means that the resource holons corresponding to trucks and forklift trucks will reserve extra drive operations if needed. More information about the decision making applied by these resources can be found in Section 5.5.2.

5.2 Batching

Various logistic resources have to be shared between different orders in order to be used efficiently. For instance, trucks, ships and trains have to make sure that they are sufficiently loaded before starting a transport operation. So, several orders need to be grouped or batched for simultaneous execution of a transport operation. This batching needs to be dynamic and determines the quality of the resulting control actions to a great extent. This section discusses how the HLES supports the batching of multiple orders.

As mentioned above, the resource holons maintain a local schedule with the reserved operations of the resource. In order to support batching, reservations now can be made for several orders together, i.e. a time slot for one transport operation can be reserved for multiple orders. Next, the resource holons are responsible for making sure that the corresponding real-world resources start executing these operations when all orders are available and to inform all involved order holons about the start and stop of the operation.

It is again the resource agents which can make use of this extra ability of the local schedule. If an order holon wants to virtually execute an operation, the resource agent has to determine how the requested operation fits in the local schedule. The agent can decide that a new time slot is needed for the operation or can assign the order to an already reserved time slot (with the same operation). For instance, if an order wants to be transported and a truck has already a time slot reserved for a transport operation to the same destination, the truck agent can decide to assign the order to that time slot. Otherwise, the agent will have to reserve a new time slot at a free period. If there is already a time slot reserved with the same operation but at another moment, the resource agent can also try to shift this time slot, but this will influence the intentions of other order holons.

The decision making mechanisms best suited for the various resources are application-specific. For the cross-docking HLES, the resource agents corresponding to trucks will make use of this batching ability. Section 5.5.2 provides more information about the concrete decision making mechanisms applied by these truck agents.

Although the resource holons support batching, it is still the order holons who have to make the reservations which result in a global solution with efficiently used resources. This outcome can be difficult to obtain without extra coordination between the various order holons, certainly if the orders are confronted with a large search space. For instance, in a cross-docking context, every order can typically be transported by (almost) every truck, resulting in a large number of batching opportunities. In the proposed cross-docking HLES,

the order holons receive advice from the staff holon (based on the VRPCD algorithm described in Section 4.2) in order to obtain a good batching. This is explained in more detail in Section 5.4. An alternative approach could be that, once a truck agent has made a reservation for a transport operation, it tries to attract other orders with the same destination, e.g. by spreading 'attractor' pheromones.

5.3 Multi-resource allocation

To perform some logistic operations, multiple resources are required at the same time. For instance, to unload an order from a truck at a cross-dock, three resources need to be available simultaneously: the involved truck, a dock door to which the truck is docked and a forklift truck to perform the unloading operation. In fact, for every operation of a forklift truck, two resources are needed: the forklift truck itself and a forklift driver. This section explains how the HLES can perform this so-called multi-resource allocation.

In order to make use of multiple resources simultaneously, the order holons first need to make the necessary reservations at the involved resource holons. The product holon indicates to its order holons if multiple resources are needed together. Instead of providing one operation as a possibility for the next operation ('selection of operation options', see Section 2.2.3), the product holon will specify a combination of operations. For instance, to unload an order from a truck, three operations need to be performed at the same time: a load operation by a forklift truck, an unload operation by the truck and a 'standby' operation of a dock door. In order to reserve the necessary time slots, the behavior of the exploring and intention ants sent out by the order holons is adapted as follows. If these ants have to (virtually) execute multiple operations simultaneously, they consider one operation as the 'main' operation and the other operations as 'passive' operations. The ants then first try to execute the main operation. This operation determines the duration for which the multiple resources have to be allocated. If this operation succeeds, the ants try to execute the passive operations, with the same start time and duration as the main operation. For instance, if an order needs to be unloaded from a truck, its ants will first visit the forklift truck agent and virtually perform the load operation. This agent determines the duration of the operation (based on its model of the corresponding real-world forklift). Subsequently, the ants try to allocate a time slot of the same duration at the involved truck agent and dock door agent. If all operations succeeded, the ants continue their journey. Otherwise, they will spread resource intentions, similar to the situation without multi-resource allocation.

The resource holon at which the main operation is reserved is then responsible for starting the execution of the real-world operation. This holon not only has to check if its corresponding physical resource can start the execution, but also has to examine the availability of the other involved resources. Once the operation is started, this resource holon also has to inform the other resource holons about the progress of the operation.

In the current implementation of the cross-docking HLES, it is assumed that a distinction can be made between the different operations that have to be executed simultaneously. In the provided combination of operations, there is one main operation which determines the duration of the operation, and the other operations are of secondary importance. This corresponds well to the situation for unloading an order from a truck. The actual operation, which also determines the duration, is performed by a forklift truck, while the other resources are passive resources and only have to be available. It is however needed to allocate these resources, as they are not available for other operations until the operation is finished. In other situations however, the duration (and result) of the operation can be dependent on the interaction between the involved resources. For instance, the duration to (automatically) make a mixed pallet ready for transportation depends on the speed and interaction of a robotic palletizer and a stretch wrapping robot. In this case, making a distinction between main and passive operation simplifies the real-world situation too much. For this kind of situations, it should be possible to determine the required availability of the resources more accurately, for instance by simulating a scenario with all involved resources. Subsequently, the necessary reservations should be made at the corresponding resource holons. Note that, if this simulation indicates that not all resources are needed for the complete duration of the operation, different time slots should be reserved at the involved resources. This approach is not implemented in the HLES prototype.

The proposed mechanism will be applied by the order holons every time the product holon indicates that multiple resources are required simultaneously. For the cross-docking application, multi-resource allocation will be used for the loading and unloading of orders.

5.4 Cooperation with scheduling systems

As the HLES is decentralized (the various holons make use of local information), the control system does not automatically obtain a good global performance. However, as explained in Section 2.2.3, a staff holon can provide the other holons with expert knowledge, for instance from a scheduling algorithm. In this way,

the HLES can benefit from a global view delivered by a centralized scheduling system in order to improve the global performance. This section explains how the cross-docking HLES can cooperate with two scheduling systems based on the approaches described in Chapter 4. In Section 5.4.1, the cooperation between the HLES and a vehicle routing scheduling system (VRSS) is explained. Section 5.4.2 discusses how the HLES can cooperate with a truck scheduling system (TSS). From another point of view, the HLES can be considered as responsible for executing the provided schedule(s). Indeed, only a schedule is not sufficient to control the operations; there also needs to be a system in place to execute this (static) schedule. The HLES performs this 'schedule execution', while accounting for (small) deviations and possible simplifications. Section 5.4.3 elaborates on this issue.

5.4.1 HLES-VRSS cooperation

The cross-docking HLES is also responsible for organizing the transport operations from and to the cross-dock. As explained in Chapter 2, each order agent searches independently for solutions and chooses - according to its own performance measure - the best solution. This does not automatically result in a good global performance. For instance, consider a situation in which multiple order agents all try to reserve a time slot at a truck for the same transport operation. Without coordination between the various order agents, there is no agreement about the start time of this operation and they try to make reservations with different start times. This will probably result in the reservation of multiple time slots (for the same operation) and the involved truck will have to drive multiple times the same trajectory, while maybe one transport operation would have been sufficient. To improve the truck utilization, and consequently the global performance, the cross-docking HLES can benefit from a global view delivered by a centralized scheduling system. More specifically, this section explains how the Holonic Logistics Execution System can cooperate with a vehicle routing scheduling system (VRSS) to obtain a good batching of the orders for transportation. The HLES follows the provided vehicle routing schedule if it is feasible. Otherwise, the autonomous decision making mechanisms of the HLES execute alternatives that resemble the original schedule.

The cooperation between the HLES and the VRSS happens through the staff holon. The staff holon provides the scheduling system with all necessary information to determine a vehicle routing schedule. This schedule only considers the transport operations from and to the cross-dock (the operations for which the orders have to be batched). When the staff holon receives the computed schedule, it passes (the relevant parts of) this schedule to the involved order

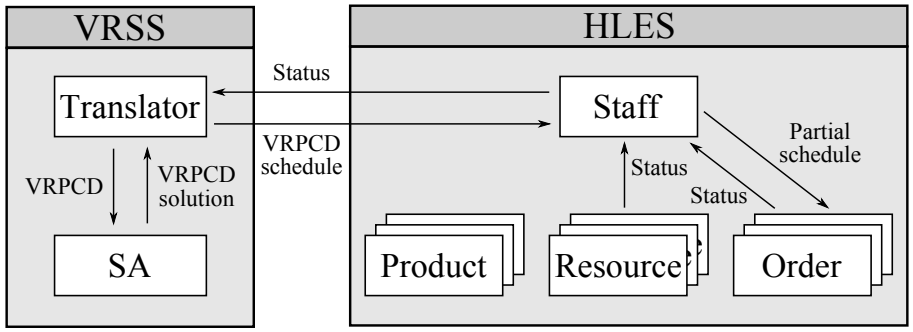


Figure 5.1: Cooperation between HLES and VRSS.

holons. The cooperation scheme between both systems is shown in Figure 5.1 and is discussed in detail in the following paragraphs.

If the staff holon decides to consult the vehicle routing scheduling system, it has to provide the scheduling system with up-to-date status information. The following information is required:

Order information The staff holon needs to know the origin, destination, release time and due date of each order.

Truck information The staff holon also has to find out the number of available pick-up and delivery trucks and the capacity of the trucks. For each truck, its current position and load have to be known. Also, the arrival times at the cross-dock of the pick-up trucks and the departure times from the cross-dock from the delivery trucks are required data.

The staff holon will collect this information by consulting the various order and resource (truck) holons. Next to the status information, the staff holon also has to provide the following data to the VRSS:

- the estimated travel times between the nodes (cross-dock, origins and destinations);
- the (fixed) time required to load or unload one order (L);
- the (fixed) time needed to transfer one order from its strip to its stack door (T);
- the weighting factors for the considered objective function (w_1 for the total travel time, w_2 for the total tardiness and w_3 for the number of trucks).

These data, the travel times excepted, are parameters of the vehicle routing problem with cross-docking that can be chosen by the staff holon. The values of these parameters used in the simulation experiments (see Chapter 6) are shown in Table 6.3. The staff holon can also specify the parameters of the simulated annealing approach applied by the VRSS.

The VRSS uses the information provided by the staff holon as input to construct a VRPCD problem corresponding to the current situation. Therefore, the VRSS has to 'translate' the real-world constraints into solution space constraints and has to put the information into the correct data format. The vehicle routing problem with cross-docking is then solved by applying the simulated annealing approach described in Section 4.2.2. Next, the solution is 'translated' into a vehicle routing schedule usable for the various order holons. To this end, the pick-up and delivery tours with their corresponding arrival and departure times are derived from this solution and put in the correct data format. This translation step can also include postprocessing of the schedule. In the obtained schedule, the pick-up trucks arrive as early as possible at the cross-dock. However, in order to have more simultaneous arrivals of the pick-up trucks, the tours of these trucks will be shifted forward in time, but without delaying the delivery trucks. Subsequently, the resulting vehicle routing schedule is sent to the staff holon which informs the order holons about the part of the schedule relevant for them. This part constitutes a partial solution for the orders, containing only information about the transport operations from and to the cross-dock (start time and which truck should transport the order).

The order holons use this (partial) schedule as a guideline and attempt to execute it. Therefore, their behavior is adapted in two ways [203, 215, 216]. First, each order agent considers 'similarity to the schedule' in its performance measure. While evaluating the solutions found by its exploring ants, order agents prefer solutions that resemble their schedule, i.e. solutions with similar resource allocations and start times. To evaluate the similarity to the schedule, a 'similarity value' SV between 0 and 1 is calculated for each solution. A higher value indicates a higher similarity to the schedule. If the solution does not have the same resource allocation as the schedule, this value is 0. Otherwise, this value for a solution a is calculated as follows:

$$SV(a) = \max\left(1 - \frac{\sum_{i=1}^{n} |s_i^a - s_i|}{e_n - s_1}, 0\right) \tag{5.1}$$

in which

- n is the number of operations in the schedule (numbered from 1 to n);
- s_i is the start time of operation i in the schedule;

- s_i^a is the start time of the operation in solution a corresponding to operation i in the schedule;
- e_i is the end time of operation i in the schedule.

So, the total deviation of the operation start times, relative to the lead time of the schedule, is used to determine how similar a solution and the schedule are. As respecting the resource allocation is more important than respecting the timing, determining the similarity value to the vehicle routing schedule in this way makes sense. Indeed, it is important that the orders are assigned to the trucks as indicated in the schedule, in order to obtain the batching from the schedule. A (small) shift in time will have less influence on the global performance. The concrete selection mechanism employed by the order agents, which makes uses of this similarity measure, is explained in Section 5.5.3.

Secondly, a certain percentage of the exploring ants sent out by the order agents will not randomly search for solutions, but will use the schedule as a guideline during their exploration. The behavior of these exploring ants changes in two ways. First, if they can choose between multiple options regarding the operation to execute next or on which resource to execute an operation, they will give priority to an option in accordance with the schedule. Note that the schedule only specifies the transport operations. If there are multiple options for the other operations (e.g. loading or unloading by a forklift truck), the exploring ant can choose one of those options randomly, similarly to the situation without schedule. Secondly, they will try to postpone the operation start times in order to match the corresponding start times indicated by the schedule. In this way, the order agents will always receive updated solutions following the schedule, as long as such a solution is feasible.

As explained in Section 4.2.4, the simulated annealing approach proposed to solve the VRPCD problem can also be used for rescheduling. The vehicle routing scheduling system will determine a new schedule when it is triggered by the staff holon. The trigger condition of the staff holon is application-specific. The trigger conditions can for instance be specified in the following ways.

- By specifying a time interval. The scheduling system is triggered at regular time intervals.
- By specifying a disruptive event (event type and related parameters). When such an event happens, the VRSS is triggered. Examples are a resource breakdown (for a certain time period), the release of a new order, etc.
- By specifying a measure to compare the schedule with the (forecasted) execution over a certain time horizon. If the obtained value is above a certain threshold, the scheduling system will be triggered.

A combination of several trigger conditions of these types is also possible. In the research prototype, the first trigger condition is implemented. Note however that very frequent schedule updates are not required as the HLES will account autonomously for small deviations. Moreover, too frequent changes make the schedule unusable in practice as well as unpopular with the workforce.

In order to limit the number of schedule updates (and to prevent schedule nervousness), two simple mechanisms are applied in the HLES prototype. Note however that extra or more advanced mechanisms can be required. First, when the VRSS determines a new solution, it makes use of the previously found VRPCD solution. This solution is used as the initial solution for the simulated annealing approach applied by the VRSS. If needed, this solution is first adapted to correspond to the current situation. The newly found solution is then also compared with the previous solution. Only if the new solution has a significantly better objective value (relative improvement of at least α_1 %), the vehicle routing schedule corresponding to this solution is sent to the staff holon. This reduces for instance the probability that a solution with a similar objective value but other resource allocation (e.g. interchange of two trucks) is used. Secondly, when an order holon receives a partial schedule from the staff holon, it compares this updated schedule with the previously received partial schedule (by using equation (5.1)). If the relative difference is only small (smaller than α_2 %), the order holon will not make use of the updated partial schedule.

When the staff holon triggers the VRSS to determine a new vehicle routing schedule, it communicates also information about the current status of the cross-docking system. If the current status of a resource is not completely known, the executable domain model of the corresponding resource holon can be used to determine the expected status. For instance, a truck does not know its exact position when the GPS receiver temporary loses its signal. Based on the last known position and its current transport operation, the truck's expected position can be determined. In a similar way, the scheduling system can be provided with a (short-term) future status. In this way, the scheduling system can react even before an event takes place, so pro-active instead of reactive rescheduling should be possible. This approach can possibly also be used to account for the time delay between capturing the status and receiving an updated schedule (due to communication delays and the calculation time of the VRSS). The time delay does however not compromise the correct functioning of the HLES, as the order holons will deviate from the schedule when needed. This last possibility is not implemented in the HLES prototype.

As mentioned above, when an order agent evaluates the solutions found by its exploring ants, the agent prefers solutions that resemble its schedule. In fact, similarity to schedule is considered as main criterion in the intention selection

process. Only if several solutions have the same similarity value, other criteria are considered (see Section 5.5.3). This approach makes sense as these other criteria are local criteria, and the effect on other orders is not considered. For instance, if flow time is considered as a criterion, the order agents will prefer solutions in which they are transported directly to the cross-dock, without batching. Of course, this would increase the flow time of other orders, which would have to wait for the return of the truck. As a result of considering similarity to schedule as main criterion, the order agents will follow the provided schedule as long as it is feasible. This assumes that following the provided schedule really leads to a good global performance. To this end, the schedule should be updated in order to be able to react to disturbances or new opportunities, for instance when a new truck is available for transportation. If updating is not possible, deviating from schedule can lead to a better performance and hence, similarity to schedule should become less important. In such a situation, the release time of the schedule should be taken into account (the older the schedule, the less similar to schedule the solution should be).

5.4.2 HLES-TSS cooperation

Next to advice regarding the vehicle routing, the cross-docking HLES can also benefit from advice from a truck scheduling system (TSS). This scheduling system computes a truck schedule based on the tabu search approach described in Section 4.1.2. The schedule determines the assignment of inbound and outbound trucks to the different dock doors of the cross-dock. The cooperation between the HLES and the truck scheduling system happens again through the staff holon (see Figure 5.2). The staff holon provides the scheduling system with all necessary information and passes the computed schedule to the cross-dock holon. The cooperation between both systems is discussed in detail in the following paragraphs.

If the staff holon consults the TSS, it has to provide the scheduling system with up-to-date status information. To this end, the staff holon collects the following information from the resource holons (cross-dock and truck holons) and the most recent vehicle routing schedule.

Cross-dock information The staff holon needs to inform the TSS about the number of strip and stack doors of the cross-dock. The staff holon also needs to know the (rectilinear) distances between the dock doors.

Truck information The staff holon also requires the number of inbound and outbound trucks, their arrival and departure times and the number of freight units that have to be transferred between the trucks.

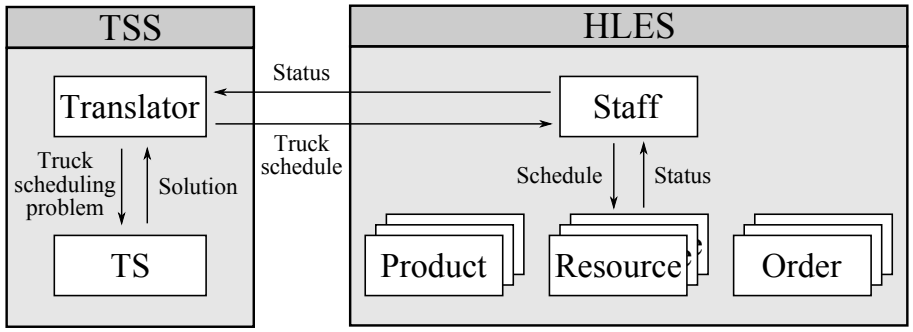


Figure 5.2: Cooperation between HLES and TSS.

Next to this information, the staff holon also has to provide the following data to the TSS:

- the estimated travel times between the dock doors (based on the rectilinear distances);
- the (fixed) time required to load or unload one order (L);
- the (fixed) truck changeover time (T);
- the weighting factors for the considered objective function (w_1 for the total travel time and w_2 for the total tardiness).

These data, the travel times between the dock doors excepted, are parameters of the truck scheduling problem. The staff holon can choose the values of these parameters. In the HLES prototype, these parameters are fixed. Their values used in the simulation experiments (see Chapter 6) are shown in Table 6.4. The staff holon can also specify the parameters of the applied tabu search approach.

The provided information is used by the scheduling system to construct a truck scheduling problem which corresponds to the current situation. Next, the TSS solves this problem by applying the tabu search approach explained in Section 4.1.2. The outcome then has to be 'translated' into a usable schedule for the cross-dock holon. To this end, the assignment of trucks to dock doors with the corresponding start and end times is derived from the solution and put in the correct data format. In the obtained truck schedule, the start times of the outbound trucks are as early as possible in order to minimize the tardiness. This can result in dock door assignments with a long duration whereof only a fraction of the time is really used for loading of the truck. To limit the time a truck spends at a dock door, the TSS can adapt the start time s_j of the

outbound truck j based on the end time indicated in the schedule:

$$s_j = \max\left(g_j - \beta\, L \sum_{i=1}^{n_1} f_{ij}, s_j\right) \tag{5.2}$$

in which

- g_j is the end time of outbound truck j;
- β determines an extra margin on the duration of the assignment;
- L is the time needed to load or unload one product unit;
- n_1 is the number of number of inbound trucks;
- f_{ij} is the number of product units that have to be transported from inbound truck i to outbound truck j.

In this expression, $L \sum_{i=1}^{n_1} f_{ij}$ is the total time needed to load the truck and $\beta \geq 1$. Subsequently, the resulting truck schedule is passed on to the staff holon which sends the schedule to the cross-dock holon.

The cross-dock holon uses this truck schedule as a guideline to organize the internal cross-dock operations. When exploring or intention ants issue requests to the cross-dock holon about loading and unloading of the orders, the cross-dock agent will consult the available truck schedule to determine the dock door at which the operation should take place. If the start time of the requested operation (from the exploring or intention ants) corresponds to the start time as indicated in the truck schedule, the corresponding dock door will be employed. Otherwise, if the deviation in start time exceeds a certain threshold, the cross-dock agent chooses a dock door based on its local decision-making mechanism (see Section 5.5.2). The forklift holons can now, based on the position of the dock door, determine if they are able to execute the operation and can give an accurate estimate about the duration of the operation. The ants can then also consult the correct dock door holon to check if a time slot with that duration is available. When an order holon actually makes a reservation at a truck holon, this truck holon will be informed about the correct dock door to which it is assigned.

The TSS will compute a new truck schedule when it is triggered by the staff holon. The trigger conditions can be specified in a similar way as for the HLES-VRSS cooperation (see Section 5.4.1). However, as the scheduling system also uses information from the latest vehicle routing schedule, the same trigger condition as for the VRSS will be used. So, the staff holon will trigger the truck scheduling system when it receives an updated vehicle routing schedule. In the HLES prototype, the TSS is not able to take the current situation completely into account. For instance, the fact that trucks can already be docked at a

certain dock door is not considered. To account for extra constraints, the tabu search approach applied by the TSS should be adapted in a similar way as in the VRSS approach (i.a. by prohibiting certain moves). However, this simplification does not render the cross-docking HLES useless, as the cross-dock holon will deviate from schedule if required.

Similar to the HLES-VRSS cooperation, it is also possible to provide the truck scheduling system with expected status information (if the current status is not known) or with short-term future status information. Also here, it is assumed that following the provided truck schedule leads to a good global performance.

5.4.3 Schedule execution

The previous sections explained how the cross-docking HLES can cooperate with two scheduling systems. The most important goal of scheduling is to efficiently organize the specified work, i.e. to perform the necessary work with the required quality and at the desired time, while trying to optimize a certain performance criterion (e.g. cost or a similar objective). Obtaining a good global performance was indeed the main reason for the cooperation presented in the previous sections. Other purposes of scheduling are for instance providing visibility of future plans, serving as a capacity check for higher-level planning systems and evaluation of performance [12].

However, in reality, there seems to be a 'gap' between the theory and practice of scheduling [65, 127, 183]. One of the reasons for this problem is the occurrence of disturbances. Three categories of disturbances can be distinguished [183]: disturbances regarding the capacity (e.g. machine breakdown), disturbances related to orders (e.g. rush order) and disturbances related to the measurement of data (e.g. poorly estimated processing time). As a result, schedules are known to become ineffectual within minutes after their release [146, 215]. To make scheduling more relevant and applicable, scheduling techniques should be able to deal with disturbances and real-time events. This is the focus of *dynamic scheduling* [144].

In the literature, three types of dynamic scheduling can be distinguished: completely reactive scheduling, predictive–reactive scheduling, and robust proactive scheduling [12, 85, 144]. In *completely reactive scheduling*, no schedule is generated in advance. Decisions are made locally and in real-time, for instance by applying dispatching rules[1]. These rules are usually intuitive and easy to

[1] A dispatching rule selects the next task to execute every time a resource becomes available [15]. The decision is only based on the current status, no look-ahead information is used [15]. Examples of dispatching rules are shortest processing time (SPT), earliest due date (EDD), minimum slack time (MST), etc.

implement, but global scheduling has the potential to significantly improve the performance as dispatching rules are myopic [144]. As examples in the context of cross-docking, [128] and [222] propose dispatching rules to dynamically assign trucks to dock doors.

In *predictive–reactive scheduling*, schedules are revised in response to real-time events. This updating of an existing schedule is generally known as rescheduling. Many predictive-reactive scheduling strategies make simple schedule adjustments accounting only for the considered performance measure. However, the new schedule can deviate significantly from the original schedule, affecting related planning activities. This can also lead to 'nervous' behavior, and rejection of the constant load of new schedules by the personnel [183]. To overcome this problem, *robust predictive-reactive scheduling* focuses on building schedules to minimize the effect of these disruptions, for instance by also taking the deviation from the original schedule into account next to the original performance measure. Predictive–reactive scheduling needs to tackle two main issues: when and how to react to real-time events [12, 144]. Three policies have been proposed in the literature for the first issue: new schedules can be generated at regular intervals (periodic policy), in response to an unforeseen event (event-driven policy) or periodically and when an event occurs (hybrid policy). Regarding the second issue, two main rescheduling strategies are considered in the literature: schedule repair and complete rescheduling. In schedule repair, only local adjustments are made to the current schedule. Complete rescheduling on the other hand generates a new schedule from scratch. For some examples of predictive–reactive scheduling approaches in manufacturing, see [47, 91, 142].

Robust pro-active scheduling or simply robust scheduling focuses on creating a schedule which, when implemented, minimizes the effect of disturbances on the considered performance measure. One approach to do this is by considering a range of scenarios representing different realizations of disruptions to the schedule. The goal is then to find a schedule that will perform well under a wide range of these scenarios. Robust pro-active scheduling can be viewed as a form of under-capacity scheduling, in which the historical performance of the equipment is used to determine the amount of work scheduled in a certain time period [12]. Some examples of robust scheduling in the context of vehicle routing can be found in [90, 179, 186].

The occurrence of disturbances is however not the only cause of the gap between the theory and practice of scheduling. To generate a schedule, a certain model of the underlying reality is considered. Because of the complex and dynamic nature of the real world, often many assumptions have to be made and this results in a simplified and approximated model of reality [61, 65, 183, 203, 215]. For instance, in order to be able to construct a mathematical model (e.g. linear programming model or (mixed) integer programming model) of

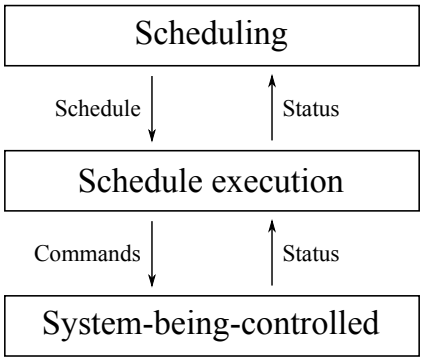


Figure 5.3: Positioning of schedule execution.

the considered problem, several aspects can only be taken into account in a simplified way while other details have to be disregarded completely. Even if it is possible to model the scheduling problem in great detail, it would require a large development effort and afterwards also a large maintenance effort to keep the model in accordance with reality. Moreover, the computational complexity of the resulting model would cause high response times which render the model useless. So, limited development and calculation time require that an abstraction is made of the real-world problem. If schedules are based on such a simplified model, they have to be adjusted manually. However, these actions will deteriorate the performance on which the schedule was based, and the reason for using such a scheduling approach even becomes doubtful [183].

These simplifications in the scheduling together with the occurrence of disturbances indicate that only a schedule is not sufficient to control the operations. In practice, there also needs to be a system in place to execute this schedule [23, 92, 215]. This system is situated on an intermediate level between the scheduling level and the system-being-controlled (see Figure 5.3). Bongaerts [23] describes *schedule execution* as "the process of taking the on-line resource allocation decisions, based on the existing schedule but also considering the actual state of the resources and orders ..., and reacting to disturbances if necessary". So, schedule execution will ensure that the operations are performed in reality, by issuing the necessary operational commands to equipment and personnel. Executing the schedule is more than strictly following the operation sequences of the schedule. The schedule execution system will take its own decisions, but considers the schedule as a kind of objective or advice [23, 92, 216]. So, schedule execution allows the schedule to be executed in reality, while accounting for deviations and aspects that are not considered by the scheduling system.

In many real-world situations, there is no schedule execution system in place and schedule execution is performed by human operators. They will autonomously manage the operations, based on their perception of the shop floor information and the provided schedule [12, 92, 215]. For instance, as in completely reactive scheduling, every time a resource becomes available, it has to be decided which task to execute next. The workers can decide to keep the sequence indicate by the schedule (even if the start times are not correct anymore) or they can apply dispatching rules (e.g. EDD) using the release times and due dates of the schedule. They also have to include (auxiliary) tasks not considered by the schedule. They have however no clear idea about the impact of their decisions on the rest of the system and on the global performance.

When a schedule execution system is in place, this system is also responsible for monitoring the operations and to inform the scheduling level about the current status (including events and status of the in-progress schedule). This allows the scheduling system to update the schedule, if this system is capable of rescheduling. The schedule execution system reacts autonomously to disturbances, but can request the scheduling level to modify its advice, i.e. the schedule [92]. This allows for a cooperation scheme in which the scheduling system adjusts the schedule when a larger disturbance occurs, while the schedule execution system deals autonomously with the smaller deviations in between. In this way, the reaction time in case of disturbances could be short, while the available time for optimization could be an order of magnitude higher [23].

Schedule execution can be considered as one of the tasks of an MES or LES (corresponding to the production dispatching and production execution management activity, see Section 2.1). The HLES implementation presented in this chapter is also capable to perform schedule execution. Indeed, Sections 5.4.1 and 5.4.2 explained how the cross-docking HLES can cooperate with two scheduling systems. The HLES will try to execute the provided schedules, while accounting for deviations and aspects that are not considered by the scheduling systems.

As indicated in Chapter 2, the logistics domain is dynamic. Disturbances and unexpected events occur and as a result, the operations will deviate from the generated schedule. For instance, traffic jams, the illness of a forklift driver or the breakdown of a truck will disturb the execution of the cross-docking operations as they are scheduled. However, because of its self-organizing nature, the HLES is able to deal with these disturbances. As the order holons keep on refreshing their intentions, the affected orders will quickly detect that their current intention is not valid anymore. The decision mechanisms of the orders and resources described in Sections 5.4.1 and 5.4.2 allow these orders to find and reserve new intentions which are similar to the provided schedule.

As indicated in the previous sections, the cross-docking HLES cooperates with a vehicle routing scheduling system and a truck scheduling system. Both scheduling systems solve a problem that is a simplified and approximated view of the real-world problem. For instance, the VRSS does not consider the internal operations inside the cross-dock and the availability of the internal resources. The HLES will however take these aspects into account. The order holons will not only consider the transport operations indicated in the truck schedule, but will also make the necessary reservations for the unloading, internal transportation and loading operations. Also the TSS does not deal with all details of the cross-docking operations. For instance, it is assumed that there is always personnel and equipment available to transfer the goods from the inbound to outbound trucks. If this is not the case, the time between unloading and loading of the goods will be longer that the estimated travel time. The HLES will notice this when the order holons try to reserve the internal transport operations at the forklift holons. The order holons then will try to adapt their start time for the loading at the outbound trucks.

The process that compensates for the simplifications in the schedule is sometimes called the *last-mile planning process* [216]. This process executes a schedule and tries to optimize all aspects that are left out of the schedule. If this process performs well, it facilitates the scheduling task. It allows the scheduler to solve easier problems because it will fill in the missing elements and make final adjustments. For instance, the VRSS takes the volume capacity of the trucks into account, but assumes that all trucks have the same capacity. This is a simplification of reality if the trucks are not homogeneous and/or if a weight capacity has to be taken into account. While it is possible to adapt the vehicle routing algorithm to account for these aspects, it can be easier to leave the handling of these issues to the last-mile planning. The different capacities can be taken into account by simply adapting the model and/or local decision mechanisms of the truck holons. Moreover, including more details into the scheduling system comes at the expense of higher development and maintenance costs. So, while it is possible to extend the scheduling algorithm to include extra aspects, this effort should be balanced with the performance loss without this extension. Besides, because the considered scheduling problem is more general, the underlying model has to be less frequently adapted to correspond to the real-world situation.

The cross-docking HLES presented in this chapter executes two schedules: a vehicle routing schedule and a truck schedule. In general, the HLES can even execute more than two schedules together, at least if they deal with distinct aspects. This allows dividing the complete scheduling problem in smaller subproblems, which can be more easily tackled.

As explained in Chapter 2, the HLES has a view on the expected future of the

system as short-term forecasts are generated. These forecasts can be used by the order and resource holons to improve their decision making. This allows for a more intelligent way of schedule execution than can be performed by human operators, who have no clear idea about the impact of their decisions when they have to deviate from the provided schedule. Moreover, these forecasts can also be used to provide extra information to the scheduling level (see Figure 5.3). The provided status information can now also include predicted events and the predicted status of the in-progress schedule. In this way, the scheduling system can react even before an event takes place, so pro-active instead of reactive rescheduling is possible.

5.5 PROSA for cross-docking

The previous section explained how an HLES implementation can cooperate with a scheduling system through the staff holon. To develop a cross-docking HLES, also the various product, resource and order holons need to be developed. This section presents which relevant entities of the world-of-interest will be reflected by holons. For these holons, models of their behavior and local decision making mechanism are required. While the models are domain-specific and can be reused for other HLES applications in the same domain, the decision mechanisms are typically application-specific.

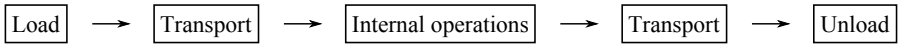
The setting for the cross-docking application considered here is the following. Orders have to be transported from their origin to their destination via a cross-dock. Each order consists of one pallet that has to be transported. Several trucks are available to perform the necessary transport operations. These trucks are responsible for pick-up of the orders at their origin and for delivery of the orders at their destination. The cross-dock has several strip and stack doors at which the trucks can be unloaded and loaded. Inside the cross-dock, one or more forklift trucks are available to unload the arriving goods, to transport these goods and to load them into the correct outbound truck. The orders can also be transported to a temporary storage area.

The next sections describe the models and decision making mechanisms for the product, resource and order holons that are considered in the cross-docking HLES.

5.5.1 Product holon

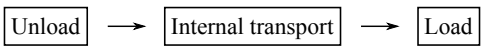
As explained in Chapter 2, a product holon corresponds to a task type and contains the knowledge on how instances of its type can be processed by the resources. For the cross-docking application, this means that there will be a product holon instance for every combination of origin-destination pair and product type. In this way, the product holons are able to determine the next possible operations for all corresponding order holons.

For the cross-docking application, a product holon will specify the following sequence of operations for its order holons:

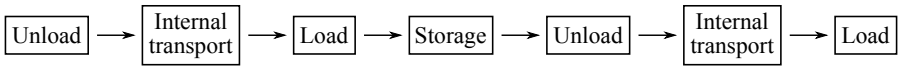


An order first has to be picked up (loaded) at its origin and then has to be transported to the cross-dock. After the internal cross-dock operations, the order has to be transported to its destination where it can be delivered (unloaded). The product holons have to give all possible options for these operations. For the transport operations from and to the cross-dock, the possibilities will depend on the *origin-destination pair*. These possibilities should account for all possible transportation means and alternative routes. As the number of possibilities can be large, the product agent has to decide which options are provided to the order holons. As a consequence, if an order holon receives advice from the VRSS, the vehicle routing schedule can indicate a transport operation that is not included in the options. The order holon is however allowed to explore this option. But, all solutions found by the order holon (or its exploring ants) will be checked by the product holon to see if they are valid, i.e. if the necessary operations are correctly executed. Note that, if needed, the product holons can include extra operations (not provided in the sequence mentioned above). For instance, if an order is somehow transported to a wrong destination, the product holon will specify an extra transport operation from its current location to the correct destination.

For the internal cross-dock operations, at least two sequences of operations are possible, one with and one without temporary storage:



or



Note that the product holons also have to specify the process parameters and possible constraints of these operations. For instance, for each transport operation, the origin and destination have to be specified. Some parameters will however be determined by the involved resource holons as they are not known by the product holon. For instance, the origins and destinations of the internal transport operations (i.e. the dock doors) depend on the truck to dock door assignment.

It can be necessary to have a product holon for every *product type*, if there are differences in how these product types have to be handled (e.g. toxic products). This can result in different operation sequences for different product types, in different constraints on the operations or in different process parameters of the operations. For instance, if the orders represent refrigerated products, the product holon has to specify the allowable temperature range to which the package can be exposed during transportation. The holon can also specify the maximum time the orders can be hold in storage and the maximum time the product can be exposed to ambient temperature during the internal transportation in the cross-dock.

Section 5.3 explained how the HLES can perform multi-resource allocation. If multiple resources are required at the same time, the product holon has to indicate this to its order holons. In the cross-docking HLES, the product holon will specify a combination of operations for the loading and unloading at the cross-dock: a load (or unload) operation by a forklift truck, an unload (or load) operation by the truck and a 'standby' operation of a dock door. Figure 5.4 then shows the complete sequence of operations specified by the product holons in Petri net format [137]. The places correspond to operations in this Petri net. Every time a transition fires, the places that contain a token indicate the next possible operation(s).

5.5.2 Resource holon

For every entity in the world-of-interest that can be considered as a resource, there will be a resource holon. For the cross-docking application, this means that for instance all trucks, forklift trucks, dock doors and temporary storage areas will be represented by a resource holon. As shown in Figure 2.3, the dock door holons, the forklift holons and the temporary storage holon together form a cross-dock holon. Figure 5.5 then shows a resource graph for a situation with one cross-dock (CD), two origins (O1 and O2), two destinations (D1 and D2), two pick-up trucks (PT1 and PT2) and one delivery truck (DT1). The connections between the resources indicate how the ants can travel over this graph. The origins are connected to the pick-up trucks, which are connected to

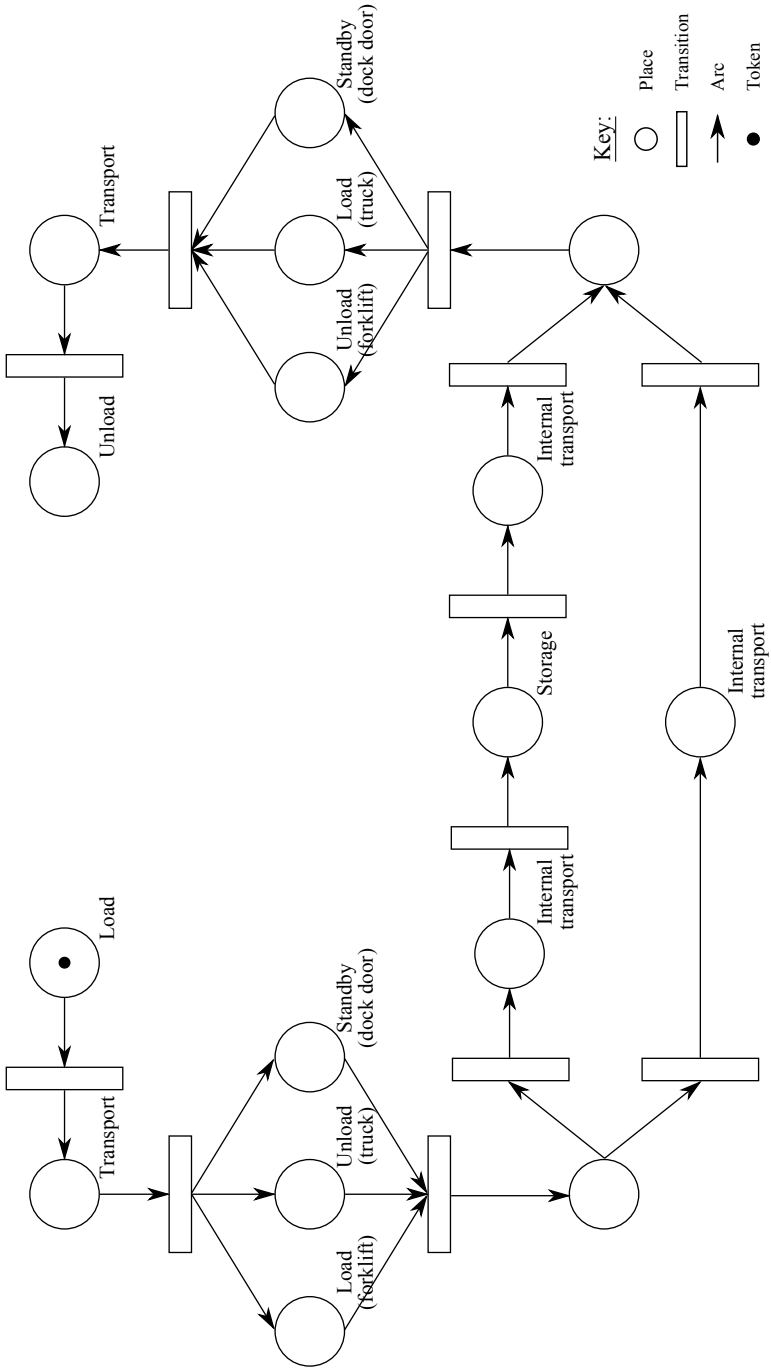


Figure 5.4: Petri net representation of the operation sequence specified by the product holons.

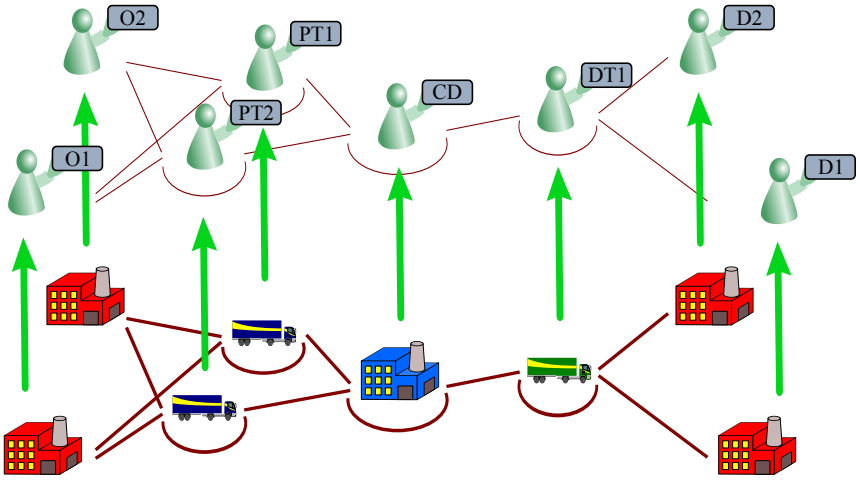


Figure 5.5: Resource graph for the cross-docking application. The physical resources are shown together with their corresponding resource holons.

the cross-dock. The cross-dock is connected to the delivery truck, which is in turn connected to the destinations. Note that the trucks and cross-dock nodes are connected to themselves, indicating that the ants can perform multiple operations sequentially at these resources.

Each resource holon has a model of the dynamic behavior of its physical resource (environment entity module) and is responsible for its own local decision making (agent module), as explained in Section 2.2.3. The following paragraphs will shortly discuss both aspects for the various resource holons.

Truck holon

To reflect the real-world trucks, every truck holon contains a multimodel that describes its behavior. Figure 5.6 shows the top-level model of this multimodel, which represents the behavior in case of breakdown and repair. When this model is executed, transition *T1* can fire because the place *Repaired* contains a token and no in boundary condition is defined. The submodel *Operating* is then executed until the *Breakdown* boundary condition is true. This submodel represents the behavior of the truck when it is operating under normal conditions. If a breakdown occurs, the out boundary condition is fulfilled and a token is placed in *BrokenDown*. Next, the truck waits (*Waiting* submodel) until the reparation starts and then the *Repairing* submodel is executed. When the

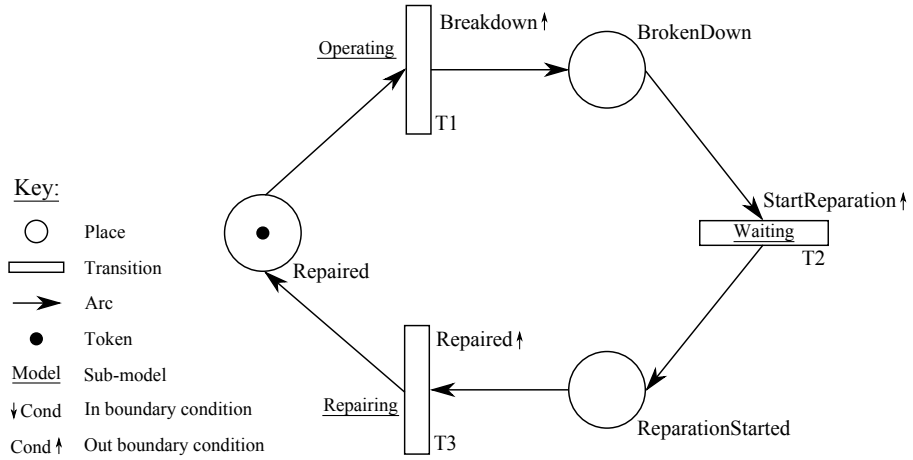


Figure 5.6: Top-level model of a truck.

Repaired out boundary condition is true, a token is placed in *Repaired* and the *Operating* submodel can be executed again. Figure 5.7 then shows this *Operating* model. The truck is standby (*Standby* submodel) until an operation is assigned to it. If the truck is ready to execute this operation (i.e. the truck is at the correct position and no auxiliary drive operation is needed), transition *T3* can fire, otherwise transition *T4* fires and the *NotReady* model indicates that the operation cannot be performed. If *T3* fires, the submodel corresponding to the operation type will be executed: *Loading* (T6), *Driving* (T7) or *Refueling* (T8). If the operation is finished, a token is placed in *Executed* and the truck will again be in standby mode.

Based on this model, the truck holons are able to estimate the duration of an operation. The holons need to know this duration in order to make reservations in their local schedule. Note that some (operation- and duration-dependent) slack time will be included in these reservations. The reservations are mostly made based on the first-come, first-served (FCFS) policy, i.e. a new reservation cannot overlap with an already made reservation. However, in some situations, the truck holons will give priority to load and unload operations compared to transport operations. In this case, the time slot reserved for the transport operation (and consecutive slots for unloading operations) will be shifted forward in time[2]. A transport slot can also be shifted forward if a new order is assigned to an already reserved time slot, but has a later start time. This mechanism allows to batch several orders (see Section 5.2). To illustrate this, consider the

[2]A maximum allowed shift time can be specified.

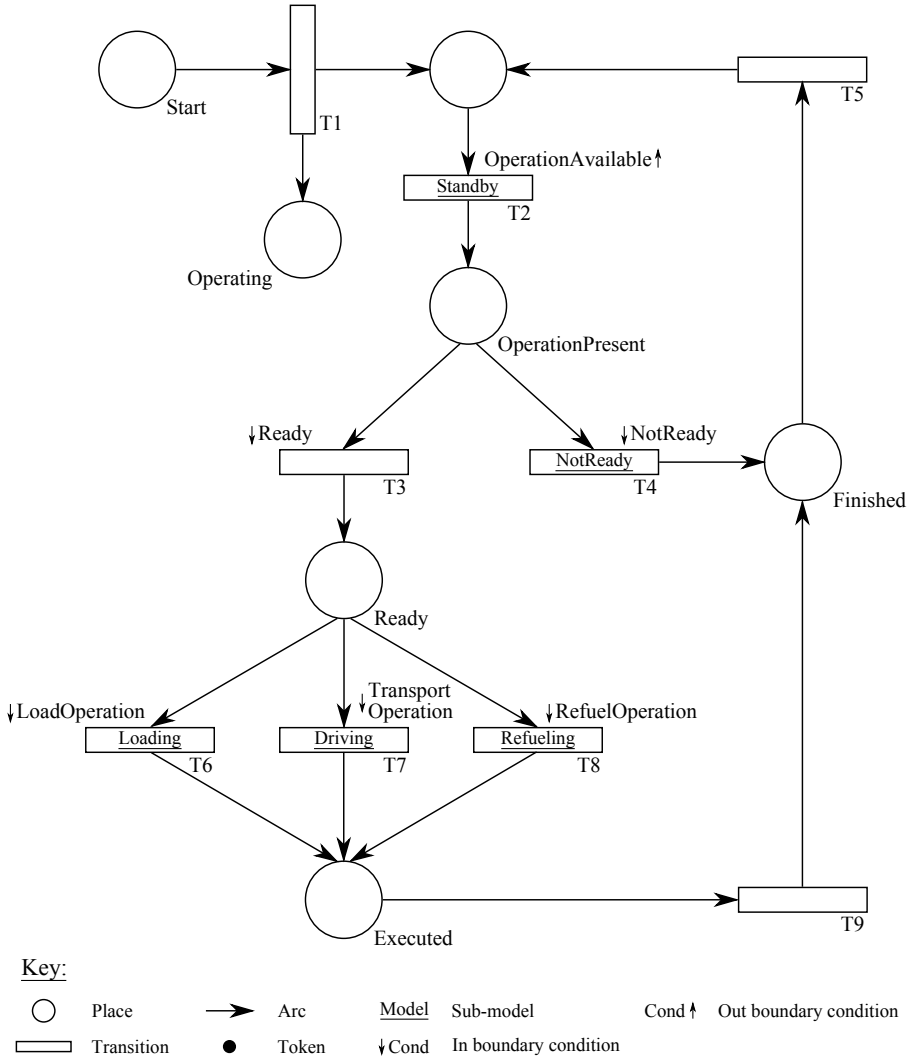
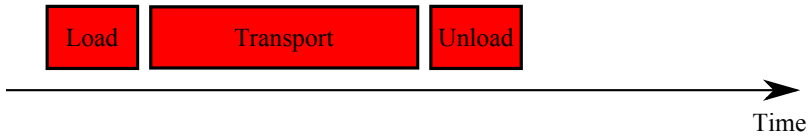
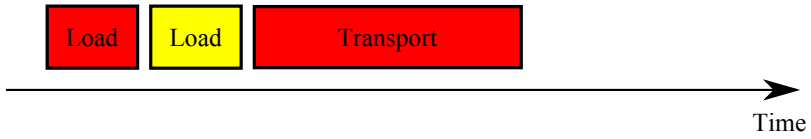


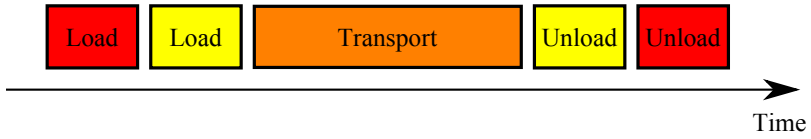
Figure 5.7: *Operating* submodel of a truck.



(a) Load, transport and unload operation of one order are reserved.



(b) Reservation of second load operation.



(c) Reservation of second transport operation and unload operations.

Figure 5.8: Example of the reservation of a load, transport and unload operation.

local schedule shown in Figure 5.8a. Three consecutive time slots are reserved; to load, transport and unload one order. If a second order wants to make a reservation for the same transport operation, this order first tries to reserve a load operation after the already reserved load slot. To this end, the truck holon will shift the transport slot forward in time (see Figure 5.8b). Then, the truck holon will assign the order to the already reserved transport slot and reserve new slots for the unload operations (see Figure 5.8c). Note that the truck holon takes the capacity of the truck into account and will not make a reservation leading to an exceeding of the capacity.

As explained in Section 5.1, the truck agents will also make reservations for auxiliary drive operations if needed. If the model indicates that the truck is not at the correct position (i.e. is not ready) to execute an operation, the agent will insert a time slot for the necessary drive operation. The duration of this operation can again be determined based on the truck model.

Two restrictions are imposed on the local scheduling. Firstly, to prevent that a truck occupies a dock door for a long period, the loading of the truck is only allowed in a certain time interval before the following transport operation. Secondly, to limit the number of shifts forward, if a transport slot will be shifted forward in time, this slot will directly be shifted for a minimum amount of time.

Blocking A drawback of the described local scheduling approach (applied in the HLES prototype) is the occurrence of '*blocking*'. Once a transport slot is reserved for multiple orders, it is impossible to reserve a new slot for these orders that overlaps with the current reservation. This is caused by the applied FCFS policy and because the involved orders have to switch one by one from the current to the new slot. To explain this in more detail, consider the following situation. Based on the provided vehicle routing schedule, two orders have reserved the same time slot for a transport operation. An update of this schedule proposes to shift the reserved slot forward in time (larger than the maximum allowed shift time, but overlapping with the current transport slot). Both order holons then try to find a new solution in accordance with the schedule. If one of the two order holons tries to reserve the indicated time slot, the resource holon does not have to consider the current reservation of this order. However, the other order holon still occupies the original period and 'blocks' the new reservation. Indeed, based on the FCFS policy, the resource holon will not allow an overlapping reservation. As a result, the order holons will only be able to reserve a new time slot after the completion time of the currently reserved slot. This leads to a reduced performance of the control system. A possible solution could be to relax the FCFS policy when the resource holon notices that all affected orders try to reserve a new transport slot. Another possibility is that the affected orders (temporarily) form an aggregated order holon which is able to reserve the new time slot on behalf of its subholons.

Cross-dock holon

As mentioned above, the cross-dock holon is an aggregated holon consisting of dock door holons, forklift holons and a temporary storage holon. It has to deal with order requests about the internal cross-dock operations and will delegate these requests to its subholons. The cross-dock holon will successively receive requests from an order to be unloaded, transported and loaded (or alternatively with temporary storage in between).

First, an order holon will try to make a reservation to be unloaded from a truck, by requesting a load operation from a forklift truck. The cross-dock holon then has to determine to which dock door the involved pick-up truck will be assigned (if no reservation is made yet). To this end, the holon will consult the available truck schedule (as explained in Section 5.4.2). If this truck schedule contains a time slot with a start time corresponding to the requested start time of the unload operation, the indicated dock door will be employed. Otherwise, if the deviation in start time exceeds a certain threshold, the cross-dock holon chooses the dock door that is first available according to the current reservations.

If the dock door is known, the cross-dock holon delegates the request to one of the forklift holons. This holon can now, based on the position of the dock door, accurately estimate the duration of the operation and determine if the operation can be executed. To select a forklift, the cross-dock holon sorts the forklift holons based on their predicted load (number of reservations). Starting from the forklift holon with the lowest load, the cross-dock holon delegates the request one by one to the forklift holons until a reservation can be made for the load operation. If no forklift holon is able to reserve a time slot, the cross-dock holon will propagate resource intentions (see Section 2.2.4). When a reservation is made, the order holon will also try to make a reservation for a standby operation. The cross-dock holon will delegate this request to the dock door holon in charge of the selected dock door.

Next, the order holon will request the cross-dock holon to reserve a time slot for an internal transport operation. The cross-dock holon will delegate this request to the forklift holon that made a reservation for the load operation. To know the destination of the transport operation, the cross-dock holon has to determine to which dock door the delivery truck will be assigned. To this end, the cross-dock holon will again consult the available truck schedule. If a reservation for the internal transport operation can be made, the cross-dock holon will also delegate the following unload request of the order holon to the same forklift holon. The subsequent request for a standby operation will then be delegated to the dock door holon corresponding to the selected dock door.

If the order holons also try to make reservations for temporary storage operations, the cross-dock holon delegates these requests to the storage holon. The following paragraphs provide more details about the scheduling decisions made by the subholons of the cross-dock holon.

Forklift truck holon

In order to be able to estimate the duration of the operations delegated to them, the forklift truck holons contain a multimodel that describes their behavior. As for the truck holons, Figure 5.6 shows the top-level model of this multimodel and Figure 5.7 shows the *Operating* submodel. The lower level models are however not the same. For instance, the *Driving* model is adapted to reflect the driving of the forklift trucks inside the cross-dock, which are assumed to move rectilinear.

For each order, the forklift truck holons will reserve consecutive time slots for a load, an internal transport and an unload operation. The reservations are made based on the FCFS policy. Similar to the truck holons, the forklift truck holons will include slack time in these reservations and will reserve time slots

for auxiliary drive operations if needed. The forklift trucks will only transport one order at a time, so batching of the orders is not possible.

Dock door holon

The various dock door holons have to make reservations for the standby operations, which are delegated to them by the cross-dock holon. They do not have an explicit model of their behavior, as the duration of these operation are determined by the forklift truck holons (as explained in Section 5.3). The dock door holons will then use the FCFS policy to reserve time slots of the indicated duration. The time slots corresponding to loading or unloading operations of the same truck are however combined into one slot, indicating that the involved truck occupies the dock door for the complete duration of the loading or unloading. The dock door holons also ensure that there is a certain minimum amount of time between the reservations for different trucks to account for a truck changeover.

Storage holon

If an order holon tries to reserve a time slot for a temporary storage operation, the cross-dock holon delegates this request to the storage holon. This holon can reserve multiple overlapping slots, but takes the capacity of the physical storage area into account. The holon will reserve a time slot from the requested start time until the start of the next operation of the order (a load operation into a forklift truck).

5.5.3 Order holon

For every physical order, there will be a corresponding order holon. In the context of cross-docking, order holons correspond to freight units that have to be transported, for instance pallets. More generally, there will be an order holon for every task that needs to be executed (e.g. refueling). The order holons are responsible for routing their real-world order through the cross-docking system. To this end, the holons will send out exploring ants to search for possible solutions on their behalf. The discovered solutions are then evaluated and the order holon selects the most attractive solution to become its intention. Subsequently, the order holon reserves this intention at the involved resource holons by sending intention ants.

The rate at which exploring ants are created by an order holon can be variable. If an order holon has no intention (initially or when its current intention is not valid anymore), it is interesting for the holon to quickly have some alternative solutions to choose from. Therefore, the rate at which exploring ants are created will be higher in this situation than in the situation when the order holon has a valid intention.

To select an intention, the order agent will first determine the 'best' solution among the solutions found by the exploring ant. Then, the agent has to decide if it will abandon its current intention in favor of this solution. In both cases, several performance measures can be considered. The current implementation takes the following performance measures into account:

Similarity to schedule The order agents prefer solutions that are similar to the provided VRPCD schedule. Equation (5.1) is used to evaluate the similarity.

Tardiness Solutions with a lower tardiness are preferred. If d is the due date and e_n^a is the time at which operation n, the unload operation at the destination, is completed, the tardiness of solution a is defined as $\max(e_n^a - d, 0)$.

Start time The order agents prefer solutions with an earlier start time, i.e. in which the loading into a pick-up truck starts earlier.

Waiting time The waiting time of a solution is the total time, from its start time until completion, that the order is not being processed. The time in temporary storage is not considered. This measure accounts for the time the order has to wait in between two operations (e.g. once the order is loaded into a truck, it has to wait until the truck departs). The order agents then have a preference for solutions with a lower waiting time.

Resource utilization The order agents prefer solutions in which the resources are efficiently used, i.e. in which batching occurs. To measure the degree of batching, a 'batching value' BV is calculated for each solution a as follows:

$$BV(a) = \frac{\sum_{i=1}^{n} b_i^a}{n} \tag{5.3}$$

in which

- n is the number of operations in solution a (numbered from 1 to n);
- b_i^a is the number of orders on which operation i is performed.

Creation time Solutions that are more recent are preferred.

To determine the best solution among the found solutions, these solutions are compared pairwise. To compare two solutions, the performance measures are analyzed in the given sequence. If one solution is significantly better than the other according to a performance measure (if the absolute or relative difference is higher than a certain threshold), that solution is preferred and the other measures are not considered. Otherwise, the next performance measure is considered.

The best solution and the current intention are then compared in a similar way. If this best solution appears to be significantly better than the current intention according to one of the considered performance measures, the order holon can adopt this solution as its new intention. However, in order to behave in a socially-acceptable manner (as explained in Section 2.2.4), the order holon will adapt its intention only with a certain probability. This probability is dependent on the specific performance measure for which the solution is better. The probability can also be a function of the performance gain that can be obtained.

This selection mechanism is part of the application-specific decision making of the order holons. It can be seen as a plug-in of the HLES that can easily be replaced. For instance, it can be more interesting in other applications to consider alternative performance measures (e.g. lead time) or to consider them in another sequence. Another possibility is to use a (weighted) combination of performance measures instead of considering these measures separately. However, if the HLES cooperates with a scheduling system, similarity to the provided schedule should be considered as a main criterion. This assumption is based on the belief that following the schedule leads to a good global performance. In any case, the order holons should behave in a socially acceptable way. They should only change their intention probabilistically and if the expected performance increase is significant.

5.6 Conclusions

This chapter started with a description of the adaptations and extensions to the basic HLES functionality. A first adaptation, in order to support mobile resources, is the ability to reserve auxiliary operations. Secondly, reservations can be made for several orders together in order to support batching. Thirdly, the HLES functionality is extended with multi-resource allocation. These adaptations are however more generally applicable and are not only valuable in a logistic context. They are for instance also of interest for manufacturing applications and can be included in HMES implementations.

The first adaptation, the ability to reserve auxiliary operations, is also useful for production processes, e.g. to account for setup times or the preheating of a furnace. The second adaptation is obviously also interesting for manufacturing applications that make use of batch production. Multi-resource allocation is not only suited for the logistics domain, but can for instance also be applied for machines that make use of additional tools.

This chapter also describes in detail how the cross-docking HLES can cooperate with a vehicle routing scheduling system and a truck scheduling system. This cooperation happens through the staff holon. In similar ways, the HLES should be able to cooperate with other scheduling systems, for instance to determine where to (temporary) store incoming freight. The provided scheduling advice supports the HLES in finding good global solutions. On the other hand, the HLES is able to execute this schedule, while accounting for (small) deviations and possible simplifications. This schedule execution is an important and often overlooked aspect of scheduling practice.

The domain-specific models and application-specific decision mechanisms for the various holons are also discussed in this chapter. The decision making components can be seen as plug-ins to the system that can be easily replaced by other ad hoc algorithms or rules. Specific rules are developed for the cross-docking HLES in order to have a functioning prototype that can be tested (see Chapter 6). Note that the implemented decision mechanisms can still be improved. For instance, for the truck holons, the rule to shift transport slots forward in time mostly works well, but there is a risk that this slot will be shifted many times, resulting in a very large deviation compared to the original reservation. The decision mechanism of the truck holons could also be adapted in order to try to avoid blocking.

If decision making mechanisms are provided, the HLES is able to provide a view on the expected short-term future of the system. This short-term forecasting ability is a very important aspect of the HLES. The consequences of decisions based on the provided decision making mechanisms now become visible in advance. In case of disturbances or changes in decision making, the generated short-term forecasts will be adapted accordingly. While the provided up-to-date visibility is a valuable contribution in itself, these forecasts also allow to improve the local decision making. For instance, based on the expected arrival time at the cross-dock of a belated pick-up truck, a delivery truck holon can make a more deliberate choice whether to wait or not for that truck.

The HLES can control the cross-docking operations in three different ways. In a first manner, the HLES acts as a decision support system. It only provides support to the cross-dock manager and has no direct control of the cross-docking operations. The HLES generates short-term forecasts, but it is the manager

who decides about the execution. The forecasts will automatically be adapted if the real execution deviates from the proposed execution. Secondly, the HLES functions autonomously. The execution system is completely in charge and can directly control the physical resources, i.e. the HLES decides when and which operation the various resources should execute. The third way is a combination of both. The HLES is in charge, but the cross-dock manager can intervene. As the consequences of the decisions of the various holons are visible, the manager can detect undesirable situations in advance and try to prevent these situations by making alternative decisions. If a simulation environment is available, based on the virtual execution capabilities of the holons, the manager can simulate the effects of certain decisions before applying them in reality. For instance, this what-if simulation can be used to see the effect of the release of a rush order on the completion times of other orders. For the simulation experiments discussed in the next chapter, the HLES operates in the second or third way.

Chapter 6

Experimental evaluation

In the previous chapter, an approach has been presented to organize the logistic operations of a cross-docking terminal. This chapter evaluates this holonic on-line approach. A research prototype is developed and tested in simulation. The goal of these tests is to show that the developed system works as intended and to show the value added of the holonic technology. The chapter is organized as follows. Section 6.1 shortly describes the simulation platform used to carry out the simulation experiments. In Section 6.2, the set-up used for the experiments is described. This section also lists the objectives of the simulation experiments. These experiments are then discussed in the next two sections. In Section 6.3, some specific scenarios are considered to study in detail the behavior of the proposed holonic control system and to indicate the value added of this approach. Section 6.4 then describes the results of replicated tests in order to show the benefits of the integration between the holonic control system and the proposed scheduling approaches. Three different control modes are compared. In the first control mode, the cross-docking HLES manages the logistic operations, without the help of an external scheduling system. In the second mode, a vehicle routing schedule and a truck schedule are provided by external scheduling systems and the HLES uses these schedules as an initial guideline. The third control mode makes use of rescheduling to adapt the initial schedule. The results of these experiments are presented and an in-depth discussion is provided. Finally, Section 6.5 concludes this chapter.

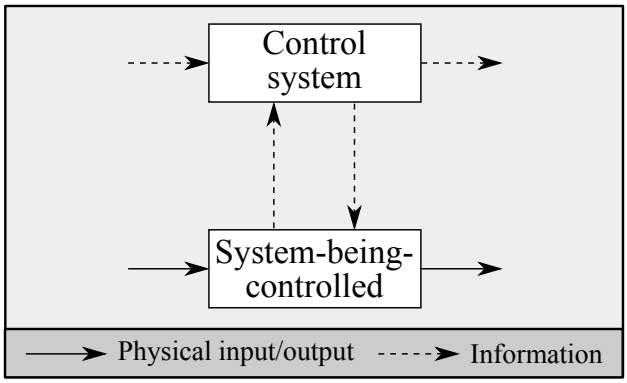


Figure 6.1: Control system versus system-being-controlled.

6.1 Simulation platform

In order to test the HLES prototype, a simulation platform is developed which allows software-in-the-loop simulation. In general, a distinction can be made between the *control system* itself and the *system-being-controlled* (see Figure 6.1) [204]. Here, the system-being-controlled corresponds to the logistic system and the control system corresponds to the HLES. Via control actions, the control system tries to influence the state of the system-being-controlled. For instance, the HLES will send commands to the logistic system in order to start or stop a transport operation of a truck or to start the loading of an order by a forklift truck. Note that the control system does not fully control the system-being-controlled as these control actions do not necessarily result in the intended effect. In order to be able to determine the proper control actions, the control system retrieves or observes state information from the system-being-controlled. For instance, the HLES is informed about the start and stop of operations and receives at regular times position updates of the various trucks.

Software-in-the-loop simulation uses the real control system (or a part of it) together with a model of the system-being-controlled (see Figure 6.2) [84, 165, 216]. The model of the system-being-controlled is called the emulation model, which is executed by a so-called *emulation*. So, the real control system, also used in real-world applications, is integrated in the simulation loop. This type of simulation allows to experiment with the control system on simulated systems before using the control system on a real system.

To perform the simulation experiments, the HLES prototype is connected with an emulation. This emulation mimics the behavior of the world-of-interest. To

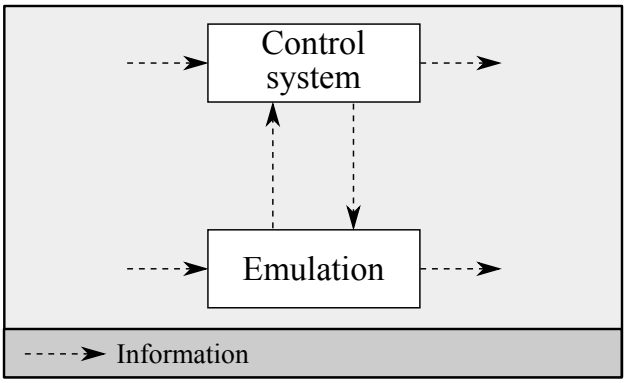


Figure 6.2: Software-in-the-loop simulation.

this end, the emulation makes use of a simulator specifically designed to execute Petri net driven multimodels (which are described in Section 2.5) [165, 216]. This simulator manages a set of so-called emulation entities, corresponding to all relevant (logistic) entities in the world-of-interest (e.g. trucks, forklift trucks, ...). Each entity is modeled by a state and a multimodel describing its life cycle. These models can be similar to the models used by the holons in the control system. However, where the values of the used parameters are the expected values for the control system, the emulation samples these values from a probability distribution. The multimodel simulator is then responsible for the management of the simulation time. It advances the time in discrete steps and keeps the states of all emulation entities synchronized with this time (by executing their multimodel).

The control system and the emulation both have their own logical time and own time management. Synchronization between these two components is required to guarantee correct behavior. For this purpose, a synchronization mechanism that can switch the emulation between a discrete-event mode and a real-time mode is used [165, 216]. In the discrete-event mode, the emulation executes as fast as possible. Updating the events happens instantaneously, i.e. the simulation time does not change. In this mode, it is possible to simulate faster than reality. Whenever the control is performing some action (e.g. making a certain calculation or executing some algorithm), the emulation switches to real-time mode. This ensures that the control system does not have less time to react as it would have in reality, but also that it does not get more time than in reality.

As mentioned above, the multimodels describing the behavior of the emulation entities make use of random variables (according to certain probability

distributions). The simulator makes use of pseudorandom number generators to generate values for these variables. The 'seeds' of these generators, which allow repeating the generated sequence of numbers, are for every simulation run written to a file. This file can then be used to initialize a next simulation run, allowing to replicate a simulation run. Note however that two simulation runs, executed with the same seeds, do not necessarily give exactly the same results, as the control system itself is also stochastic. This approach allows however to isolate and to demonstrate the effect of the control system.

During simulation, the emulation logs all relevant data. The resulting simulation log then contains the evolution over time of the state of all emulation entities and of the generated short-term forecasts. Based on this log, performance measures can be calculated and the execution of the simulation run can be visualized (see for instance Figures 6.4 and 6.5).

The simulation platform is, like the research prototype, written in Java (Java SE 6) and can run on any platform supporting Java.

6.2 Experimental set-up

This section describes the set-up used for the simulation experiments presented in the following sections. If the set-up of an experiment is different from the set-up described here, the specific deviations or additions will be mentioned in the corresponding description of the experiment.

General setting One cross-dock is considered and orders have to be transported from their origin to their destination via this cross-dock. Each order consists of one pallet that has to be transported. Several trucks are available at the cross-dock which are controlled by the cross-dock manager. These trucks are responsible for pick-up of the orders at their origin and for delivery of the orders at their destination. The trucks do not perform direct transportation from an origin to a destination and each truck can transport up to 33 pallets at a time. Inside the cross-dock, one or more forklift trucks are available to unload the arriving goods, to transport these goods and to load them into the correct outbound truck. The orders can also be transported to a temporary storage area. It is assumed that equipment and personnel is always available, i.e. breaks or shifts are not taken into account. Another simplification is that the sequence in which a truck can be loaded or unloaded is not considered. If a delivery truck has finished its operations, it stays at its current position and does not return to the cross-dock (although the VRSS assumes that all trucks make complete tours starting and ending at the cross-dock).

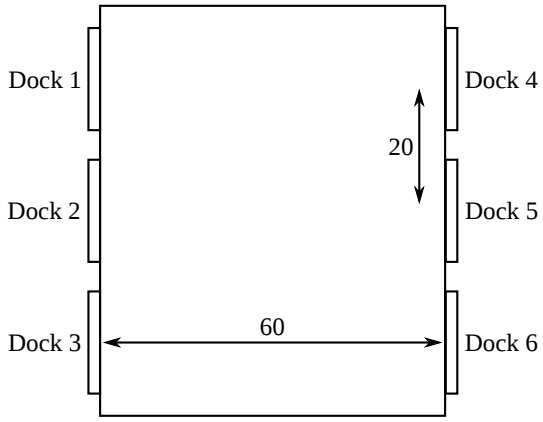


Figure 6.3: Schematic representation of the cross-dock considered in the experiments.

Resource set-up The experiments in Sections 6.3 and 6.4 use one of the following base set-ups for the cross-dock and the available resources.

Set-up 1 The considered cross-dock (CD) has six dock doors (three strip and three stack doors). Figure 6.3 shows a schematic representation of the I-shaped cross-dock indicating the (rectilinear) distances between the dock doors (in m). Inside the cross-dock, two forklift trucks (with driver) are available to process the orders. Temporary storage of these orders is possible. The orders have to be picked up at one of two possible origins (O1 and O2) and delivered to one of two destinations (D1 and D2). The expected travel times between the cross-dock, origins and destinations are given in Table 6.1 (based on straight line distance and expected average speed). There are three trucks available at the cross-dock to perform the transportation, two for freight pick-up at their origins (PT1 and PT2) and one for delivery to their destinations (DT1). The resource graph for this set-up was shown in Figure 5.5.

Set-up 2 This set-up is similar to set-up 1, but now four trucks are available for transporting the orders, two for pick-up (PT1 and PT2) and two for delivery of the goods (DT1 and DT2).

Set-up 3 This set-up is similar to set-up 1, but there is only one truck for pick-up of the orders (PT1) and two trucks for delivery (DT1 and DT2).

The number of pick-up and delivery trucks that are available in the various set-ups are summarized in Table 6.2.

Table 6.1: Expected travel times (min).

	CD	O1	O2	D1	D2
CD	0.0	343.8	277.2	384.4	217.4
O1	343.8	0.0	467.6	384.4	554.4
O2	277.2	467.6	0.0	652.3	384.4
D1	384.4	384.4	652.3	0.0	467.6
D2	217.4	554.4	384.4	467.6	0.0

Table 6.2: The number of pick-up and delivery trucks for the considered set-ups.

Set-up(s)	# pick-up trucks	# delivery trucks
Set-up 1	2	1
Set-up 2	2	2
Set-up 3	1	2

Order set-up The number of orders will vary for the different experiments. The origin and destination of each order is randomly chosen from the available origins and destinations[1]. Each order has a predefined release time and due date. Unless stated otherwise, the release times are randomly sampled between 400 and 600 min and the due dates are randomly chosen between 750 and 950 min after their corresponding release times. For all experiments, the (sampled) origins, destinations, release times and due dates of the orders can be found in Appendix A. For the experiments in Section 6.3, four different sets of eight orders are considered (see Section A.1).

Control modes Three control modes are considered for the experiments.

Control mode 1 The holonic LES controls the logistic operations, without advice from the staff holon.

Control mode 2 The staff holon gives an initial advice to the order holons (from the VRSS) and the cross-dock holon (from the TSS).

Control mode 3 The staff holon gives an initial advice to the order holons and the cross-dock holon, and updates this advice at regular time intervals (every 30 min) based on the latest available information.

[1]Note that situations in which direct transportation is more beneficial are not considered, i.e. if all orders from an origin have the same destination.

Performance measures During the experiments, the following performance measures are recorded.

Average tardiness The average tardiness of all orders (expressed in min). If there are n orders, e_i is the time at which order i is unloaded at its destination and s_i the due date of order i, the average tardiness is defined as $\frac{1}{n}\sum_{i=1}^{n}\max(e_i - s_i, 0)$.

Average flow time The average flow time of all orders (expressed in min). This measure is defined as $\frac{1}{n}\sum_{i=1}^{n}(e_i - r_i)$, in which r_i is the release time of order i.

Makespan The time required to process all orders, i.e. to transport all orders to their destination (expressed in min). This value is equal to $\max_i e_i$.

Total travel distance The total distance travelled by all trucks (expressed in km).

For the replicated tests, the mean values are displayed together with error bars to show the variability of the performance measures over multiple replications. These error bars indicate the standard error and are symmetrically displayed above and below the mean (with a total length of two times the standard error). If $a_1, a_2, \ldots, a_n$ represent the values of n replications, the (sample) mean is defined as:

$$\overline{a} = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{6.1}$$

The standard error (of the mean) is then defined as:

$$SE_{\overline{a}} = \frac{s}{\sqrt{n}} \tag{6.2}$$

In this expression, s represents the (sample) standard deviation:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(a_i - \overline{a})^2} \tag{6.3}$$

Emulation settings The simulation time is expressed in minutes. The simulation starts at time 0 and ends when all orders have arrived at their destination. The behavior of the various resources is described by multimodels. These models are similar to the models used by the intelligent beings, but are stochastic. For the trucks, the speed is a stochastic variable following a normal distribution with a mean of 80 km/h and a standard deviation of 2. If the truck

breaks down, the time to repair is also assumed to be normally distributed with a mean of 180 min and a standard deviation of 6. The forklift trucks have an expected speed of 6 km/h (with a standard deviation of 0.2) and the time to load or unload one pallet is 1 min. In case of a breakdown, the time required to repair the forklift is also a normally distributed variable with a mean of 90 min and a standard deviation of 5).

Control settings The product, resource, order and staff holons behave as described in the previous chapters. Some specific parameters that are applied for the simulation experiments are the following[2].

Exploring interval If an order has an intention, it sends out exploring ants on average every 2 min, otherwise on average every 0.4 min.

Refreshing interval If an order has an intention, it sends out refreshing and intention ants on average every 10 min, otherwise on average every 2 min.

Evaporation Pheromones evaporate after 25 min.

Feasibility ants Feasibility ants are only sent out *once* - at the start of the simulation - to search for feasible paths. This information does not evaporate.

Guidance percentage If the staff holon cooperates with an external scheduling system and gives advice to the order holons, 75 % of the exploring ants try to follow this advice.

Reschedule interval If the staff holon receives updates from the external scheduling system (control mode 3), it will update its advice for the cross-dock holon and the order holons every 30 min.

As explained in Chapter 5, the staff holon also has to provide data to both cooperating scheduling systems (VRSS and TSS). These data corresponds to parameters of the considered vehicle routing and truck scheduling problems. For the VRSS, the value of the transfer time is determined as $T = 10 + 5 * nbr$, in which nbr is the number of orders. The other parameters are fixed and their values are shown in Table 6.3. The values of the TSS parameters are displayed in Table 6.4. If for a simulation experiment some parameter values are different than explained here, the specific values will be mentioned in the corresponding description of the experiment.

[2]These parameters have been determined experimentally and based on experience. Some guidelines on how to calibrate these parameters can be found in [151].

Table 6.3: VRSS parameter values.

Parameter	Value
L	2.5
w_1	1
w_2	5
w_3	1000
α_1	5 %
α_2	5 %

Table 6.4: TSS parameter values.

Parameter	Value
L	4
T	15
w_1	1
w_2	1

Objectives The goal of the simulation experiments is to show that the developed coordination and control system works as expected and to show some of the value added of the holonic technology. More concretely, the objectives of the experiments are to indicate that:

1. the HLES functions as intended (multi-resource allocation, batching, consolidation, etc.);
2. the HLES can cooperate with external scheduling algorithms;
3. the cooperation between the HLES and external scheduling algorithms improves the performance (e.g. because better batching of orders is obtained);
4. the HLES can deal with aspects not taken into account by the external scheduling algorithms;
5. the HLES is robust against disturbances and uncertainty and can respond reactively as well as proactively to disturbing events;
6. the short-term forecasts allow to take better (informed) decisions;
7. the visibility provided by the HLES allows for intervention by the (cross-dock) management.

6.3 Scenario tests

This section describes the simulation experiments executed to test the behavior of the proposed holonic control system for specific scenarios. The results are described and discussed in-depth to illustrate the 'logic' of the system in the different cases. The experiments are grouped according to their main objective. In the next section, an experiment to verify the correct functioning of the software is described. Section 6.3.2 then presents several experiments in which the control system cooperates with the two scheduling systems. In Section 6.3.3, two experiments in which situations occur that cannot be considered by the scheduling system are presented and Section 6.3.4 describes an experiment to show that the HLES can control the logistic operations reactively as well as proactively. To give an overview of the experiments, Table 6.5 provides a summary of the characteristics of all simulation runs. A run is specified by the set-up of the experiment and the applied control system. The set-up is determined by the resource set-up, the order set and possibly an event (e.g. a breakdown). The control system is specified by the applied control mode and possible adaptations to the control behavior (e.g. adapted parameter values or intervention from the cross-dock manager).

6.3.1 Verification

The first experiment is carried out to show that the developed research prototype functions as intended (objective 1). This includes among others that orders are batched for transportation and are consolidated at the cross-dock, that auxiliary drive operations are executed if needed and multi-resource allocation is performed to load and unload the trucks at the cross-dock.

Experiment 1

Set-up The scenario for this experiment represents a simple scenario with normal operating conditions. Set-up 1 is used and eight orders have to be transported (order set 1, see Table A.1). Two simulation runs are carried out. The control mode used for both runs is mode 1.

Results Table 6.6 shows the performance measures of two simulation runs. In the first run, both pick-up trucks visit another origin (PT1 visits O2, PT2 visits O1). PT2 picks up all orders in one time (orders 2, 3, 4 and 8), while PT1 only picks up order 6. PT2 then picks up the remaining orders at O2 (orders 1,

Table 6.5: Overview of the experiments in Section 6.3.

Experiment(s)	Run(s)	Set-up			Control system	
		Resource set-up	Order set	Event	Control mode	Adaptations
Experiment 1	Run 1 / Run 2	1	1		1	
Experiment 2	Run 1	1	1		2	
Experiment 3	Run 1	1	1	Breakdown of pick-up truck	3	Staff holon not informed about expected repair time
	Run 2				3	Staff holon informed about expected repair time
Experiment 4	Run 1	1	1	Extra delivery and forklift truck	2	Advice for orders cleared at $t = 800$
	Run 2				3	
Experiment 5	Run 1	2	2	Traffic jam for pick-up and expected traffic jam for delivery	2	
	Run 2				2	Cross-dock manager intervenes
	Run 3				3	Staff holon is informed about expected traffic jam, $\alpha_1 = 2\,\%$, $w_3 = 500$
Experiment 6	Run 1	3	3	New order	3	
	Run 2				3	Cross-dock manager intervenes
Experiment 7	Run 1	1	1	Breakdown of forklift trucks	3	
Experiment 8	Run 1	1	1	Break for forklift trucks	3	
	Run 2				3	Cross-dock manager intervenes
Experiment 9	Run 1	1	4	Traffic jam for pick-up	2	No position updates
	Run 2				2	Position updates every 30 min

Table 6.6: Performance measures of experiment 1.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	298.3	1054.2	1948.5	3402.8
Run 2	269.5	1081.2	1837.3	2724.1

5 and 7) after it has delivered the orders from O1 at the cross-dock. Shortly after the (first) arrival of both pick-up trucks, the delivery truck DT1 leaves the cross-dock with all orders for destination D2 (orders 2, 3, 4 and 6). After delivery, DT1 returns to the cross-dock to load the remaining orders (orders 1, 5, 7 and 8) and to bring them to D2.

In the second simulation run, both pick-up trucks also travel to another origin, but now all orders are picked up in one time (orders 2, 3, 4 and 8 by PT1, orders 1, 5, 6 and 7 by PT2). After arrival at the cross-dock, all orders are loaded into delivery truck DT1 and this truck delivers all orders by first visiting D1 and then D2 (without returning to the cross-dock in between).

Discussion Figure 6.4 shows a schematic representation of the execution of the first simulation run at time $t = 750$. Truck PT1 has picked up one order at O2, PT2 four orders at O1. Both pick-up trucks are on the way to the cross-dock, where delivery truck DT1 is waiting. The intentions of the orders at the same time are shown in Figure 6.5. These intentions correspond to the current situation (orders 2, 3, 4 and 8 being transported by PT2 and order 6 by PT1), but also indicate the future operations. According to these intentions, PT2 will pick up orders 1, 5 and 7 after delivery of its current orders at the cross-dock. Meanwhile, DT1 will deliver orders 2, 3, 4 and 6 to D2 while order 8 is kept in storage. DT1 will eventually also transport orders 1, 5, 7 and 8 to their destination D1.

Figure 6.5 clearly indicates that the holonic control system is capable of batching and consolidating orders. PT2 first transports orders 2, 3, 4 and 8 together, and then orders 1, 5 and 7. Subsequently, the orders are consolidated at the cross-dock and transported in batch to their destination for delivery. As explained in Section 5.5.2, the local decision rules employed by the truck agents allow for this batching. For the current scenario however, the batching of the orders could be improved if PT1 transports orders 1, 5, 6 and 7 together. An external schedule provided by a staff holon could steer the order agents to find a similar solution.

The second simulation run shows that it is also possible to obtain this good

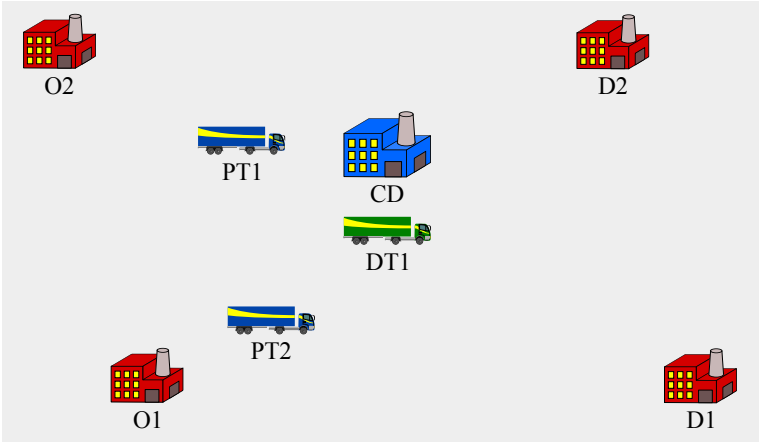


Figure 6.4: Snapshot of the execution at $t = 750$ (experiment 1, run 1).

batching without an external schedule, as now both pick-up trucks succeed in transporting four orders together. Moreover, truck DT1 delivers all orders by making one tour (via D1 to D2). This is reflected in the measured values; the average tardiness, makespan and the total travel distance are lower for simulation run 2 than for run 1. The average flow time is slightly increased as the orders for destination D2 are now longer in transit.

Figure 6.6 demonstrates that the HLES is capable of multi-resource allocation. As Section 5.3 explained, during the loading or unloading of a truck at the cross-dock, three resources have to be allocated at the same time: a forklift truck, a truck and a dock door. In Figure 6.6, the intentions at $t = 750$ of the two forklift trucks, pick-up truck PT2, delivery truck DT1 and two dock doors are shown for simulation run 1. Note that this figure zooms in on the intentions between $t = 920$ and $t = 980$. The intentions of the forklifts clearly show that the orders reserve three consecutive time slots; one for unloading the order, one for transporting the order and one for loading the order[3]. As can be seen, for every unloading slot, there is a corresponding time slot reserved at pick-up truck PT2. Similarly, for every loading slot there is also a slot reserved at DT1. Finally, also the corresponding dock doors are reserved. A time slot spanning the complete unloading duration of PT2 is allocated at dock door 2 and dock door 4 has a slot reserved from start to end of the loading of DT1.

[3]Order 4 has even six slots reserved at forklift truck 2, as the order is (shortly) placed in the temporary storage area between unloading and loading.

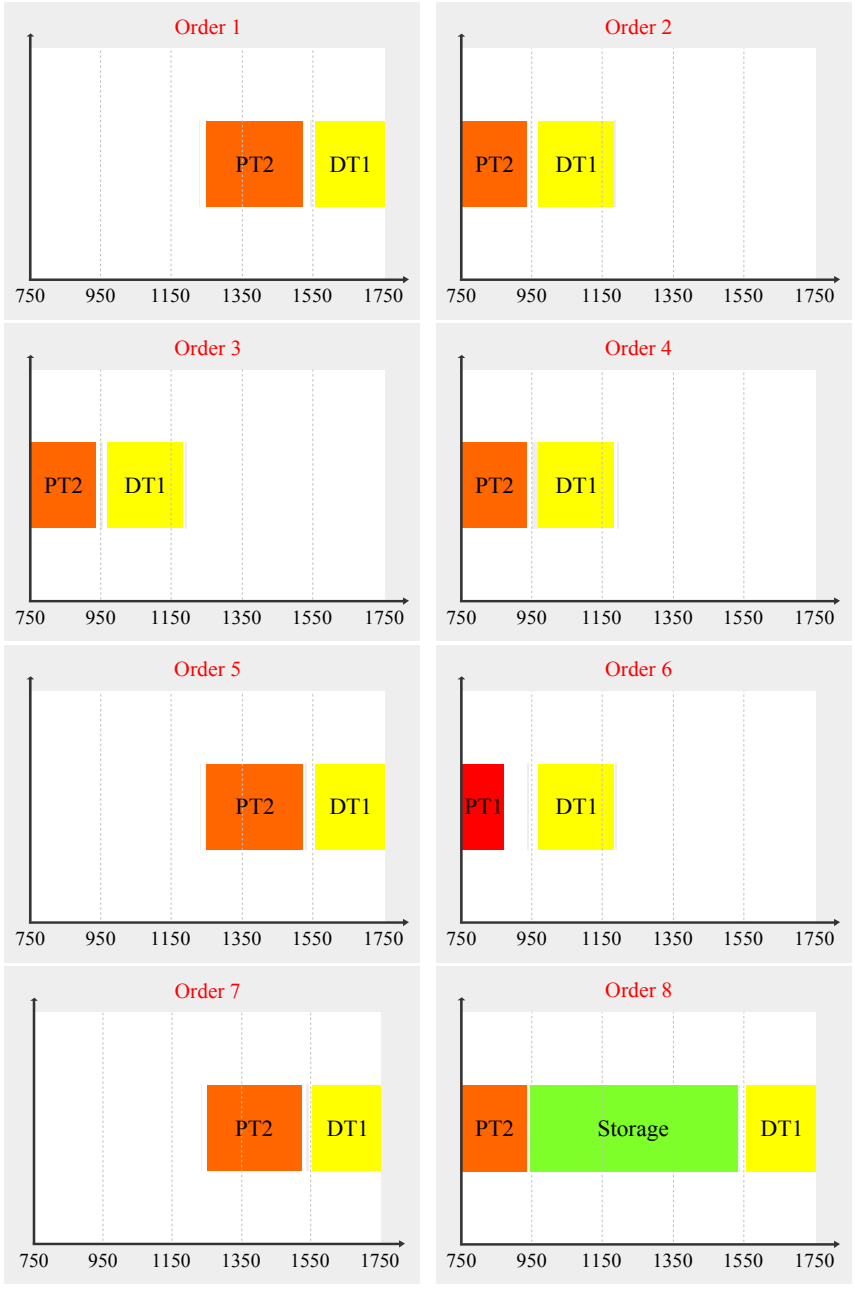


Figure 6.5: The intentions of the orders at $t = 750$ (experiment 1, run 1).

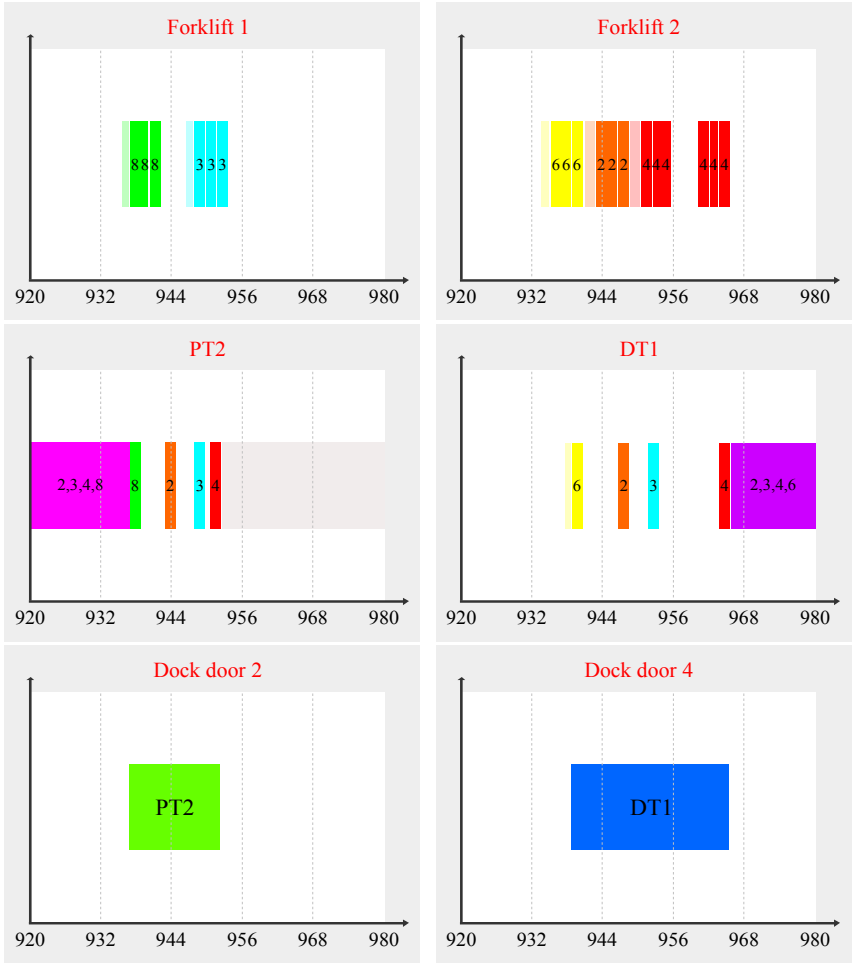


Figure 6.6: The intentions of several resources at $t = 750$ (experiment 1, run 1).

6.3.2 Cooperation with scheduling systems

The following experiments show how the holonic control system can cooperate with two scheduling systems (objective 2). As explained in Section 5.4, the truck scheduling system assists with assigning trucks to dock doors, and the vehicle routing scheduling system with the batching of orders. The latter system tries to minimize the weighted combination of three objectives, including two of the considered performance measures (total tardiness and total travel distance).

Experiment 2

Set-up The same set-up as experiment 1 is used (set-up 1 and order set 1 (see Table A.1)), but control mode 2 organizes the logistic operations. One simulation run is executed.

Results The performance measures of this experiment are shown in Table 6.7. All orders at O1 are picked up in one go by PT1, and all orders at O2 by PT2. When both trucks have arrived at the cross-dock, the eight orders are transferred to DT1 and this truck makes a tour via D2 to D1 to deliver all orders.

Discussion In this experiment, the cross-dock holon and order holons receive an initial advice from the staff holon. Figure 6.7 shows the Gantt charts corresponding to the schedules of the TSS and VRSS. The cross-dock agent takes this advice into account while assigning trucks to dock doors. Correspondingly, PT1 and PT2 are unloaded at dock door 2 and DT1 is loaded at dock door 5. The order agents make use of the advice while searching for new solutions and to select their intention. The intentions of several orders are displayed in Figure 6.8. As can be seen, the order agents are able to reserve intentions that correspond very well to the provided advice (except for a small shift in time (15 min) of the transport operation of PT1). Note that the order agents do not only make reservations for the operations indicated by the provided schedule, but also reserve resources for the necessary operations in between (e.g. the

Table 6.7: Performance measures of experiment 2.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	182.5	953.3	1709.4	2507.1

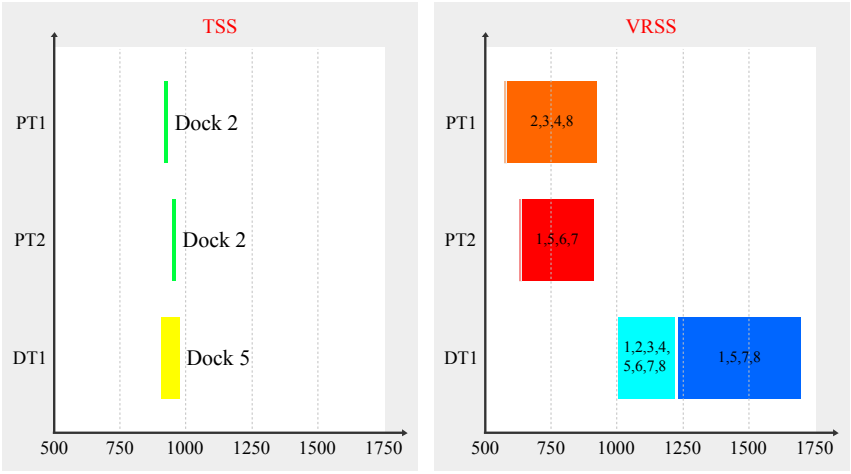


Figure 6.7: Initial advice of the staff holon (experiment 2, run 1).

transport operations inside the cross-dock). In this way, the HLES fills in the gaps of the provided schedule (objective 4).

This experiment not only indicates that the HLES can cooperate with external scheduling systems, but also that this cooperation improves the performance (objective 3). Compared to simulation run 1 of experiment 1, all performance measures have ameliorated (see Table 6.6 and 6.7). This can be explained by a good batching of the orders, as advised by the staff holon. The performance measures also have improved compared to run 2 of experiment 1. Although a similar batching is obtained in both simulation runs, the tour made by delivery truck DT1 is different. In the second simulation run of experiment 1, DT1 first visits D1 before delivering goods at D2. In this experiment, the delivery truck visits the destinations in reverse order. As the truck does not return to the cross-dock and the distance CD-D2-D1 is shorter than CD-D1-D2, this results in a shorter route for DT1. This shortened trajectory explains the reduced makespan and total travel distance. Moreover, the reverse order of the tour also leads to an (on average) earlier delivery of the goods, explaining the improved average tardiness and flow time.

Figure 6.8: The intentions of several orders at $t = 500$ (experiment 2, run 1).

Experiment 3

Set-up The set-up is the same as experiment 1 (set-up 1 and order set 1 (see Table A.1)), but with the addition of a breakdown. One of the pick-up trucks breaks down when it is transporting orders to the cross-dock. Two simulation runs are carried out. The control mode used for both runs is now mode 3, so the staff holon recalculates its advice based on the current situation (i.a. truck positions). In the first simulation run, the staff holon is only informed about the current positions of the trucks and has no clue about the estimated repair time. In the second run, the staff holon is also informed about the estimated arrival time of PT2 at the cross-dock as determined by the truck agent (which takes the expected repair time into account).

Results The execution of both simulation runs is similar. PT2 breaks down at time $t = 772.0$, when it is moving the orders originating from O1 to the cross-dock. The truck is repaired at $t = 951.2$ and continues its trip to the cross-dock. At that time, PT1 has already arrived at the cross-dock. The loading of delivery truck DT1 only starts once PT2 has arrived at the cross-dock

Table 6.8: Performance measures of experiment 3.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	337.1	1151.0	1909.4	2507.5
Run 2	323.1	1136.4	1891.3	2507.1

and all orders are delivered in one tour (via D2 to D1). The corresponding performance measures are shown in Table 6.8.

Discussion In both simulation runs, the staff holon provides the order agents with a similar initial advice as in experiment 2 (see Figure 6.7). The orders are executing this schedule when PT2 breaks down. As soon as the truck agent corresponding to PT2 is informed about the breakdown, it will adapt its intention based on the expected repair time (specified in the model of the corresponding intelligent being). When the affected orders refresh their intention, they will be informed about the delayed arrival time at the cross-dock. As the delay makes their current intention infeasible, the order agents abandon it and select a new intention. Eventually, all orders will have found a new solution. The collection of these solutions is similar to the original schedule, but the delivery tour is delayed. This explains the deterioration of the performance measures compared to experiment 2.

The order agents make use of the advice of the staff holon when searching for new solutions. As control mode 3 is used, this advice is updated at regular times and will change significantly once PT2 breaks down. In simulation run 1, this updated advice is based on the current positions of the trucks, which determine the expected arrival times. As the position of PT2 does not change as long as it is broken down, the expected arrival time is shifted forward and causes the departure time of DT1 in the resulting VRPCD schedule to be shifted forward with the same amount. Figure 6.9 shows the advice of the staff holon at time $t = 1000$, when the truck has resumed its transport operation to the cross-dock[4]. However, the expected arrival time is shifted at every update, also shifting DT1's departure time. These variations in advice of the staff holon cause the order agents to change their intentions multiple times.

In the second simulation run, the staff holon is directly informed about the estimated arrival time of PT2 at the cross-dock (with the repair time taken into account by the truck agent). This causes a major shift of DT1's departure time in the resulting VRPCD schedule, right away at the first update of the schedule.

[4]There is no advice for PT1 as this truck has already arrived at the cross-dock.

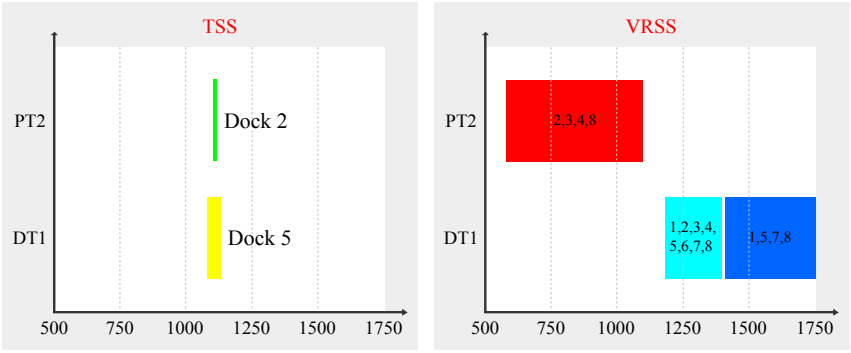


Figure 6.9: Advice of the staff holon at $t = 1000$ (experiment 3, run 1).

But from then on, the schedule remains stable over time. As a consequence, the number of times the order agents change their intention has decreased. The performance measures for both runs are however similar.

Experiment 4

Set-up The same set-up as in experiment 1 is used (set-up 1 and order set 1 (see Table A.1)), but an extra delivery truck (DT2) and an extra forklift truck become available at a random time between $t = 600$ and 800. Two simulation runs are executed. For run 1, control mode 2 is applied. Additionally, the advice for the orders is cleared at time $t = 800$. The control mode used for the second simulation run is mode 3.

Results For both simulation runs, the two pick-up trucks visit another origin to collect all orders (PT1 visits O1, PT2 visits O2). Once arrived at the cross-dock, the orders are transferred to the two delivery trucks and consolidated based on their destination. For this transfer, also the extra forklift truck that becomes available at $t = 663.6$ is used. Subsequently, DT1 delivers the orders at D2 and DT2, which becomes available at $t = 692.4$, transports the other orders to D1. This similar behavior of both simulation runs is also reflected in similar values of the performance measures (see Table 6.9).

Discussion In both simulation runs, the eight orders easily find a solution corresponding to the initial VRPCD schedule provided by the staff holon (see Figure 6.10a). The orders are being transported to the cross-dock when the

Table 6.9: Performance measures of experiment 4.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	45.3	815.8	1434.5	2398.9
Run 2	36.3	806.8	1419.2	2398.9

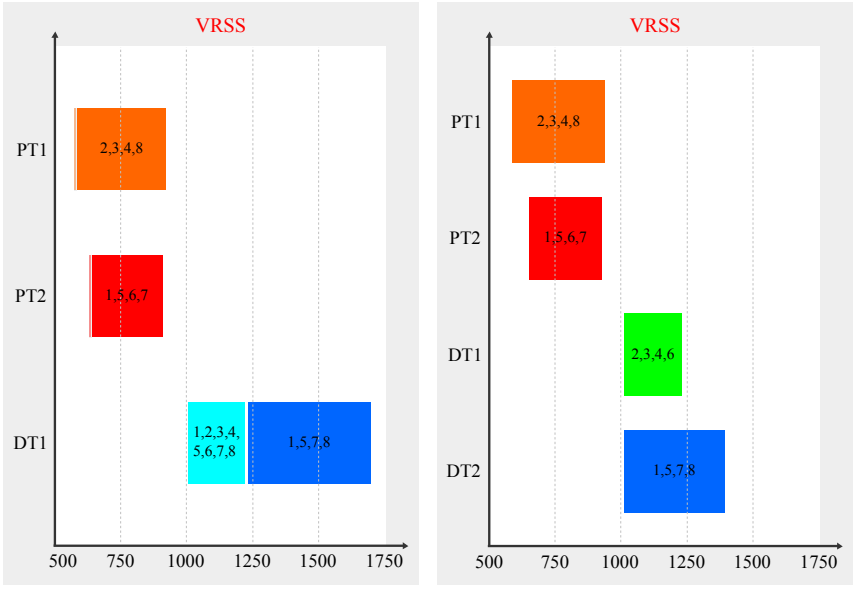
extra resources become available. From then on, the order agents will also discover solutions making use of DT2 to be delivered at their destination.

In the first simulation run, the order agents will not immediately select an intention which makes use of the newly available delivery truck. Indeed, they prefer solutions that correspond to the initial VRPCD schedule which does not consider DT2. Only if two solutions have a similar accordance with the original schedule, other criteria are taken into account (see Section 5.5.3). The order agents can however directly choose intentions using the newly available forklift truck for the internal cross-dock operations, as no advice is given for these operations. Once the advice for the order agents is cleared (at $t = 800$), the orders will consider tardiness as first criterion instead of similarity to the schedule. This allows orders 1, 5, 7 and 8 to switch their intention to a solution in which they are directly transported to D1 by DT2. Because of the use of this extra truck, all performance measures have improved compared to experiment 2 (see Tables 6.7 and 6.9). But, due to this extra truck, the total cost is increased.

Note that this simulation run also indicates that the short-term forecasts allow to take better (informed) decisions (objective 6). Because of these forecasts, the order agents are able to find alternative solutions which predict their future execution in a reliable way. Based on these predictions, they can then make a well-informed selection between those alternatives.

In simulation run 2, the VRPCD schedule is regularly updated, and when DT2 becomes available, the staff holon advises the orders to make use of this delivery truck (see Figure 6.10b). As the order agents prefer an intention corresponding to the provided schedule, the affected orders will eventually adapt their intention and make use of DT2 to be transported to destination D1. As Table 6.9 shows, this results in very similar performance measures as simulation run 1. Also in this run the total cost is increased because of the use of the extra truck, but this is - according to the staff holon - compensated by the lower tardiness of several orders.

This experiment also indicates that the control system functions as intended (objective 1). The only change to the software required to make use of the two newly available resources is the addition of corresponding resource holons. No



(a) $t = 500$ (b) $t = 750$

Figure 6.10: Advice of the staff holon for the order holons (experiment 4, run 2).

changes have to be made to other resource holons or to order or product holons. This is a result of the active role of the order agents, who search for the services they need and continuously seek alternatives. Their behavior is not changed, for them just the number of service providers is increased. In this case, also the staff holon can remain unchanged, as the VRPCD algorithm can be used for any number of (homogenous) trucks. In other cases however, changes to the underlying infrastructure require adaptations of the scheduling algorithm and make the schedule obtained without these adaptations suboptimal or even infeasible (e.g. if the added truck has another capacity).

Experiment 5

Set-up As base set-up, set-up 2 is used. There are eight orders, two for each origin-destination pair (order set 2). The due dates are randomly chosen between 750 and 950 min after the release time, except for the orders with D1 as destination. For these orders, the due dates are adapted to become stricter for the orders originating from O2 and less strict for the orders from O1. Table A.2 shows the origins, destinations, release times and due dates of the eight orders.

One of the pick-up trucks encounters a traffic jam on its way to the cross-dock. As a consequence, its average speed decreases to 40 km/h for about 150 min. Around $t = 800$, the control system is informed about an expected traffic jam on the route to D1. From $t = 1050$ on, the expected average speed decreases to 50 km/h.

Three simulation runs are executed. For run 1, control mode 2 is applied. This mode is also used in simulation run 2, but in this run the cross-dock manager intervenes and decides that the truck for D1 has to leave earlier in order to avoid (partially) the delay that will be caused by the expected traffic jam. In the third simulation run, control mode 3 is applied. The values of two VRSS parameters are changed for this run: α_1 is set to 2 % and the truck cost w_3 is set to 500. The VRSS is also informed about the expected traffic jam and adapts the travel times correspondingly.

Results The performance measures of the three simulation runs are shown in Table 6.10. For all runs, the orders at O1 are picked up by PT1 and the orders at O2 by PT2. During its trip to the cross-dock, PT1 is delayed by the heavy traffic and arrives later than originally expected at the cross-dock.

In the first simulation run, both delivery trucks do not depart before the delayed pick-up truck has arrived. All orders are then transferred to the delivery trucks and consolidated by destination. Subsequently, DT1 delivers the orders at D1 and DT2 at D2.

There is an intervention of the cross-dock manager in the second simulation run. The manager decides at about $t = 830$ that DT1 does not have to wait for the arrival of PT1 and can leave at time $t = 960$. At that time, only orders 5 and 6 are loaded and these orders are delivered by DT1. The other orders are in two times delivered at their destination by DT2. The orders for D2 are first delivered. Then DT2 returns to the cross-dock to pick up the remaining two orders and to deliver them at D1.

In simulation run 3, DT1 also does not wait for the arrival of PT1 and leaves the cross-dock with orders 5 and 6 at time $t = 973$. All other orders are delivered at their destination by DT2, which makes a tour via D2 to D1.

Discussion For the three simulation runs, the staff holon gives the initial schedule shown in Figure 6.11 as advice to the orders. The eight orders easily reserve an intention corresponding to this advice. However, as PT1 is delayed by the traffic jam, the orders will have to adapt their intentions.

Table 6.10: Performance measures of experiment 5.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	110.0	962.2	1678.3	2398.9
Run 2	94.7	977.4	1894.8	3182.1
Run 3	85.0	960.0	1766.6	3007.3

In the first simulation run, the orders react by postponing the departure times of both delivery trucks. The composed solution is similar to the schedule provided by the staff holon, but shifted in time. Figure 6.12 shows the adapted reservations of the four trucks. Once the truck agent corresponding to DT1 is informed about the expected traffic jam on its route to D1, it will extend its reserved time slot (see Figure 6.13). In this way, the (expected) effect becomes immediately visible. The order agents do not change their intention however. The average tardiness of the orders is equal to 110.0 min. However, only orders 5 and 6 contribute to this value. These two orders are very tardy (372.9 and 507.4 min respectively), while all other orders arrive on time at their destination.

Because the effect of the expected traffic jam has become visible, it would be possible for the cross-dock manager to intervene beforehand. As the two orders with the earlier due date (orders 5 and 6) will be very tardy, the manager can decide that DT1 does not have to wait for the orders picked up by the delayed PT1, but has to depart at $t = 960$. This departure time can for instance be determined in what-if mode. This decision is implied in simulation run 2 by shifting the currently reserved transportation slot of DT1 backward in time. As a consequence, the affected orders have to adapt their intention. Only orders 5 and 6 will be able to reserve again this shifted time slot, while orders 1 and 2 find a new solution in which they are transported to their delivery by DT2 (see Figure 6.14). The effect of the manager's decision on the performance measures can be seen in Table 6.10. The total travel distance is increased because of the extra trip DT2 has to make and this also increases the makespan. The average flow time is also slightly increased. The flow time of orders 5 and 6 is reduced, but this reduction is completely compensated by the increase of flow time of orders 1 and 2. The tardiness of orders 5 and 6 is also reduced, but now orders 1 and 2 are tardy. However, the reduction in tardiness is larger than the increase, so the average tardiness is reduced. This simulation run indicates that the visibility provided by the HLES allows the cross-dock manager to intervene on time and to adjust the (local) decision making if necessary (objective 7).

In the third simulation run, the staff holon makes use of rescheduling. The VRSS comes up with a different schedule once it is informed about the expected

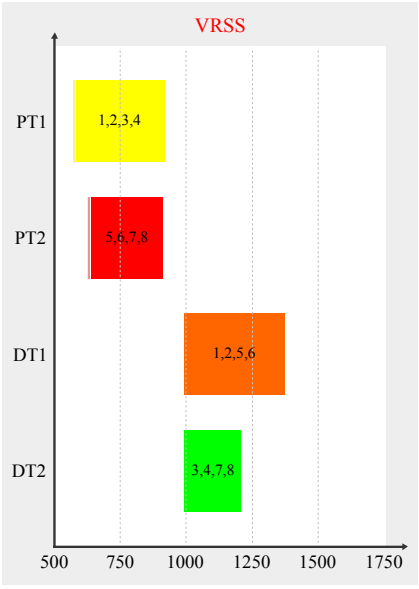


Figure 6.11: Initial advice of the staff holon (experiment 5).

traffic jam (and has adapted the travel time correspondingly). In this schedule, DT1 only transports orders 5 and 6 and DT2 delivers all other orders by making a tour via D2 to D1 (see Figure 6.15). The orders adapt their solutions according to this updated VRPCD schedule and the logistic operations are executed according to these intentions. This simulation run indicates that the (continuous) cooperation between the HLES and the VRSS works well and can improve the performance (objective 3). Compared to simulation run 1, the makespan and total travel distance are increased, but this is - according to the staff holon - compensated by the decrease (of 25 min) of the average tardiness. Compared to simulation run 2, the total travel distance and makespan are reduced as DT2 now directly moves from D2 to D1 without returning to the cross-dock. The tardiness and flow time of orders 5 and 6 is increased a little as DT1 leaves the cross-dock later than in run 2, but this is compensated by the earlier arrival of orders 1 and 2 and leads to an improved average tardiness and average flow time.

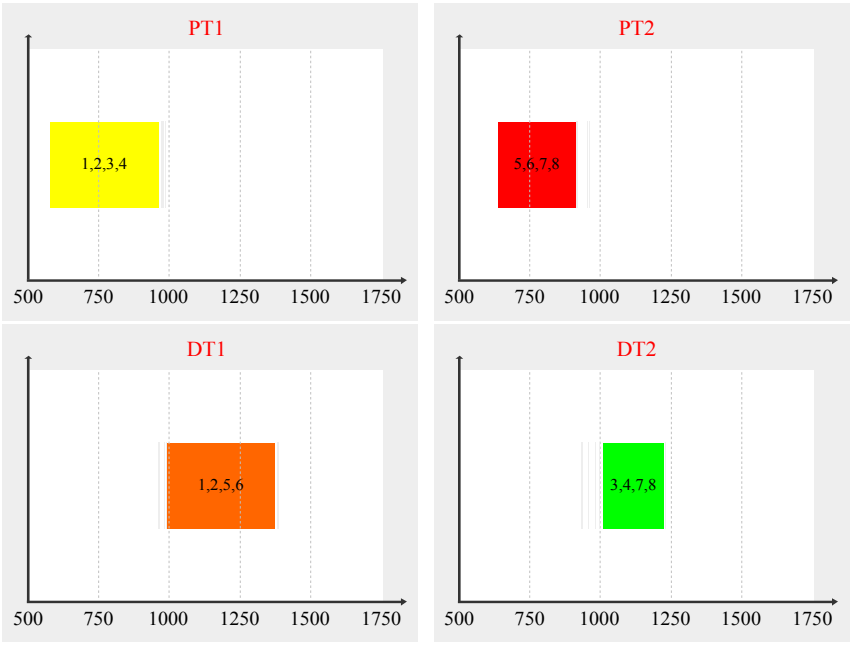


Figure 6.12: The intentions of the trucks at $t = 750$ (experiment 5, run 1).

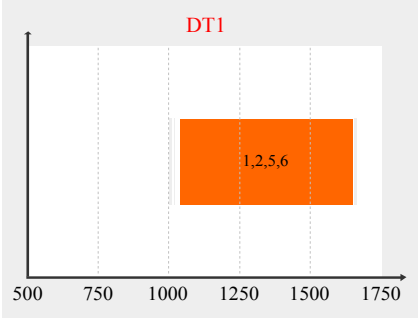


Figure 6.13: The intentions of delivery truck DT1 at $t = 1000$ (experiment 5, run 1).

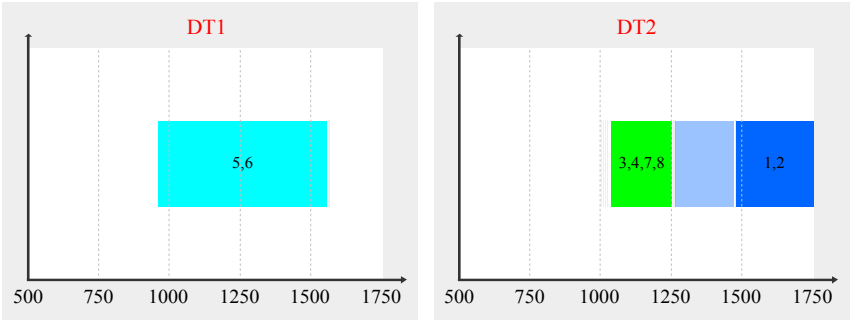


Figure 6.14: The intentions of the delivery trucks at $t = 1000$ (experiment 5, run 2).

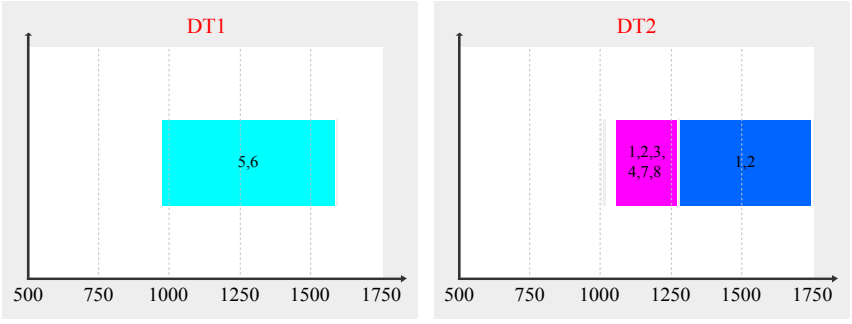


Figure 6.15: The intentions of the delivery trucks at $t = 1000$ (experiment 5, run 3).

Experiment 6

Set-up Set-up 3 is used and eight orders have to be transported (order set 3, see Table A.3). At $t = 450$, the control system is informed about a new order (order 9) that has to be transported from O1 to D1. It will be released at $t = 700$ (later than all other orders) and is due at its destination at $t = 3000$. Two simulation runs are carried out. For both runs, control mode 3 is applied. In the second run however, the cross-dock manager intervenes in the decision making and decides that PT1 does not have to wait at O1 for the release of the new order.

Results The performance measures of both simulation runs are shown in Table 6.11. In the first simulation run, PT1 picks up all nine orders by making

Table 6.11: Performance measures of experiment 6.

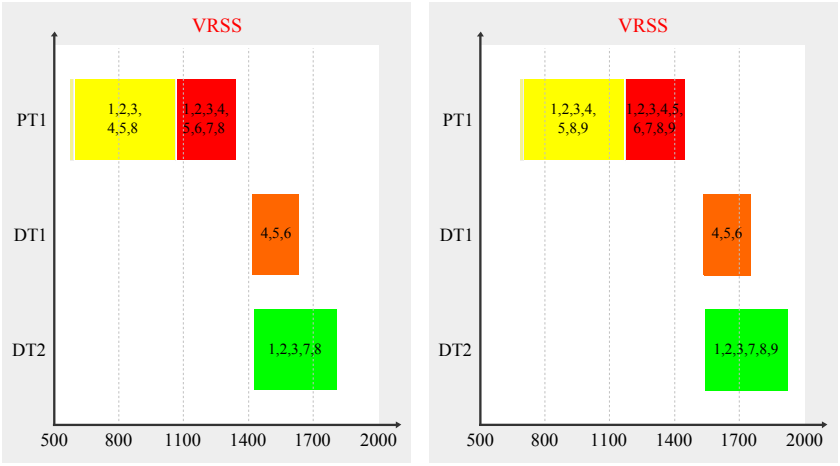
Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	476.2	1342.6	1937.2	2199.2
Run 2	406.2	1349.8	2621.7	3876.8

a tour from O1 via O2 to the cross-dock. At O1, the truck has to wait for about 125 min until the new order is released. When PT1 eventually arrives at the cross-dock, all orders are transferred to the available delivery trucks and are transported to their destination. DT2 delivers all orders for D1 and DT1 all orders for D2.

In simulation run 2, PT1 makes the same tour, but does not wait at O1 for the release of the new order. After its arrival at the cross-dock, the eight picked up orders are transferred to the delivery trucks and the orders with D1 as destination are transported by DT2, the other orders by DT1. When PT1 is unloaded, the pick-up truck returns to O1 to pick up order 9 and to transport it to the cross-dock. In the meanwhile, DT1 has delivered its orders and is returned to the cross-dock. Order 9 is then transferred to DT1 which delivers it at its destination.

Discussion For both simulation runs, the initial advice of the staff holon is to first pick up the orders at O1 and then the orders at O2 (see Figure 6.16a). All order agents quickly find an intention corresponding to this advice. When the control system is informed about the new order, PT1 is on its way to O1. The staff agent will adapt its advice for the orders. As shown in Figure 6.16b, the staff holon suggests that PT1 waits at O1 until order 9 is released before continuing its tour to O2 and the cross-dock. As the order agents prefer intentions corresponding to the given advice (which is assumed to have a good global performance), they will adapt their intentions based on this new advice. In the first simulation run, the logistic operations are executed according to these intentions and this results in the performance measures shown in Table 6.11.

In the second simulation run, the cross-dock manager intervenes at $t = 520$. As the new order does not have a strict due date, it may not be necessary for PT1 to wait for the release of this order. Indeed, it can be more interesting to first transport the other orders - which have an earlier due date - to the cross-dock. So, the manager decides that PT1 should leave O1 at $t = 620$ and continue its tour. This decision is implied by shifting the currently reserved transportation slot of PT1 backward in time. As a consequence, the orders have to adapt



(a) $t = 400$ (b) $t = 500$

Figure 6.16: Advice of the staff holon for the order holons (experiment 6, run 1).

their intention. All orders originating from O1 are able to reserve again this shifted time slot, except for order 9. The new order will eventually reserve a new time slot in order to be transported directly to the cross-dock after PT1 has completed its first pick-up tour (see Figure 6.17). The effect of the manager's decision on the performance measures can be seen in Table 6.11. As PT1 and DT1 have to make an extra trip, the total travel distance and the makespan are increased compared with the first simulation run. The average flow time is similar in both simulation runs. In the second run, the flow time of orders 1 to 8 has decreased with about 80 min, but this is completely compensated by the large increase in flow time of order 9. The average tardiness of the orders is however largely reduced as order 9 is not tardy, despite its late arrival time at D1. This simulation run also indicates that the cross-dock manager can intervene on time in the decision making if the predicted situation is not satisfactory (objective 7). Although the forecasted situation was in this case in correspondence with the provided schedule, the manager can still intervene and try to adjust the decision making. The effect of the intervention becomes visible again, as the order agents will adapt their intentions based on the modifications.

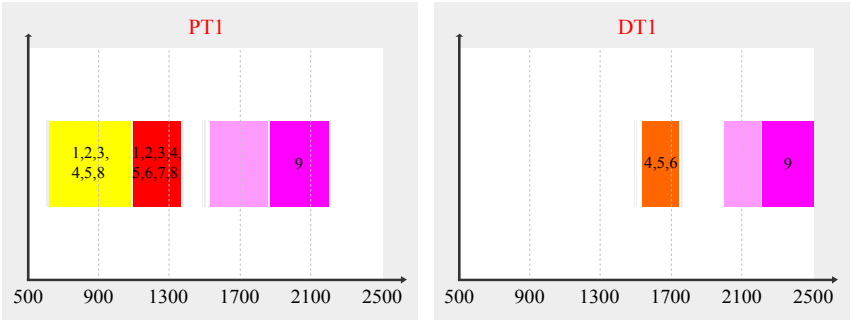


Figure 6.17: The intentions of PT1 and DT1 at $t = 600$ (experiment 6, run 2).

6.3.3 Scheduling abstraction

To generate a schedule, an abstraction is made of the real world problem. It is in general not possible to consider all aspects of the problem (as development and/or calculation time are limited), so simplifications are introduced and some details are not considered. This is also the case for the truck schedule and vehicle routing algorithms used by the cooperating scheduling systems. This section presents two experiments in which the scheduling system cannot react to a disturbance, as this disturbance affects resources that are not explicitly considered in the scheduling abstraction. These experiments indicate how the self-organizing control system can account for these simplified or omitted aspects (objective 4).

Experiment 7

Set-up The same set-up as experiment 1 is used (set-up 1 and order set 1 (see Table A.1)), but both forklift trucks break down during their operations (both at a different random time between $t = 900$ and 950). To control the logistic operations, control mode 3 is applied.

Results The performance measures of this experiment are shown in Table 6.12. All orders are picked up at once by PT1 and PT2. Before the unloading of both trucks can start, the forklift trucks inside the cross-dock break down. When they are repaired, they start transferring the goods to DT1. This truck then delivers all orders at their destination, first at D2 and then at D1.

Table 6.12: Performance measures of experiment 7.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	216.0	999.6	1757.0	2507.3

Discussion The staff holon gives a similar initial advice to the orders as in experiment 2 (see Figure 6.7). The eight orders are quickly able to reserve a solution corresponding to this advice. However, as the two forklift trucks break down, the orders will have to adapt their intentions. They find new solutions by postponing the departure time of delivery truck DT1. The later departure of DT1 is reflected in the performance measures. Compared to experiment 2 (see Table 6.7), the average tardiness, average flow time and makespan all have increased. The increase is however smaller than the time required to repair the forklift trucks. The original schedule provided by the staff holon includes some margin (as the assumed transfer time is generous), which is now consumed to reduce the effect of the breakdowns.

During their search for new solutions, the order agents are not supported by the advice of the staff holon (at least not by the timing information[5]). Despite the fact that this advice is updated, it is not changed as the forklift trucks are not explicitly considered by the scheduling algorithm. The VRPCD algorithm only takes a fixed transfer time into account, assuming that there is always personnel and equipment available to perform the necessary unloading, transferring and loading operations. It could be possible to adapt the VRPCD algorithm to take this aspect into account, but this will require extra development effort. The amount of extra effort will depend on which part of the development workflow has to be done all over again. For instance, for the VRPCD algorithm described in Chapter 4, if only the neighborhood should be adapted, the effort is limited. If however also the solution representation should be adapted, the heuristic method should be redeveloped from scratch. Moreover, as the algorithm becomes more complex, possibly also the calculation time increases. So, while it is possible to extend the scheduling algorithm to include extra aspects, this effort should be balanced with the performance loss without this extension. For this experiment, the order agents deal very well with the situation, so an adaptation of the VRPCD algorithm is not necessary. In any case, the effect of the breakdowns would become visible and this permits the cross-dock manager to take action if required.

[5]The VRPCD schedule contains allocation information (which vehicle(s) will transport the order) and timing information (when will these transport operations take place).

Table 6.13: Performance measures of experiment 8.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	212.9	995.0	1755.0	2507.0
Run 2	214.5	997.6	1755.0	2507.0

Experiment 8

Set-up Similar to the previous experiment, the set-up is the same as experiment 1, but now a break is introduced for (the drivers of) the two forklift trucks. Both forklift trucks are unavailable for one hour (from $t = 930$ until 990). Note that, in contrast to the previous experiment, this unavailability of the forklift trucks is known beforehand. Two simulation runs are carried out. For run 1, control mode 3 is applied. This mode is also used in simulation run 2, but now there is an intervention of the cross-dock manager who decides that PT1 has to depart 30 min later from its pick-up origin (e.g. to refuel in the meantime).

Results Table 6.13 displays he performance measures of this experiment. In the first simulation run, all orders are picked up at once by PT1 and PT2. Both trucks arrive around $t = 920$ at the cross-dock and have to wait until the end of the break before they can be transferred to DT1. This truck then makes a tour via D2 to D1 to deliver all orders. In the second run, the execution is similar but PT1 waits for about 30 min at O1 before departing to the cross-dock (so the truck could have been refueled in the meantime). DT1 leaves the cross-dock however at more or less the same time as in simulation run 1, explaining the very similar performance measures for both runs.

Discussion For the two simulation runs, the initial advice for the orders is again similar as in experiment 2 (see Figure 6.7). This advice is however not feasible because of the planned break of the forklift trucks. As the forklift agents know from the start of the simulation that this period is not available, they do not allow the order agents to make reservations in this period. So, the orders will have to deviate from the schedule and will find new solutions by postponing the departure time of DT1.

In simulation run 1, the orders reserve intentions as shown in Figure 6.18. The delivery truck leaves the cross-dock about 40 min later than indicated by the VRPCD schedule, so about 20 min from the 60 min break are recovered (because

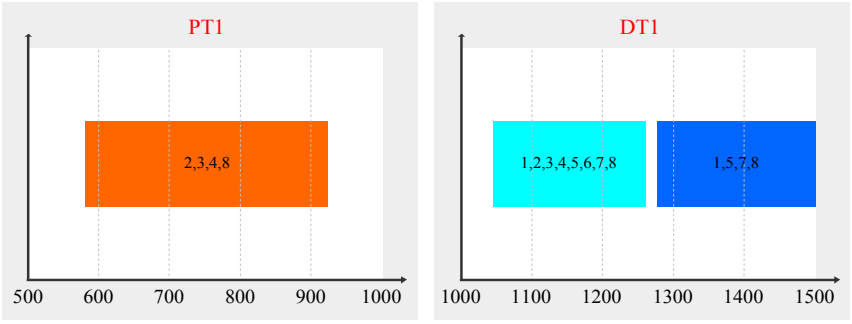


Figure 6.18: The intentions of PT1 and DT1 at $t = 550$ (experiment 8, run 1).

the transfer time used by the scheduling algorithm is overestimated). As a result, the average tardiness, average flow time and makespan all have increased with 30 to 45 min (see Tables 6.7 and 6.13).

In the second simulation run, the cross-dock manager intervenes in the decision making. Based on the short-term forecasts, the manager can see that PT1 has to wait for about 50 min before the unloading starts and can decide that PT1 should depart later. In this way, the driver can take some rest or the truck can be refueled. In run 2, it is assumed that the manager decides at time $t = 400$ to postpone the transport operation of PT1 with 30 min. This decision is implied by shifting the currently reserved transportation slot of DT1 forward in time. Subsequently, the affected orders adapt their intention. As shown in Figure 6.19, they are able to change their intention without postponing the departure time of DT1. As a result, a very similar performance is obtained in run 2 as in simulation run 1. This experiment indicates again that the provided short-term forecasts allow the responsible decision makers to see problems and opportunities in advance, and allow them to react if needed or desired (objective 7).

This experiment also indicates that the HLES can take situations into account that are not considered by the scheduling algorithms. Similar to experiment 7, the VRPCD algorithm can possibly be adapted to consider breaks (e.g. by making the transfer time a time-dependent variable), but this will require extra development effort.

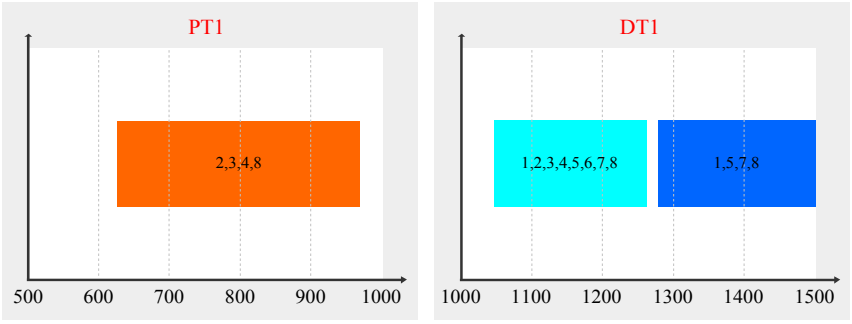


Figure 6.19: The intentions of PT1 and DT1 at $t = 550$ (experiment 8, run 2).

6.3.4 Reactive and proactive control

The following experiment indicates that the control system can work reactively as well as proactively (objective 5).

Experiment 9

Set-up For this experiment, set-up 1 is used and eight orders have to be transported (order set 4, see Table A.4). One of the pick-up trucks encounters a traffic jam on its way from O1 to the cross-dock. As a consequence, its average speed decreases to 20 km/h for about 150 min. To control the logistic operations, control mode 2 is applied. In the first simulation run, the truck agents do not receive position updates from their real-world counterpart. In the second run, the trucks send position updates every 30 min (while driving).

Results In both simulation runs, all orders are picked up at once by the two pick-up trucks and are delivered at the cross-dock. PT2 picks up the orders at O1 and arrives with a delay of about 100 min at the cross-dock because of the heavy traffic. The orders are then loaded into DT1 to be delivered at their destination. In the first simulation run, DT1 leaves the cross-dock with only seven out of the eight orders loaded. It delivers these orders by making a tour via D2 to D1. Then the delivery truck returns to the cross-dock to pick-up the remaining order and to deliver it at D1. In the second run, all orders are immediately loaded into DT1 and delivered at their destination by making the same tour. This is reflected in improved performance measures for simulation run 2 compared to run 1 (see Table 6.14).

Table 6.14: Performance measures of experiment 9.

Run(s)	Average tardiness	Average flow time	Makespan	Total travel distance
Run 1	365.9	1153.7	2593.0	3507.4
Run 2	260.4	1044.3	1807.3	2507.3

Discussion In both simulation runs, all orders quickly find a solution corresponding to the VRPCD schedule provided by the staff holon. Because of the traffic jam encountered by PT2, this schedule cannot be executed without adaptations. In the first simulation run, the order and truck agents only notice that there is a delay at the time when PT2 should have arrived at the cross-dock according to its intention. The orders transported by PT2 then have to abandon their solution and send out exploring ants to search for new solutions. However, as the order agents have no clue about their arrival time at the cross-dock, they assume to arrive soon. Consequently, the discovered solutions are quickly outdated and the order agents will adapt their intention many times. Only once PT2 has arrived at the cross-dock, the orders can make more reliable reservations. At that moment, not much time is left before the scheduled departure time of DT1 and not all orders are able to make the necessary reservations in this short period.

In the second simulation run, the truck agent recalculates its expected arrival time every time it receives a position update. In this way, the truck agent of PT2 notices already during the traffic jam that the current reserved time slot becomes invalid. As a result, the truck agent will extend the reservation until the expected arrival time. The order agents will see this change and adapt their intentions (by also postponing the departure time of DT1). However, as the truck agent only receives position updates but is not informed about the length of the traffic jam, it will underestimate the delay. So, eventually, the reserved slot will have to be extended again and the order agents will also have to change their current intention. This will continue until the truck is again in free flow traffic and the arrival time estimation of the truck agent is more accurate. At that moment, there is still enough time for the order agents to find a good solution (in which all orders are delivered by one tour of DT1). This solution is similar to the provided advice, but shifted forward in time.

This experiment indicates that the HLES can react proactively and that this can improve the performance of the performed control activities. This proactiveness is not only possible because of the position updates, but also because of the presence of the intelligent resources and the delegate MAS pattern. The intelligent resources use their internal model to predict the effect of the update

or disturbance for the current resource. By using the delegate MAS pattern, this effect can then be propagated to other resources.

In both simulation runs, the departure time of delivery truck DT1 is postponed. This is a result of the local rules that are employed by the truck agents. As these agents have an up to date view about the orders they transport (at least about the expected arrival times of the orders), they can make informed (and probably better) decisions when changes occur (objective 6), e.g. about whether or not to wait for a belated order. In the current prototype, the truck agents allow to shift a reserved transport slot forward in time, at least if this shift is not too large. However, the number of times a slot can be shifted is currently not limited, so an extended time shift is still possible. The local rules could however be adapted to take this into account.

The global effect of shifting the departure time of DT1 is not considered in this experiment, although the effect becomes visible and allows the cross-dock manager to intervene and to overrule some local rules if desired. Of course, to take the global performance into account, the staff holon could make use of rescheduling (control mode 3) to react to the encountered traffic jam. For the rescheduling, the staff holon can use the expected travel times as forecasted by the truck agents.

6.4 Replicated tests

This section describes the replicated simulation experiments executed to show the benefits of the integration between the holonic control system and the proposed scheduling approaches. Replicated tests allow to show the overall performance, despite the stochastic nature of the control system. The three different control modes are used for various scenarios and their results are compared. The considered scenarios are constructed by varying the following two factors:

- the workload, i.e. the number of orders that have to be transported. Two values will be distinguished: *8 orders* and *16 orders*.
- the occurrence of a disturbance (the breakdown of a pick-up truck). There are three values: *no breakdown*, a *short breakdown* of one of the pick-up trucks during its trip to the cross-dock (with orders loaded), and a *long breakdown* of one of the pick-up trucks during its trip from the cross-dock to an origin (with no orders loaded yet).

For each scenario, ten replications are simulated (with the same resource and order set-up) so that the stochasticity of the control system is averaged out.

For each replication, average tardiness, average flow time, makespan and total travel distance are measured.

The next section describes experiments with 8 orders. In total, 15 scenarios are executed. First, five scenarios without a truck breakdown are considered. Then, the same five scenarios are considered, but with the addition of a short breakdown of one of the pick-up trucks heading for the cross-dock. For scenarios 11 to 15, this short breakdown is replaced by a long breakdown when the truck has departed from the cross-dock. The section thereafter discusses the results of the simulation experiments with 16 orders. Again, 15 scenarios are considered: five scenarios without a breakdown, five scenarios with the occurrence of a short breakdown and five scenarios with the occurrence of a long breakdown.

6.4.1 Eight orders

In a first series of scenarios, 8 orders have to be transported to their destination. In total, 15 different scenarios are considered: five without a breakdown, five with a short truck breakdown and five with a long breakdown.

Set-up For scenarios 1 to 5, set-up 1 is used as base set-up. For each of these scenarios, 8 orders are processed with randomly chosen origins, destinations, release times and due dates. The sampled values are shown in Tables A.5 to A.9 (Appendix A). Scenarios 6 to 10 are then similar to the first five scenarios, but a truck breakdown is added. For each scenario, one of the two pick-up trucks breaks down at a random time between 600 and 700 min, when the truck is transporting orders to the cross-dock. The breakdown is defined based on the origin of the truck, and not on the truck itself, i.e. the pick-up truck originating from that origin breaks down, not a predefined pick-up truck. The time to repair is normally distributed with a mean of 180 min and a standard deviation of 6. For scenarios 11 to 15, a truck breakdown between 350 and 450 min is added compared to the first five scenarios. At that time, the truck is on its way to one of the origins and has no orders loaded yet. Now the breakdown is defined based on the origin to which the pick-up truck is driving; the truck driving to that origin breaks down. The time to repair is longer for these scenarios and is assumed to be normally distributed with a mean of 600 min and a standard deviation of 20.

The three different control modes are applied to all scenarios. If control mode 3 is used, the VRSS is informed about the expected repair time (estimated by the truck agent), which is then included in the corresponding travel time. For the scenarios with a long breakdown (scenarios 11 to 15), the intentions of the

order agents are cleared to prevent blocking when the staff holon updates its advice after the pick-up truck breaks down.

Results For scenarios 1 to 5, each origin has 3, 4 or 5 orders that have to be picked up, except for scenario 4, in which only 1 order has to be picked up at O1 and 7 orders at O2. After consolidation, these orders have to be transported to D1 or D2. Both destinations are expecting 3 to 5 of the 8 orders. When control mode 1 is applied, the behavior that can be observed is very divergent. Both pick-up trucks travel once or multiple times to O1 and O2 to pick up all orders. To deliver the 8 orders to their destinations, DT1 makes one tour or several trips to D1 and D2. When control mode 2 or 3 is applied, the behavior that can be observed is more consistent. Each pick-up truck visits another origin to transport the orders to the cross-dock. After consolidation, DT1 delivers all orders by making a tour to the two destinations. Figure 6.20 shows the mean and standard error (as defined by equation (6.1) and (6.2)) of the considered performance measures for scenarios 1 to 5.

For the scenarios with a short breakdown (scenarios 6 to 10), the mean and standard error of the four performance measures are shown in Figure 6.21. For control mode 1, the observed behavior is again very different for the various replications. To pick up the orders, PT1 and PT2 travel once or multiple times to the origins and PT1 or PT2 breaks down during one of its trips to the cross-dock. DT1 then makes one tour or several trips to D1 and D2 to deliver all orders. When control mode 2 or 3 is applied, the behavior that can be observed is again more consistent. Mostly, each pick-up truck visits another origin to transport the orders to the cross-dock. Because of the breakdown of PT1 or PT2, one truck arrives later than the other at the cross-dock. DT1 does not depart before this belated truck has arrived and delivers the 8 orders by making a tour to the two destinations.

In scenarios 11 to 15, a pick-up trucks breaks down on its way to one of the origins. If the operations are controlled by control mode 1, the behavior that can be observed is again very divergent. Both pick-up trucks travel once or several times to the two origins to pick up all orders. DT1 delivers already some orders at their destination before the broken down truck arrives for the first time at the cross-dock and then delivers the remaining orders by making one tour or multiple trips to D1 and D2. If control mode 2 is applied, the representative behavior is that each pick-up truck visits another origin to transport the orders to the cross-dock. Because of the breakdown of PT1 or PT2, one truck arrives much later at the cross-dock. DT1 does not wait for this truck and delivers the first arrived orders by making a tour to the two destinations whereupon DT1 returns to the cross-dock. In the meanwhile, the broken down truck has arrived

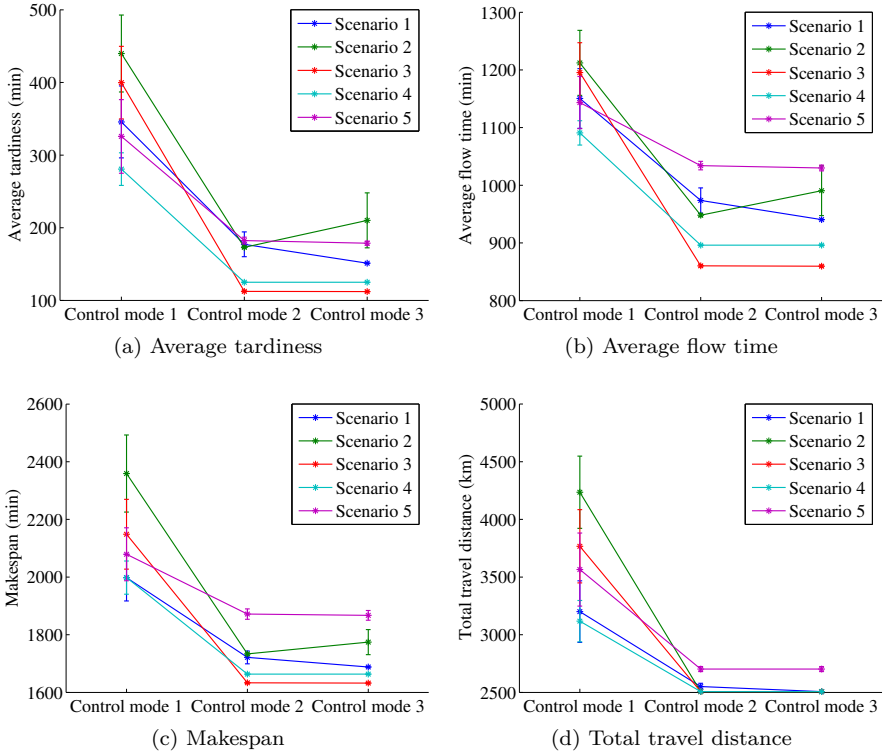
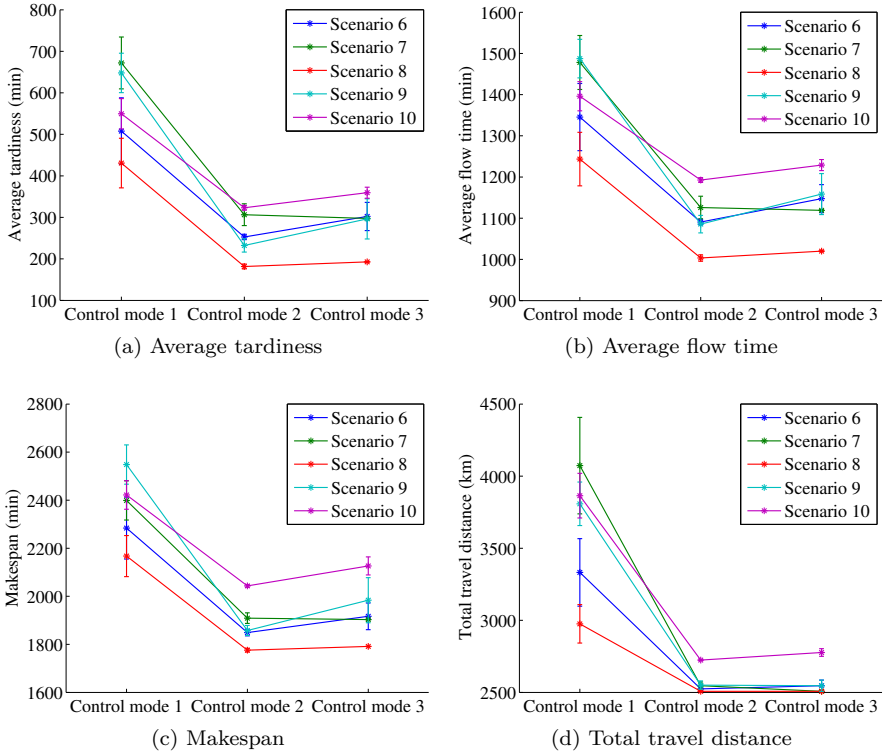


Figure 6.20: Performance measures for scenarios 1 to 5 (8 orders, no breakdown).

at the cross-dock with the remaining orders and DT1 delivers these orders by making the same tour to the destinations. If control mode 3 is used, the broken down truck does not pick up the orders at the origin it was going to. Instead, the other pick-up truck will, after it has picked up the other orders, directly drive to this origin to pick-up these orders and to transport them to the cross-dock. So, all orders arrive together at the cross-dock and are then delivered at their destinations by DT1 which makes a tour to the two destinations. Figure 6.22 shows the mean and standard error of the considered performance measures for scenarios 11 to 15.

Discussion As can be clearly noticed in Figure 6.20 for the scenarios without breakdown, control mode 1 is outperformed by mode 2 and 3 for all considered performance measures. Not only the average values are higher, but also the variation over the 10 replications is larger (as indicated by the standard error).



(a) Average tardiness

(b) Average flow time

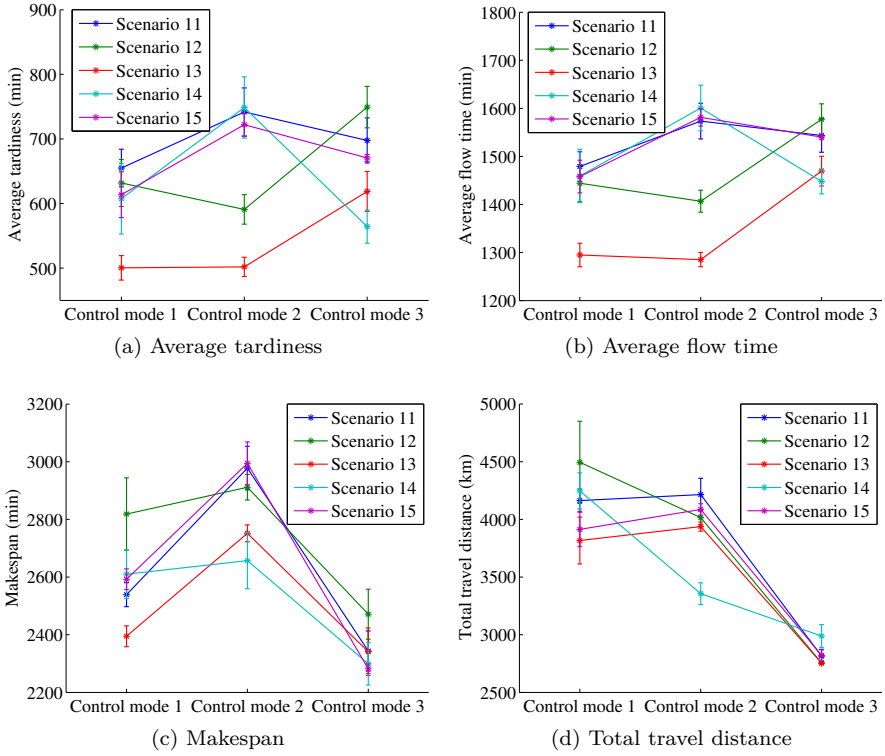
(c) Makespan

(d) Total travel distance

Figure 6.21: Performance measures for scenarios 6 to 10 (8 orders, short breakdown).

Without advice from the staff holon, it is difficult for the various order agents to find a good batching. Although in a few replications a similar execution is obtained as if advice was received, in most simulation runs the trucks are used less efficiently.

When control mode 2 or 3 is applied, all performance measures improve and the variation over the 10 replications is smaller. The order agents now make use of the advice in which PT1 and PT2 pick up orders at different origins and DT1 makes one tour to deliver all orders (via D2 to D1 for scenarios 1 to 4, via D1 to D2 for scenario 5). This explains the better average values and also the smaller variations. There are two reasons for the small variations in performance for control mode 2 and 3. Firstly, these variations are caused by variations in the unloading sequence of the orders at their destination. Secondly, in some simulation runs, the local rules employed by the truck agents have



(a) Average tardiness (b) Average flow time

(c) Makespan (d) Total travel distance

Figure 6.22: Performance measures for scenarios 11 to 15 (8 orders, long breakdown).

shifted the transport operations (slightly) forward in time compared to the advice given by the staff holon. For some scenarios, there are larger variations, notably in scenario 1 for control mode 2 and in scenario 2 for control mode 3. In scenario 1, this variation is caused by two replications in which the initial advice is not optimal (DT1 is advised to make a tour via D1 to D2 instead of vice versa). In scenario 2, larger performance measures are obtained in one replication in which blocking occurs and causes PT1 to arrive much later at the cross-dock. These three replications with deviating behavior also give rise to higher average values. This distorts a bit the comparison between control mode 2 and 3, but certainly for scenarios 3, 4 and 5 it can be seen that both control modes obtain similar values for all considered performance measures. As there are no disturbances, the schedule updates provided by the staff holon in control mode 3 are not beneficial. Note that, for control mode 2 and 3, the

total travel distance is higher in scenario 5 than in the other scenarios (see Figure 6.20d). Indeed, in scenario 5, the advice of the staff holon for DT1 is to make a larger tour (via D1 to D2), causing a higher total travel distance. This results also in a higher average flow time and makespan, as can be seen in Figures 6.20b and 6.20c.

For scenarios 6 to 10, Figure 6.21 also indicates that control mode 1 is outperformed by mode 2 and 3 for all considered performance measures. Again, it is difficult for the order agents to obtain a good batching without advice from the staff holon. As a result of the breakdown of one of the pick-up trucks, the average tardiness, average flow time and makespan have increased compared with scenarios 1 to 5.

When control mode 2 is applied, all performance measures improve and the variation over the 10 replications is smaller. Similarly as in scenarios 1 to 5, the order agents are able to find better solutions by using the advice provided by the staff holon. When one of the pick-up trucks breaks down, the order agents have to adapt their intentions. Due to the local rules employed by the truck agents, they are able to find new solutions with the same resource allocation, but shifted forward in time. So, the departure time of DT1 will be postponed until all orders have arrived at the cross-dock and can be delivered at their destination by one tour of DT1. As a result, the average tardiness, average flow time and makespan of scenarios 6 to 10 have increased with about 70 to 170 min compared to the corresponding scenario without breakdown. The total travel distance does not increase as the trajectories of the trucks are still the same. The variation in the performance measures (especially for scenarios 7 and 9) is due to a few replications in which the staff holon's initial advice is not optimal. Another reason is that DT1 makes two separate trips to D1 and D2 instead of one tour.

If control mode 3 is applied to coordinate and control the operations, the staff holon updates its advice for the order agents after the truck breakdown by shifting the departure time of DT1 forward in time. The order agents are able to adapt their intentions according to the new advice. This results in a similar execution as with control mode 2. Again, some variation in the performance measures can be noticed (for scenarios 6, 9 and 10). This variation is once more caused by the occurrence of blocking in a few replications or because the initial VRPCD schedule provided by the staff holon is not optimal. The average tardiness, average flow time and makespan are again higher compared to the corresponding scenarios without breakdown (about 80 to 210 min). These values are also slightly higher compared to the measures obtained with control mode 2. This can be explained by the margin included in the original VRPCD schedule (as the assumed transfer time is generous). If control mode 2 is applied, this margin will be (partially) consumed to reduce the effect of the breakdown. For

mode 3 however, the same margin will be included in the updated schedule and this margin cannot be consumed. So, the departure time of DT1 will be later compared to control mode 2.

For the scenarios with a long breakdown (scenarios 11 to 15), the results are more ambiguous. Moreover, they are different for the various scenarios as the specific characteristics of the scenarios become more important. If control mode 1 is applied, it is still difficult for the order agents to find intentions with a good batching and the trucks have to make multiple trips. The truck breakdown causes the affected orders to be transported later, but the effect on the performance measures is not very large. While the expected repair time is 600 min, the makespan increases with about 280 to 570 min compared to the scenarios without breakdown, and the average tardiness and average flow time increase with about 140 to 410 min.

For control mode 2, the advice given by the staff holon helps the order agents to find good solutions. However, when one of the pick-up trucks breaks down, they have to adapt their intentions. As the local rules employed by the truck agents do not allow a large shift in time, the belated orders will have to make new reservations for DT1 after the delivery truck has completed its first tour. As the order agents will use (the resource allocation of) the staff holon's advice as a guideline, in most replications a solution is found in which DT1 delivers these orders in one tour. In several replications however, DT1 delivers the belated orders in two separate trips to D1 and D2 or needs an extra trip to one of the destinations, which causes variation in the performance measures. Other causes of variation are the occurrence of blocking and a suboptimal initial advice (in which the tour of DT1 is reversed). As can be seen in Figure 6.22c, the makespan of scenarios 11 to 15 has increased compared to control mode 1. Because DT1 directly delivers the first arrived orders by making a tour, the broken down truck arrives before DT1 at the cross-dock and the belated orders have to wait for a considerable amount of time for the return of the delivery truck. The effect of this on the average tardiness and average flow time is different for the various scenarios (see Figures 6.22a and 6.22b). For scenarios 11, 14, and 15, this increased waiting time of the belated orders results in an increase of these performance measures. For scenarios 12 and 13 however, the average tardiness and average flow time are similar or even smaller compared to control mode 1. This can be explained by the number of orders that are belated and how many of these orders have to be delivered at the end of DT1's tour. For scenarios 12 and 13, only 3 orders are transported by the broken down truck (compared to 4 or 7 for the other scenarios) and only 1 of these orders has to be delivered at the last stop of DT1's tour (compared to 2 to 4 orders for scenarios 11, 14, and 15). The total travel distance is similar as with control mode 1, except for scenario 14 (see Figure 6.22d). In this scenario, there is only 1 order that is

not delayed and so DT1 does not have to make a complete tour to deliver the orders that are not belated.

If control mode 3 is applied, the staff holon updates the VRPCD schedule after one of the pick-up trucks breaks down. The staff holon now advises all order agents to be picked up by the other truck. To adapt their intentions according to this new advice, the time slot reserved for a transport operation to the cross-dock has to be replaced by a transport operation between the origins. This is difficult because this current slot 'blocks' the pick-up truck and does not allow to make other reservations. To make it possible for the order agents to adapt to the new advice and as the current implementation of the research prototype has no correct mechanism to deal with blocking (e.g. aggregation), the intentions of the order agents are cleared when the VRPCD schedule is adapted. In most replications, this allows the order agents to find a new solution according to the updated advice. In several replications however, blocking still occurs for the reservations on DT1 and this results in DT1 waiting for a considerable amount of time at its first stop before delivering the orders at its final destination. Together with a few replications in which the initial VRPCD schedule provided by the staff holon is not optimal, this explains the variation in the considered performance measures. Compared to control mode 2, the makespan of scenarios 11 to 15 has improved (see Figure 6.22c). All orders are now picked up by one truck and the affected orders arrive a bit earlier at the cross-dock. Moreover, they do not have to wait for the return of DT1 (which is still at the cross-dock) but are directly transported to their destinations in one tour. The makespan is also better compared to control mode 1. As now the orders that would have been picked up by the broken down truck arrive earlier at their destination, but the other orders later, the effect on the average tardiness and average flow time is also dependent on the number of orders affected by the breakdown and how many of these orders have to be delivered at the end of DT1's tour (see Figures 6.22a and 6.22b). For scenarios 11, 14, and 15, 4 to 7 orders arrive now much earlier at their destination, while the delay for the remaining orders is smaller. As a consequence, the average tardiness and average flow time improve. For scenarios 12 and 13 however, the improvement in tardiness and flow time of 3 orders does not compensate the delay of the other orders and so the average tardiness and flow time increase. As the delivery truck now only makes one tour to deliver all orders, the total travel distance has decreased for all scenarios compared to control mode 2 (see Figure 6.22d). Note that for scenario 14, the total travel distance is larger compared to the other scenarios. In this scenario, the pick-up truck makes a tour in reverse order (according to the updated advice), and this tour is somewhat longer.

As can be noticed in Figures 6.20, 6.21 and 6.22, the variation in the considered performance measures can be large. This can be explained by the nonlinear

nature of the system. A small deviation can have a large impact on the performance measures. For instance, if a pick-up truck departs from an origin without all orders loaded, it has to return to pick-up the remaining orders, increasing the total travel distance with several hundred km and adding a few hours to the tardiness and flow time of these orders. For control mode 2 and 3, the standard error is larger for the scenarios with a breakdown. Because of the disturbance, the order agents have to adapt their intentions and blocking can occur, certainly if they also have to adapt the resource allocation (scenarios 11 to 15). For these cases, mechanisms to prevent blocking (e.g. aggregation) are required to have a more consistent performance.

For scenarios 1 to 10, it is clear that control mode 2 and 3 have a similar performance and outperform mode 1 (objective 3). The picture is however not so clear for the scenarios with a long breakdown. For all scenarios and performance measures, control mode 2 or 3 obtain better results than mode 1. It can be argued that control mode 3 is better than mode 2 as it is assumed that the external scheduling system provides a good schedule from a global point of view and the order agents make use of the updated advice. Apparently, this updated schedule prefers a shorter travel distance over earlier arrival times. This is reflected in a lower total travel distance (and makespan) obtained with control mode 3. A consequence of following the advice is a higher average tardiness and average flow time for some scenarios.

If control mode 2 is applied, the order agents are guided by the staff holon's original advice, even after the occurrence of the truck breakdown. As a result, in scenarios 11 to 15, DT1 makes two times a tour to both destinations to deliver all orders. While this can lead to a performance improvement (e.g. average tardiness for scenario 12), this is not generally true. So, in a next implementation of the software, it might be better to abandon the provided advice after a (large) disturbance (at least if the advice is not updated) and search for solutions in a self-organizing manner. If control mode 3 is applied for scenarios 11 to 15, the intentions of all order agents are cleared in order to allow them to find new solutions according to the updated VRPCD schedule. Generally, this is not a good approach. Indeed, the order agents can throw away good intentions, while the provided schedule may even not be feasible. Moreover, the problem of blocking is not completely solved and the root causes should be addressed (for instance by aggregation).

6.4.2 Sixteen orders

This section describes the results of a series of scenarios in which 16 orders have to be transported. Similar to the experiments with 8 orders, 15 different

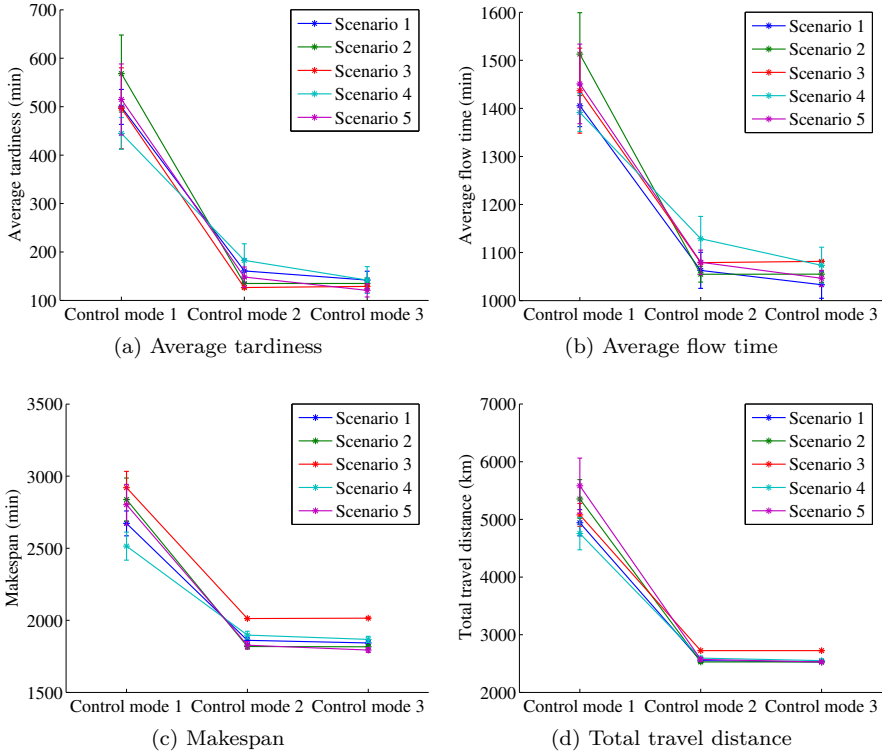
scenarios are considered: five without a breakdown, five with a short truck breakdown and five with a long breakdown.

Set-up For scenarios 1 to 5, set-up 1 is again used as base set-up. For each of these scenarios, 16 orders are processed with randomly chosen origins, destinations, release times and due dates. The due dates are randomly sampled between 800 and 1200 min after their corresponding release times. Tables A.10 to A.14 show the chosen values (Appendix A). To construct scenarios 6 to 10, a truck breakdown is added to the first five scenarios. One of the two pick-up trucks breaks down at a random time between 650 and 750 min, when the truck is transporting orders to the cross-dock. The time to repair is normally distributed with a mean of 180 min and a standard deviation of 6. The breakdown is again linked to the origin of the truck. Scenarios 11 to 15 are constructed in a similar way as for the experiments with 8 orders, by adding a truck breakdown when the truck is on its way to one of the origins (between 350 and 450 min). The time to repair is again normally distributed with a mean of 600 min and a standard deviation of 20. Similar to the experiments with 8 orders, the three control modes are applied to the 15 scenarios. For each scenario, ten replications are simulated.

Results The mean and standard error of the considered performance measures are shown in Figure 6.23 for scenarios 1 to 5, in Figure 6.24 for the scenarios with a short breakdown (scenarios 6 to 10) and in Figure 6.25 for scenarios 11 to 15. The results are in line with the results of the experiments with 8 orders.

If control mode 1 is applied, the observed behavior is very different for the various replications. The pick-up trucks PT1 and PT2 travel once or multiple times to the origins and DT1 makes one tour or several trips to D1 and D2 to deliver all orders.

When control mode 2 is applied, the behavior that can be observed is more consistent, certainly for scenarios 1 to 5. Each pick-up truck visits another origin to transport the orders to the cross-dock. After consolidation, DT1 delivers all orders by making a tour to the two destinations. The behavior for scenarios 6 to 10 is similar, and DT1 waits until the belated truck has arrived at the cross-dock to start its delivery tour. For scenarios 11 to 15, one pick-up truck arrives much later at the cross-dock. DT1 does not wait for this truck and delivers the first arrived orders by making a tour to the two destinations. The remaining orders are then delivered by making another tour to the destinations or by multiple trips to D1 and D2.



(a) Average tardiness (b) Average flow time

(c) Makespan (d) Total travel distance

Figure 6.23: Performance measures for scenarios 1 to 5 (16 orders, no breakdown).

For scenarios 1 to 10, the behavior is similar if control mode 3 is applied as if mode 2 is used to control the logistic operations. A different behavior can however be observed for the scenarios with a long breakdown (scenarios 11 to 15). For these scenarios, the broken down truck does not pick up any order, but all orders are collected by the other pick-up truck which makes a tour to both origins. So, all orders arrive together at the cross-dock and are then delivered at their destinations by DT1 which makes a tour to D1 and D2.

Discussion For the scenarios without breakdown (scenarios 1 to 5), the results are in accordance with the results of the experiments with 8 orders. Figure 6.23 shows that control mode 2 and 3 outperform control mode 1 for all considered performance measures, both in average value and variation over the 10 replications. Without advice from the staff holon, it is difficult for the

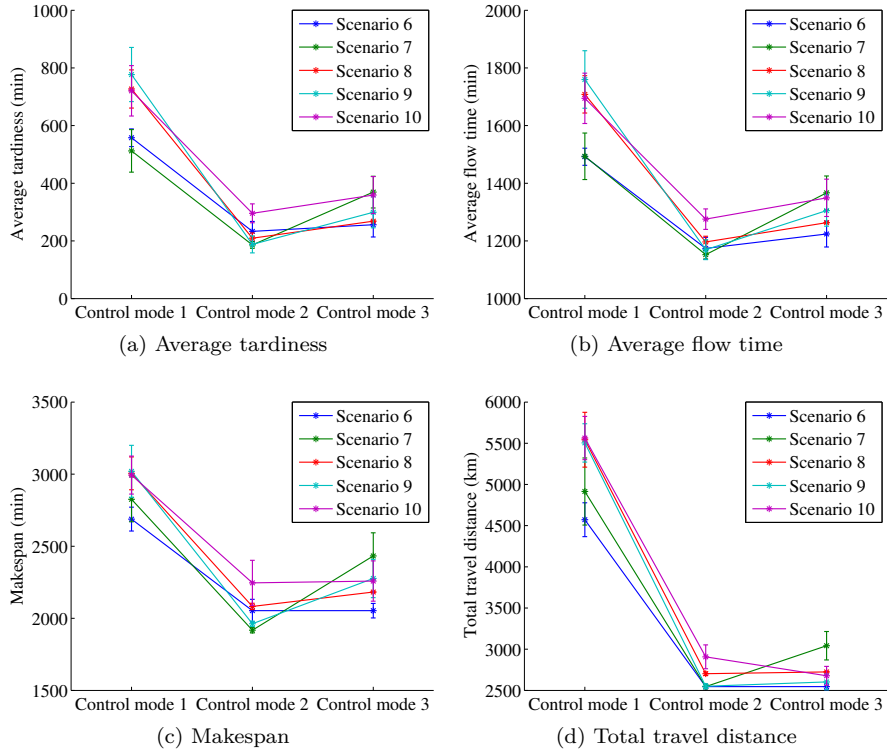
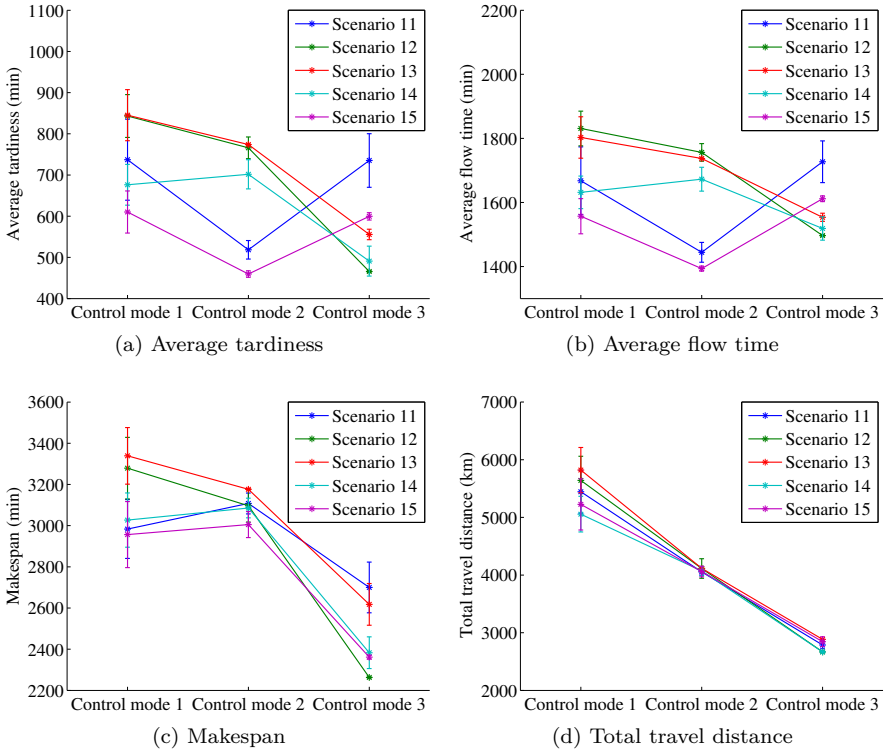


Figure 6.24: Performance measures for scenarios 6 to 10 (16 orders, short breakdown).

order agents to efficiently make use of the trucks. If advice is provided, the order agents are able to execute this advice: PT1 and PT2 pick up orders at different origins and DT1 makes one tour to deliver all orders (via D2 to D1 for scenarios 1, 2, 4 and 5, via D1 to D2 for scenario 3). Control mode 2 and 3 obtain similar performance measures. Indeed, the schedule updates provided by the staff holon in control mode 3 are not beneficial, as there are no disturbances. The variations in performance for control mode 2 and 3 are mainly caused by replications in which the (initial) advice is not optimal (DT1 is advised to make a tour via D1 to D2 instead of vice versa).

For scenarios 6 to 10, the results are again in line with the results of the experiments with 8 orders. As expected, the breakdown has a negative effect compared to scenarios 1 to 5. Once more, it is difficult for the order agents to obtain a good batching without advice from the staff holon. So, the values of



(a) Average tardiness (b) Average flow time

(c) Makespan (d) Total travel distance

Figure 6.25: Performance measures for scenarios 11 to 15 (16 orders, long breakdown).

all considered performance measures are worse for control mode 1 compared to mode 2 and 3 (see Figure 6.24). If control mode 2 is applied, the order agents have to adapt their initial solutions (based on the provided advice) after the breakdown of one of the pick-up trucks. They are able to find new solutions with the same resource allocation, but shifted forward in time. So, the departure time of delivery truck DT1 is postponed until all orders have arrived at the cross-dock. Subsequently, all orders can be delivered at their destination by one tour of DT1. The variation in the performance measures (especially for scenario 10) is mainly due to the occurrence of blocking in several replications or because the advice schedule provided by the staff holon is not optimal. Another reason is that in a few replications the delivery truck DT1 makes two separate trips to D1 and D2 instead of one tour. If control mode 3 is applied to control the operations, the staff holon's advice is updated after the truck breakdown

by shifting the departure time of DT1 forward in time. The order agents are able to adapt their intentions according to the new advice. This results in a similar execution as with control mode 2. Again, variation in the performance measures can be noticed (especially for scenarios 7 and 10). The reason for this variation is mainly the occurrence of blocking, which results in extra trips of DT1 and/or DT1 which unnecessary waits at the cross-dock or one of the destinations.

The results are more ambiguous for the scenarios with a long breakdown (scenarios 11 to 15), but once more they are in accordance with the results of the experiments with 8 orders. As can be seen in Figure 6.25, it is again difficult for the order agents to find intentions with a good batching without advice (control mode 1). This results in different behavior for the various replications and explains the large standard error. For control mode 2, the order agents have to adapt their intentions (based on the provided advice) after the truck breakdown. As a large shift in time is not allowed by the truck agent, the belated orders will have to make new reservations for DT1 after the delivery truck has completed its first tour. Based on (the resource allocation of) the staff holon's advice, in most replications a solution is found in which DT1 delivers these orders in one tour. There are however replications in which DT1 needs an extra trip to one of the destinations or delivers the belated orders in two separate trips to D1 and D2. This leads to variation in the performance measures. Other causes of variation are the occurrence of blocking and an initial advice in which the delivery tour is reversed. Because DT1 directly starts delivering the first arrived orders, the broken down truck arrives before the delivery truck at the cross-dock and the belated orders have to wait for the return of DT1. The effect of this waiting on the average tardiness and average flow time depends on the specific characteristics of the scenarios (see Figures 6.25a and 6.25b). For scenarios 11 and 15, less than half of the orders are transported by the broken down truck and arrive considerably later at their destination. For the other scenarios, at least 8 orders are affected by the breakdown and arrive much later at their destination. This explains the increased average tardiness and average flow time for these scenarios compared to the decreased values for scenarios 11 and 15.

If control mode 3 is applied, the advice for the orders is updated after the truck breakdown and indicates that all orders should be picked up by the other truck. In most replications, the order agents are able to find a new solution according to this updated advice. In several replications however, blocking still occurs for the reservations on DT1 and this results in DT1 waiting for a considerable amount of time at the cross-dock or at its first stop. This blocking is the main cause of the variation in the considered performance measures. Different from control mode 2, all orders are now picked up by one truck and the affected

orders arrive earlier at the cross-dock. As DT1 is waiting at the cross-dock, the orders are then directly transported to their destinations (in one tour). This leads to a serious improvement in makespan (see Figure 6.25c). The effect on the average tardiness and average flow time is again dependent on the specific characteristics of the scenarios. As can be seen in Figures 6.25a and 6.25b, The average tardiness and average flow time have improved for scenarios 12, 13 and 14, but have increased for scenarios 11 and 15. This difference can again be explained by the number of orders affected by the breakdown. As delivery truck DT1 now only makes one tour, the total travel distance has decreased for all scenarios compared to control mode 2 (see Figure 6.25d).

6.5 Conclusions

This chapter presents the results of the experimental evaluation of the proposed on-line control system. To this end, a research prototype was developed and tested in simulation. Two types of simulation experiments were described. The first type of experiments considered specific scenarios to highlight the value added of the holonic approach. These experiments confirm that the HLES is able to cooperate with external scheduling algorithms and that this cooperation improves the performance. Moreover, if some aspects are not taken into account by the scheduling algorithms, the HLES is able to deal with these aspects. This approach allows for a cooperation scheme in which the staff holon gives advice to the orders and resources when a larger disturbance occurs, and the order, resource and product agents deal in a self-organizing way with the smaller deviations in between. The experiments also indicate that the holonic system provides visibility about the current and future resource and order states. This allows the responsible decision makers to intervene on time if necessary or desired.

The second type of experiments consists of multiple replications of the same scenario to compare the three considered control modes. In total, 900 replications are executed. These experiments confirm that the cooperation between the HLES and external scheduling systems improves the performance. The cooperation with the TSS allows for a good ‘truck to dock door’ assignment, while especially the cooperation with the VRSS has a positive impact on the performance measures by improving the batching of the orders (for transportation). In case of disturbances, the original advice from the staff holon becomes obsolete, and the performance can be improved if the advice is updated (unless the updated advice is just a shift forward, in which case the self-organizing approach is able to find a similar solution). However, to obtain a better and more consistent performance, the current implementation of the research prototype should be

adapted in order to deal with blocking. This adaptation should also make it unnecessary to clear the intentions of the order agents in order to be able to find solutions in accordance with the updated schedule.

Chapter 7

Conclusions

7.1 Summary of conclusions

The aim of this thesis was the development of a Logistics Execution System or LES in accordance with the concepts and principles of the Holonic Manufacturing Execution System (HMES). Similar to Manufacturing Execution Systems, an LES is responsible for the real-time control of the logistic operations. It is situated between the physical processes and the planning at office level. The considered HMES makes use of the ideas of the holonic paradigm to organize the manufacturing control. Its software architecture combines the PROSA reference architecture and the delegate MAS architectural pattern. By using this pattern, the HMES is able to provide a view on the expected short-term future of the manufacturing system.

The HMES consists of a reusable 'core' which provides the basic functionality, completed with domain-specific models and application-specific decision mechanisms. For all relevant entities in the world-of-interest (products, resources and orders), there is a clear separation between reflection of reality (being or environment entity) and decision making (agent). As a result, the models to reflect the real-world entities can be easily reused in other applications containing these entities. So, they can be considered as domain-specific building blocks. On the other hand, the various decision mechanisms (e.g. the selection of an intention by the order agents) are application-specific and need to be developed in order to meet the specific requirements and concerns of the considered application. Because of the clear separation, the decision making can be considered as a plug-in to the system which can be easily replaced by another

algorithm or rule. By a better structure of the software, the development of new implementations could be facilitated. In fact, new applications can be tackled by implementing or reusing the models of the real-world entities and by providing the necessary decision mechanisms. This reuse of software components reduces the development time and costs of a new application. Moreover, the reusable software becomes gradually more mature, resulting in increased reliability and performance and reduced maintenance costs. The clear separation of concerns and single-source-of-truth design also guarantee a high degree of (software) flexibility for evolving circumstances. The existing functionality can be easily adapted as changes are limited to local software modifications.

As the logistics domain is very broad, this thesis focused on a specific logistic strategy: cross-docking. This strategy enables (rapid) consolidation, leading to fewer vehicles that each carries more freight. The resulting increase in efficiency comes however at the expense of a more difficult organization of the logistic operations. For instance, a better information flow is necessary for the coordination between inbound and outbound vehicles. The thesis introduced the cross-docking concept and provides *a review and classification of the existing literature about cross-docking.* The considered papers are classified based on the problem type (ranging from strategic or tactical to operational problems). This review revealed some limitations and opportunities in the field of cross-docking. The available literature does not consider all types of cross-docks, not all problem types are extensively discussed and most papers are concerned with just one problem. The main drawback of the presented approaches is however the limited applicability. Many simplifying assumptions are made and the reviewed approaches are not suited for a dynamic environment. By proposing a LES to organize the cross-docking operations, the thesis tried to overcome these last limitations.

This LES can be supported by a global view from a scheduling system in order to improve its performance. To this end, the thesis proposed two scheduling approaches related to cross-docking. Firstly, *a scheduling method for the truck scheduling problem* was developed. This problem is concerned with the assignment of trucks to the different dock doors of a cross-dock. Secondly, the thesis proposed *a scheduling method for the vehicle routing problem with cross-docking*, which is concerned with the assignment of orders to trucks. For both problems, a heuristic method was developed to find good results in a reasonable amount of time. Indeed, for the truck scheduling experiments, the average relative deviation compared to the best solution found is lower than 0.25 % and the average calculation time is less than 1.2 s. For the vehicle routing experiments, the average relative deviation is lower than 0.4 % and all solutions were found in less than 0.1 s. The thesis also proposed adaptations to the heuristic approach of the VRPCD problem in order to be able to reschedule.

Although both problems consider many aspects in a realistic manner, they still provide a simplified and approximated view of the real-world problem.

The organization of cross-docking operations, or more generally logistic operations, is a complex task. A severe international competition and ever-increasing traffic (congestion) make that logistic systems should be able to operate efficiently in uncertain and dynamic environments. Moreover, the operational control of logistic operations is a going concern, so 'one-shot optimization' is not sufficient. As the logistics domain is characterized by similar properties as the manufacturing domain (e.g. large decision space, nonlinearity, uncertainty, etc.), this thesis proposes to apply the concepts and principles of the HMES in order to develop a Holonic LES or HLES. Its software architecture then also combines the PROSA reference architecture and the delegate MAS pattern into a working control system. Therefore, this thesis indicates that *PROSA and delegate MAS can be applied in the logistics domain*, well beyond the original application range.

As indicated higher, the HMES consists of a reusable core, domain-specific models and application-specific decision mechanisms. *To develop a holonic on-line control system for cross-docking*, it should be sufficient to implement or reuse models of the relevant physical entities and to provide the required decision making mechanisms. However, as some elements were missing in the core of the latest HMES implementation, this thesis proposed several adaptations and extensions to the basic functionality. These adaptations allow the HLES to offer support for mobile resources, batching and multi-resource allocation. As these changes involve the core of the coordination and control system, they are also available for other implementations. They can for instance be used by HMES implementations, as these adaptations are also of interest for manufacturing applications.

The thesis also presents the necessary models and decision mechanisms for the entities relevant in the context of cross-docking. Based on the provided decision making mechanisms, the HLES is able to provide a view on the expected short-term future of the system. This short-term forecasting ability is a very important aspect of the HLES. The consequences of decisions based on the provided decision making mechanisms now become visible beforehand. In case of disturbances or changes in decision making, the generated short-term forecasts will be adapted accordingly. The provided up-to-date visibility is a valuable contribution. It allows for instance the detection of potential capacity conflicts in advance so that the necessary actions can be taken. Moreover, better (informed) decisions can be made based on these forecasts.

The cross-docking HLES can be supported in finding good global solutions by cooperating with the proposed vehicle routing scheduling system and truck

scheduling system. To this end, *a cooperation mechanism between the holonic on-line control system and these scheduling systems* is implemented. This cooperation happens through the staff holon and allows the HLES to execute the provided schedules, while accounting for (small) deviations and possible simplifications. The proposed scheduling approaches are not able to account for all details, which is indeed not an obvious task. As the HLES compensates for possible simplifications, this problem is reduced. Moreover, the HLES also deals with deviations from the schedule, making the scheduling approach more robust against disturbances and uncertainty. The schedule execution is an important and often overlooked aspect of scheduling practice.

Based on the presented approach to develop a cross-docking HLES, a research prototype is implemented. This *HLES prototype is tested* in simulation to show that the implemented system works as intended (multi-resource allocation, batching, etc.). Two types of simulation experiments were performed. The first type of experiments considered specific scenarios to highlight the value added of the proposed approach. These experiments confirmed that the HLES can cooperate with external scheduling systems and that this cooperation leads to an improved performance. Moreover, the HLES was able to deal with aspects that were not taken into account by the vehicle routing and truck scheduling system. The simulation experiments also indicated that the HLES provides visibility about the current and future resource and order states. Based on these short-term forecasts, the HLES can react proactively instead of purely reactively. This visibility also allows the responsible decision makers to intervene on time if necessary or desired. For the second type of experiments, 900 replications in total were executed in order to compare the three considered control modes. The experiments confirmed that the cooperation between the HLES and external scheduling systems improves the performance (as measured by the average tardiness, average flow time, makespan and total travel distance). The provided truck schedule allowed for a good 'truck to dock door' assignment, while especially the vehicle routing schedule had a positive impact on the performance measures by improving the batching of the orders (for transportation). It has been shown that the HLES concept is a valuable option for a cross-docking LES.

7.2 Suggestions for future work

This thesis presented the development of an HLES implementation to coordinate and control the cross-docking operations. A functioning prototype was built, but there are still some issues that need to be tackled in order to be applicable in practice. A major issue is the occurrence of 'blocking'. As this leads to

performance loss, a sound solution is required to deal with this problem. A possible solution could be that the affected orders (temporarily) form an aggregated order holon that makes reservations on behalf of its subholons. Some smaller issues are related to the application-specific decision mechanisms. For instance, for the truck holons, the rule to shift a transport slot forward in time can result in a (too) large deviation of this slot compared to the original reservation. Improvements can also be made to the intention selection mechanism of the order holons, e.g. by considering other performance measures.

The overview of the available literature about cross-docking revealed that there are still opportunities for future research about scheduling approaches. More specifically, attention can be paid to several cross-dock types and problem types which are not (or less) considered, for instance cross-dock layout design. Also, the applicability of the scheduling approaches could be increased by taking more real-world aspects into account (e.g. limited storage capacity) and by accounting for disturbances (e.g. truck breakdowns or traffic jams).

Although this thesis is focused on cross-docking, the applicability of the HMES is more general. A similar coordination and control problem exists also in other logistic situations. A HLES can be implemented in nodes of the logistic network where goods are consolidated for further transport. Examples are ports, intermodal hubs and parcel hubs. The cross-docking strategy can also be extended by the integration of upstream and downstream processes. In this thesis, the inbound and outbound transport operations are integrated with the cross-dock operations to improve the global performance. A further step could be the integration of several holonic execution systems, for instance a HMES and HLES. This could be realized by (exploring and intention) ants travelling across the borders of the holonic systems. Note that it is not necessary to disclose the internal functioning of a subsystem to the outside world. It is sufficient to reveal the end result of the virtual execution.

A similar method can be used to organize distribution networks. For instance, in distribution networks for retail, freight is transported from a main cross-dock or distribution center to a regional cross-dock from where the goods are delivered at retail outlets. The holonic approach can be applied to cover complete logistic networks. It is clear that more integrated logistic coordination and control systems - such as the proposed HLES - are required to cope with today's worldwide trade and global competition. In a network operating HLES's provide a promising solution which deserves to be further explored.

Appendix A

Experimental set-up

This appendix contains detailed data, omitted in Chapter 6, about the orders considered in the simulation experiments. More concretely, for all experiments, the origins, destinations, release times and due dates of the orders are provided here. Times are expressed in minutes. Section A.1 contains the data for the scenario tests (described in Section 6.3). The data for the replicated tests (discussed in Section 6.4) are displayed in Section A.2, more specifically in Section A.2.1 for the experiments with eight orders and in Section A.2.2 for the experiments with sixteen orders.

A.1 Scenario tests

Table A.1: Order set 1 considered in experiments 1, 2, 3, 4, 7 and 8.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O2	D1	551.0	1347.3
Order 2	O1	D2	576.1	1412.7
Order 3	O1	D2	429.5	1185.7
Order 4	O1	D2	470.4	1372.7
Order 5	O2	D1	586.9	1357.4
Order 6	O2	D2	454.7	1245.6
Order 7	O2	D1	457.1	1334.7
Order 8	O1	D1	579.1	1360.3

Table A.2: Order set 2 considered in experiment 5.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D1	418.7	1763.0
Order 2	O1	D1	577.2	1891.6
Order 3	O1	D2	493.4	1330.9
Order 4	O1	D2	531.3	1391.5
Order 5	O2	D1	591.1	1305.4
Order 6	O2	D1	426.0	1164.9
Order 7	O2	D2	569.5	1486.0
Order 8	O2	D2	504.5	1359.6

Table A.3: Order set 3 considered in experiment 6.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D1	568.4	1330.4
Order 2	O1	D1	550.6	1381.5
Order 3	O1	D1	468.2	1396.2
Order 4	O1	D2	590.2	1382.1
Order 5	O1	D2	454.8	1222.6
Order 6	O2	D2	573.6	1430.0
Order 7	O2	D1	447.2	1220.5
Order 8	O1	D1	421.0	1281.3

Table A.4: Order set 4 considered in experiment 9.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D2	524.6	1321.5
Order 2	O1	D1	534.5	1383.2
Order 3	O2	D2	482.4	1256.6
Order 4	O1	D2	591.1	1467.8
Order 5	O2	D2	562.9	1480.5
Order 6	O1	D1	522.2	1311.2
Order 7	O2	D1	450.6	1253.4
Order 8	O2	D1	481.3	1254.3

A.2 Replicated tests

A.2.1 Eight orders

Table A.5: Eight orders considered in scenarios 1, 6 and 11.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D1	555.5	1416.1
Order 2	O2	D1	584.9	1496.8
Order 3	O1	D1	452.9	1357.9
Order 4	O2	D2	417.8	1280.9
Order 5	O1	D2	549.4	1383.3
Order 6	O1	D2	555.9	1371.5
Order 7	O2	D1	461.0	1252.4
Order 8	O2	D2	463.4	1248.3

Table A.6: Eight orders considered in scenarios 2, 7 and 12.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D2	466.4	1350.4
Order 2	O2	D2	473.6	1259.7
Order 3	O2	D1	414.6	1329.4
Order 4	O1	D1	430.7	1190.2
Order 5	O2	D1	541.9	1294.5
Order 6	O1	D2	598.7	1471.9
Order 7	O2	D2	530.9	1340.1
Order 8	O2	D2	401.8	1246.9

Table A.7: Eight orders considered in scenarios 3, 8 and 13.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O2	D2	432.5	1226.0
Order 2	O2	D1	554.5	1491.3
Order 3	O2	D2	563.3	1490.1
Order 4	O1	D1	447.3	1278.3
Order 5	O1	D2	428.6	1317.4
Order 6	O2	D1	411.7	1206.0
Order 7	O1	D2	440.6	1275.9
Order 8	O2	D2	478.7	1277.6

Table A.8: Eight orders considered in scenarios 4, 9 and 14.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D2	524.6	1434.0
Order 2	O2	D1	509.1	1340.9
Order 3	O2	D1	458.0	1385.4
Order 4	O2	D1	592.2	1489.5
Order 5	O2	D2	551.1	1397.2
Order 6	O2	D1	496.6	1425.7
Order 7	O2	D2	572.9	1477.5
Order 8	O2	D2	499.4	1322.4

Table A.9: Eight orders considered in scenarios 5, 10 and 15.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D1	574.9	1467.6
Order 2	O1	D1	585.3	1356.7
Order 3	O2	D1	579.8	1397.9
Order 4	O2	D1	586.8	1422.6
Order 5	O1	D2	545.9	1495.7
Order 6	O2	D1	451.3	1287.2
Order 7	O2	D2	507.0	1410.2
Order 8	O1	D2	542.8	1492.6

A.2.2 Sixteen orders

Table A.10: Sixteen orders considered in scenarios 1, 6 and 11.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O2	D2	490.2	1344.4
Order 2	O2	D2	510.3	1561.4
Order 3	O2	D2	491.1	1459.3
Order 4	O2	D2	566.3	1373.1
Order 5	O1	D1	555.8	1451.8
Order 6	O2	D2	440.8	1571.8
Order 7	O2	D1	565.8	1548.7
Order 8	O2	D2	483.5	1623.4
Order 9	O2	D1	460.6	1339.3
Order 10	O1	D1	598.9	1623.4
Order 11	O1	D2	449.4	1497.8
Order 12	O1	D2	592.6	1445.8
Order 13	O2	D2	513.7	1670.9
Order 14	O1	D2	525.1	1503.0
Order 15	O2	D1	444.9	1498.5
Order 16	O2	D1	447.3	1493.4

Table A.11: Sixteen orders considered in scenarios 2, 7 and 12.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D1	500.8	1628.5
Order 2	O2	D2	482.2	1414.2
Order 3	O2	D2	539.6	1631.6
Order 4	O1	D1	451.3	1472.6
Order 5	O2	D2	584.0	1386.3
Order 6	O2	D1	487.8	1301.7
Order 7	O2	D1	572.7	1711.6
Order 8	O2	D2	424.3	1565.5
Order 9	O1	D1	556.0	1615.6
Order 10	O1	D1	528.0	1555.4
Order 11	O2	D1	512.7	1711.5
Order 12	O2	D2	527.2	1620.2
Order 13	O1	D2	473.1	1618.8
Order 14	O2	D1	447.8	1360.0
Order 15	O1	D2	548.4	1405.4
Order 16	O1	D2	406.1	1540.9

Table A.12: Sixteen orders considered in scenarios 3, 8 and 13.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D1	414.4	1561.4
Order 2	O2	D1	570.6	1503.7
Order 3	O1	D1	489.5	1552.8
Order 4	O1	D1	503.7	1628.6
Order 5	O1	D1	471.1	1578.1
Order 6	O2	D1	432.1	1608.6
Order 7	O2	D1	541.5	1511.0
Order 8	O2	D1	459.1	1606.6
Order 9	O2	D1	461.1	1275.5
Order 10	O1	D1	593.2	1745.5
Order 11	O2	D1	416.1	1485.5
Order 12	O1	D1	434.1	1283.6
Order 13	O2	D1	545.9	1386.9
Order 14	O2	D2	588.8	1467.1
Order 15	O1	D2	522.8	1379.0
Order 16	O2	D1	428.7	1266.0

Table A.13: Sixteen orders considered in scenarios 4, 9 and 14.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O1	D2	463.7	1510.4
Order 2	O1	D2	575.9	1739.8
Order 3	O2	D2	476.0	1446.8
Order 4	O2	D2	478.6	1536.3
Order 5	O2	D2	586.4	1467.3
Order 6	O1	D1	444.8	1576.8
Order 7	O1	D2	408.4	1432.8
Order 8	O2	D1	575.2	1710.0
Order 9	O2	D1	491.6	1350.0
Order 10	O1	D1	595.6	1447.7
Order 11	O2	D2	510.2	1477.2
Order 12	O2	D2	405.9	1537.9
Order 13	O1	D1	544.5	1647.2
Order 14	O1	D2	491.9	1620.8
Order 15	O1	D1	518.5	1648.3
Order 16	O2	D2	491.0	1352.8

Table A.14: Sixteen orders considered in scenarios 5, 10 and 15.

Order(s)	Origin	Destination	Release time	Due date
Order 1	O2	D2	465.8	1523.5
Order 2	O1	D1	489.7	1662.7
Order 3	O2	D2	542.7	1468.3
Order 4	O2	D2	544.5	1549.1
Order 5	O1	D2	476.0	1400.2
Order 6	O2	D2	518.8	1522.8
Order 7	O2	D2	451.3	1578.9
Order 8	O2	D1	518.8	1569.5
Order 9	O2	D1	527.9	1530.3
Order 10	O2	D2	406.6	1260.4
Order 11	O1	D2	450.8	1336.9
Order 12	O1	D1	420.3	1525.4
Order 13	O2	D1	554.6	1537.9
Order 14	O1	D1	551.3	1748.6
Order 15	O2	D1	422.2	1499.5
Order 16	O2	D1	573.7	1393.5

Bibliography

[1] *2008 Cross-Docking Trends Report*. White paper, Saddle Creek Corporation. `http://www.saddlecrk.com/whitepaper`. 2008.

[2] *2011 Cross-Docking Trends Report*. White paper, Saddle Creek Corporation. `http://www.saddlecrk.com/whitepaper`. 2011.

[3] M. K. Acar. "Robust Dock Assignments at Less-Than-Truckload Terminals". Master's thesis. University of South Florida, 2004.

[4] O. Ali, P. Valckenaers, J. Van Belle, B. Saint Germain, P. Verstraete, and D. Van Oudheusden. "Towards online planning for open-air engineering processes". *Computers in Industry* 64(3), pp. 242–251, 2013.

[5] O. Ali. "Operational Planning for Outdoor Engineering Processes". PhD thesis. Katholieke Universiteit Leuven, 2010.

[6] G. Alpan, R. Larbi, and B. Penz. "A bounded dynamic programming approach to schedule operations in a cross docking platform". *Computers & Industrial Engineering* 60(3), pp. 385–396, 2011.

[7] G. A. Álvarez-Pérez, J. L. González-Velarde, and J. W. Fowler. "Crossdocking—Just in Time scheduling: an alternative solution approach". *Journal of the Operational Research Society* 60(4), pp. 554–564, 2009.

[8] U. M. Apte and S. Viswanathan. "Effective Cross Docking for Improving Distribution Efficiencies". *International Journal of Logistics: Research and Applications* 3(3), pp. 291–302, 2000.

[9] A. R. B. Arabani, S. M. T. F. Ghomi, and M. Zandieh. "A multi-criteria cross-docking scheduling with just-in-time approach". *The International Journal of Advanced Manufacturing Technology* 49(5-8), pp. 741–756, 2010.

[10] A. R. B. Arabani, S. M. T. F. Ghomi, and M. Zandieh. "Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage". *Expert Systems with Applications* 38(3), pp. 1964–1979, 2011.

[11] N. J. Ashford, S. A. Mumayiz, and P. H. Wright. *Airport Engineering: Planning, Design, and Development of 21st-Century Airports*. 4th ed. Wiley, 2011.

[12] H. Aytug, M. A. Lawley, K. McKay, S. Mohan, and R. Uzsoy. "Executing production schedules in the face of uncertainties: A review and some future directions". *European Journal of Operational Research* 161(1), pp. 86–110, 2005.

[13] R. F. Babiceanu and F. F. Chen. "Development and applications of holonic manufacturing systems: a survey". *Journal of Intelligent Manufacturing* 17(1), pp. 111–131, 2006.

[14] M. Bachlaus, M. K. Pandey, C. Mahajan, R. Shankar, and M. K. Tiwari. "Designing an integrated multi-echelon agile supply chain network: a hybrid taguchi-particle swarm optimization approach". *Journal of Intelligent Manufacturing* 19(6), pp. 747–761, 2008.

[15] K. R. Baker. *Elements of sequencing and scheduling*. Amos Tuck School of Business Administration, Dartmouth College, 1994.

[16] J. J. Bartholdi III and K. R. Gue. "Reducing Labor Costs in an LTL Crossdocking Terminal". *Operations Research* 48(6), pp. 823–832, 2000.

[17] J. J. Bartholdi III and K. R. Gue. "The Best Shape for a Crossdock". *Transportation Science* 38(2), pp. 235–244, 2004.

[18] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. SEI Series in Software Engineering. Addison-Wesley, 1998.

[19] R. Bermúdez and M. H. Cole. *A Genetic Algorithm Approach to Door Assignments in Breakbulk Terminals*. Tech. rep. MBTC 1084. Mack-Blackwell Rural Transportation Center, University of Arkansas, 2001.

[20] D. J. Bertsimas. "A Vehicle Routing Problem with Stochastic Demand". *Operations Research* 40(3), pp. 574–585, 1992.

[21] D. Bjørner. "On "The Right" Software". *International Journal of Software and Informatics* 5(3), pp. 509–523, 2011.

[22] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. "Routing in Telecommunications Networks with Ant-Like Agents". In: *Intelligent Agents for Telecommunication Applications*, pp. 60–71. Ed. by S. Albayrak, F. J. Garijo, J. Siekmann, and J. G. Carbonell. Lecture Notes in Artificial Intelligence 1437. Springer, 1999.

[23] L. Bongaerts. “Integration of scheduling and control in holonic manufacturing systems”. PhD thesis. Katholieke Universiteit Leuven, 1998.

[24] L. Bongaerts, L. Monostori, D. McFarlane, and B. Kádár. “Hierarchy in distributed shop floor control”. *Computers in Industry* 43(2). Special issue on intelligent manufacturing systems, pp. 123–137, 2000.

[25] N. Boysen. “Truck scheduling at zero-inventory cross docking terminals”. *Computers & Operations Research* 37(1), pp. 32–41, 2010.

[26] N. Boysen and M. Fliedner. “Cross dock scheduling: Classification, literature review and research agenda”. *Omega* 38(6), pp. 413–422, 2010.

[27] N. Boysen, M. Fliedner, and A. Scholl. “Scheduling inbound and outbound trucks at cross docking terminals”. *OR Spectrum* 32(1), pp. 135–161, 2010.

[28] Y. A. Bozer and H. J. Carlo. “Optimizing inbound and outbound door assignments in less-than-truckload crossdocks”. *IIE Transactions* 40(11), pp. 1007–1018, 2008.

[29] O. Bräysy and M. Gendreau. “Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms”. *Transportation Science* 39(1), pp. 104–118, 2005.

[30] O. Bräysy and M. Gendreau. “Vehicle Routing Problem with Time Windows, Part II: Metaheuristics”. *Transportation Science* 39(1), pp. 119–139, 2005.

[31] R. A. Brooks. “A Robust Layered Control System For A Mobile Robot”. *IEEE Journal of Robotics and Automatiom* 2(1), pp. 14–23, 1986.

[32] A. M. Brown. “Improving the Efficiency of Hub Operations in a Less-than-Truckload Distribution Network”. Master’s thesis. Virginia Polytechnic Institute and State University, 2003.

[33] S. Brückner. “Return From The Ant—Synthetic Ecosystems for Manufacturing Control”. PhD thesis. Humboldt-Universität zu Berlin, 2000.

[34] H.-J. Bürckert, K. Fisher, and G. Vierke. “Transportation Scheduling with Holonic MAS - The TeleTruck Approach”. In: *Proceedings of the 3rd International Conference on Practical Applications of Intelligent Agents and Multiagents (PAAM 1998)*, pp. 577–590. London, UK, 1998.

[35] B. Burmeister, A. Haddadi, and G. Matylis. “Application of multi-agent systems in traffic and transportation”. *IEE Proceedings - Software Engineering* 144(1), pp. 51–60, 1997.

[36] M. Caridi and S. Cavalieri. “Multi-agent systems in production planning and control: an overview”. *Production Planning & Control* 15(2), pp. 106–118, 2004.

[37] F. Chen and C.-Y. Lee. "Minimizing the makespan in a two-machine cross-docking flow shop problem". *European Journal of Operational Research* 193(1), pp. 59–72, 2009.

[38] F. Chen and K. Song. "Minimizing makespan in two-stage hybrid cross docking scheduling problem". *Computers & Operations Research* 36(6), pp. 2066–2073, 2009.

[39] P. Chen, Y. Guo, A. Lim, and B. Rodrigues. "Multiple crossdocks with inventory and time windows". *Computers & Operations Research* 33(1), pp. 43–63, 2006.

[40] A. Chmielewski, B. Naujoks, M. Janas, and U. Clausen. "Optimizing the Door Assignment in LTL-Terminals". *Transportation Science* 43(2), pp. 198–210, 2009.

[41] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. 2nd ed. SEI Series in Software Engineering. Addison-Wesley, 2011.

[42] W. Clermonts and W. Ploos van Amstel. "Speed docking: meer lading met minder vrachtwagens". In: *Bijdragen vervoerslogistieke werkdagen 2012*. Venlo, The Netherlands, 2012.

[43] *CO_2 Emissions from Fuel Combustion - Highlights*. International Energy Agency (IEA). `http://www.iea.org/publications/freepublications/publication/name,32870,en.html`. 2012.

[44] Y. Cohen and B. Keren. "Trailer to door assignment in a synchronous cross-dock operation". *International Journal of Logistics Systems and Management* 5(5), pp. 574–590, 2009.

[45] R. L. Cook, B. Gibson, and D. MacCurdy. "A lean approach to cross docking". *Supply Chain Management Review* 9(2), pp. 54–59, 2005.

[46] J. Cottyn. "Design of a Lean Manufacturing Execution System Framework". PhD thesis. Universiteit Gent, 2012.

[47] P. Cowling and M. Johansson. "Using real time information for effective dynamic scheduling". *European Journal of Operational Research* 139(2), pp. 230–244, 2002.

[48] G. B. Dantzig and J. H. Ramser. "The Truck Dispatching Problem". *Management Science* 6(1), pp. 80–91, 1959.

[49] P. Davidsson, H. Lawrence, L. Ramstedt, J. Törnquist, and F. Wernstedt. "An analysis of agent-based approaches to transport logistics". *Transportation Research Part C* 13(4), pp. 255–271, 2005.

[50] K. De Swert, P. Valckenaers, B. Saint Germain, P. Verstraete, Hadeli, and H. Van Brussel. "Coordination and control for railroad networks inspired by manufacturing control". In: *Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS 2006)*, pp. 201–206. Prague, Czech Republic, 2006.

[51] G. Di Caro and M. Dorigo. "AntNet: Distributed Stigmergetic Control for Communications Networks". *Journal of Artificial Intelligence Research* 9, pp. 317–365, 1998.

[52] D. M. Dilts, N. P. Boyd, and H. H. Whorms. "The Evolution of Control Architectures for Automated Manufacturing Systems". *Journal of Manufacturing Systems* 10(1), pp. 79–93, 1991.

[53] H. Donaldson, E. L. Johnson, H. D. Ratliff, and M. Zhang. *Schedule-Driven Cross-Docking Networks*. Tech. rep. 9904. Georgia Institute of Technology, 1999.

[54] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella. "Time dependent vehicle routing problem with a multi ant colony system". *European Journal of Operational Research* 185(3), pp. 1174–1191, 2008.

[55] M. Dorigo and G. Di Caro. "The Ant Colony Optimization Meta-Heuristic". In: *New ideas in optimization*, chap. 2, pp. 11–32. Ed. by D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price. Advanced Topics in Computer Science Series. McGraw-Hill, 1999.

[56] M. Dorigo and T. Stützle. "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances". In: *Handbook of Metaheuristics*, chap. 9, pp. 250–285. Ed. by F. Glover and G. A. Kochenberger. International Series in Operations Research & Management Science 57. Springer, 2003.

[57] U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch. "Flight gate scheduling: State-of-the-art and recent developments". *Omega* 35(3), pp. 326–334, 2007.

[58] *Enterprise-Control System Integration, Part 1: Models and Terminology*. ANSI/ISA-95.00.01-2000. Instrument Society of America (ISA), 2000.

[59] *Enterprise-Control System Integration, Part 3: Models of Manufacturing Operations Management*. ANSI/ISA-95.00.03-2005. Instrument Society of America (ISA), 2005.

[60] *EURopean Inter-Disciplinary research on Intelligent Cargo for Efficient, safe and environment-friendly logistics*. `http://www.euridice-project.eu`. 2010.

[61] M. J. Euwe and H. Wortmann. “Planning systems in the next century (I)”. *Computers in Industry* 34(2), pp. 233–237, 1997.

[62] K. Fisher, J. P. Müller, and M. Pischel. “Cooperative Transportation Scheduling: An Application Domain for DAI”. *Applied Artificial Intelligence* 10(1), pp. 1–33, 1996.

[63] P. A. Fishwick and B. P. Zeigler. “A Multimodel Methodology for Qualitative Model Engineering”. *ACM Transactions on Modeling and Computer Simulation* 2(1), pp. 52–81, 1992.

[64] S. Forouharfard and M. Zandieh. “An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems”. *The International Journal of Advanced Manufacturing Technology* 51(9), pp. 1179–1193, 2010.

[65] J. M. Framinan and R. Ruiz. “Architecture of manufacturing scheduling systems: Literature review and an integrated proposal”. *European Journal of Operational Research* 205(2), pp. 237–246, 2010.

[66] *Freight Transport Logistics in Europe – the key to sustainable mobility.* Communication from the Commission to the Council, the European Parliament, the European Economic and Social Committee and the Committee of the Regions (COM(2006) 336 final). `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:52006DC0336:EN:NOT`. 2006.

[67] M. R. Galbreth, J. A. Hill, and S. Handley. “An Investigation of the Value of Cross-Docking for Supply Chain Management”. *Journal of Business Logistics* 29(1), pp. 225–239, 2008.

[68] M. Gendreau, G. Laporte, and R. Séguin. “Stochastic vehicle routing”. *European Journal of Operational Research* 88(1), pp. 3–12, 1996.

[69] *Glossary of the Material Handling Institute (MHI).* `http://www.mhi.org/glossary`. Consulted: April 2013.

[70] F. Glover. “Tabu Search—Part I”. *ORSA Journal on Computing* 1(3), pp. 190–206, 1989.

[71] F. Glover. “Tabu Search—Part II”. *ORSA Journal on Computing* 2(1), pp. 4–32, 1990.

[72] A. Goel and V. Gruhn. “A General Vehicle Routing Problem”. *European Journal of Operational Research* 191(3), pp. 650–660, 2008.

[73] R. Goodwin. “Formalizing Properties of Agents”. *Journal of Logic and Computation* 5(6), pp. 763–781, 1995.

[74] P.-P. Grassé. “La reconstruction du nid et les coordinations Inter-Individuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d’interprétation du comportement des termites constructeurs”. *Insectes Sociaux* 6(1), pp. 41–80, 1959.

[75] K. R. Gue. “The Effects of Trailer Scheduling on the Layout of Freight Terminals”. *Transportation Science* 33(4), pp. 419–428, 1999.

[76] K. R. Gue and K. Kang. “Staging queues in material handling and transportation systems”. In: *Proceedings of the 33rd conference on Winter simulation*, pp. 1104–1108. Arlington, Virginia, USA, 2001.

[77] M. Gümüş and J. H. Bookbinder. “Cross-docking and its Implications in Location-Distribution Systems”. *Journal of Business Logistics* 25(2), pp. 199–228, 2004.

[78] Hadeli. “Bio-inspired multi-agent manufacturing control systems with social behaviour”. PhD thesis. Katholieke Universiteit Leuven, 2006.

[79] Hadeli, P. Valckenaers, M. Kollingbaum, and H. Van Brussel. “Multi-agent coordination and control using stigmergy”. *Computers in Industry* 53(1), pp. 75–96, 2004.

[80] Hadeli, P. Valckenaers, H. Van Brussel, B. Saint Germain, P. Verstraete, and J. Van Belle. “Towards the Design of Autonomic Nervousness Handling in Holonic Manufacturing Execution Systems”. In: *Proceedings of the 2007 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2883–2888. Montreal, Canada, 2007.

[81] Hadeli, P. Valckenaers, H. Van Brussel, P. Verstraete, B. Saint Germain, and J. Van Belle. “Production planning and control in bio-inspired holonic manufacturing execution systems”. In: *Preprints of the 8th IFAC International Workshop on Intelligent Manufacturing Systems (IMS 2007)*, pp. 48–55. Alicante, Spain, 2007.

[82] Hadeli, P. Valckenaers, P. Verstraete, B. Saint Germain, and H. Van Brussel. “A Study of System Nervousness in Multi-agent Manufacturing Control System”. In: *Engineering Self-Organising Systems*, pp. 232–243. Ed. by S. A. Brueckner, G. D. M. Serugendo, D. Hales, and F. Zambonelli. Lecture Notes in Artificial Intelligence 3910. Springer, 2006.

[83] L. Harjunkoski, R. Nyström, and A. Horch. “Integration of scheduling and control—Theory or practice?” *Computers and Chemical Engineering* 33(12), pp. 1909–1918, 2009.

[84] A. Helleboogh. “Simulation of Distributed Control Applications in Dynamic Environments”. PhD thesis. Katholieke Universiteit Leuven, 2007.

[85] W. Herroelen and R. Leus. “Robust and reactive project scheduling: a review and classification of procedures”. *International Journal of Production Research* 42(8), pp. 1599–1620, 2004.

[86] M. Hinchey, M. Jackson, P. Cousot, B. Cook, J. P. Bowen, and T. Margaria. “Software Engineering and Formal Methods”. *Communications of the ACM* 51(9), pp. 54–59, 2008.

[87] T. Holvoet and P. Valckenaers. “Exploiting the Environment for Coordinating Agent Intentions”. In: *Environments for Multi-Agent Systems III*, pp. 51–66. Ed. by D. Weyns, H. V. D. Parunak, and F. Michel. Lecture Notes in Artificial Intelligence 4389. Springer, 2007.

[88] T. Holvoet, D. Weyns, and P. Valckenaers. “Patterns of Delegate MAS”. In: *Proceedings of the 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2009)*, pp. 1–9. San Francisco, California, USA, 2009.

[89] S. Ichoua, M. Gendreau, and J.-Y. Potvin. “Vehicle dispatching with time-dependent travel times”. *European Journal of Operational Research* 144(2), pp. 379–396, 2003.

[90] O. Jabali, T. Van Woensel, C. Lecluyse, H. Peremans, and A. G. de Kok. “Stochastic vehicle routing with random time dependent travel times subject to perturbations”. In: *Bijdragen Vervoerslogistieke Werkdagen (VLW 2007)*. Ed. by F. J. Witlox and C. J. Ruijgrok, pp. 585–597. Grobbendonk, Belgium, 2007.

[91] A. K. Jain and H. A. Elmaraghy. “Production scheduling/rescheduling in flexible manufacturing”. *International Journal of Production Research* 35(1), pp. 281–309, 1997.

[92] S. Jain and W. J. Foley. “Impact of Interruptions on Schedule Execution in Flexible Manufacturing Systems”. *International Journal of Flexible Manufacturing Systems* 14(4), pp. 319–344, 2002.

[93] V. Jayaraman and A. Ross. “A simulated annealing methodology to distribution network design and management”. *European Journal of Operational Research* 144(3), pp. 629–645, 2003.

[94] S. Kallas. *Using freight to help European transport move to a sustainable future*. Speech of Siim Kallas, Vice-President of the European Commission and European Commissioner responsible for Transport, at the Launch of the Green Freight Europe initiative in Brussels (March 2012). `http://europa.eu/rapid/press-release_SPEECH-12-230_en.htm`. 2012.

[95] M. Kärkkäinen, J. Holmström, K. Främling, and K. Artto. “Intelligent products—a step towards a more effective project delivery chain”. *Computers in Industry* 50(2), pp. 141–151, 2003.

[96] *Keep Europe moving - Sustainable mobility for our continent.* Communication from the Commission to the Council and the European Parliament (COM(2006) 314 final). `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:52006DC0314:EN:HTML:NOT`. 2006.

[97] E. Kinnear. "Is there any magic in cross-docking?" *Supply Chain Management: An International Journal* 2(2), pp. 49–52, 1997.

[98] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. "Optimization by Simulated Annealing". *Science* 220(4598), pp. 671–680, 1983.

[99] A. Koestler. *The Ghost in the Machine.* Macmillan, 1967.

[100] D. Konur and M. M. Golias. "Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach". *Transportation Research Part E: Logistics and Transportation Review* 1(49), pp. 71–91, 2013.

[101] V. B. Kreng and F.-T. Chen. "The benefits of a cross-docking delivery strategy: a supply chain collaboration approach". *Production Planning & Control* 19(3), pp. 229–241, 2008.

[102] S. Kumar. "A study of the supermarket industry and its growing logistics capabilities". *International Journal of Retail & Distribution Management* 36(3), pp. 192–211, 2008.

[103] G. Laporte. "The Vehicle Routing Problem: An overview of exact and approximate algorithms". *European Journal of Operational Research* 59(3), pp. 345–358, 1992.

[104] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. "Classical and modern heuristics for the vehicle routing problem". *International Transactions in Operational Research* 7(4-5), pp. 285–300, 2000.

[105] G. Laporte, F. Louveaux, and H. Mercure. "The Vehicle Routing Problem with Stochastic Travel Times". *Transportation Science* 26(3), pp. 161–170, 1992.

[106] R. Larbi, G. Alpan, P. Baptiste, and B. Penz. "Scheduling cross docking operations under full, partial and no information on inbound arrivals". *Computers & Operations Research* 38(6), pp. 889–900, 2011.

[107] C. Lecluyse, T. Van Woensel, and H. Peremans. "Vehicle routing with stochastic time-dependent travel times". *4OR: A Quarterly Journal of Operations Research* 7(4), pp. 363–377, 2009.

[108] Y. H. Lee, J. W. Jung, and K. M. Lee. "Vehicle routing scheduling for cross-docking in the supply chain". *Computers & Industrial Engineering* 51(2), pp. 247–256, 2006.

[109] P. Leitão. “Holonic Rationale and Bio-inspiration on Design of Complex Emergent and Evolvable Systems”. *Transactions on Large-Scale Data- and Knowledge-Centered Systems I* 5740, pp. 243–266, 2009.

[110] P. Leitão and F. Restivo. “ADACOR: A holonic architecture for agile and adaptive manufacturing control”. *Computers in Industry* 57(2), pp. 121–130, 2006.

[111] P. Leitão and F. Restivo. “A holonic approach to dynamic manufacturing scheduling”. *Robotics and Computer-Integrated Manufacturing* 24(5), pp. 625–634, 2008.

[112] M. A. Lejeune and N. Yakova. “On characterizing the 4 C’s in supply chain management”. *Journal of Operations Management* 23(1), pp. 81–100, 2005.

[113] Y. Li, A. Lim, and B. Rodrigues. “Crossdocking—JIT Scheduling with Time Windows”. *The Journal of the Operational Research Society* 55(12), pp. 1342–1351, 2004.

[114] Z. P. Li, M. Y. H. Low, M. Shakeri, and Y. G. Lim. *Cross-docking planning and scheduling: Problems and algorithms*. Tech. rep. STR_V10_N3_06_POM. Singapore Institute of Manufacturing Technology, 2009.

[115] Z. Li, M. Y. H. Low, Y. G. Lim, and B. Ma. “Optimal Decision-making on Product Ranking For Crossdocking/Warehousing Operations”. In: *Proceedings of the 6th IEEE International Conference on Industrial Informatics (INDIN 2008)*, pp. 871–876. Daejeon, Korea, 2008.

[116] C.-J. Liao, Y. Lin, and S. C. Shih. “Vehicle routing with cross-docking in the supply chain”. *Expert Systems with Applications* 37(10), pp. 6868–6873, 2010.

[117] A. Lim, H. Ma, and Z. Miao. “Truck Dock Assignment Problem with Operational Time Constraint Within Crossdocks”. In: *Advances in Applied Artificial Intelligence*, pp. 262–271. Lecture Notes in Artificial Intelligence 4031. Springer, 2006.

[118] A. Lim, H. Ma, and Z. Miao. “Truck Dock Assignment Problem with Time Windows and Capacity Constraint in Transshipment Network Through Crossdocks”. In: *Computational Science and Its Applications - ICCSA 2006*, pp. 688–697. Lecture Notes in Computer Science 3982. Springer, 2006.

[119] A. Lim, Z. Miao, B. Rodrigues, and Z. Xu. “Transshipment through Crossdocks with Inventory and Time Windows”. *Naval Research Logistics* 52(8), pp. 724–733, 2005.

[120] H. Ma, Z. Miao, A. Lim, and B. Rodrigues. "Crossdocking distribution networks with setup cost and time window constraint". *Omega* 39(1), pp. 64–72, 2011.

[121] G. M. Magableh, M. D. Rossetti, and S. Mason. "Modeling and analysis of a generic cross-docking facility". In: *Proceedings of the 37th conference on Winter simulation*, pp. 1613–1620. Orlando, Florida, USA, 2005.

[122] C. Malandraki and M. S. Daskin. "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms". *Transportation Science* 26(3), pp. 185–200, 1992.

[123] S. Mallik. "Customer Service in Supply Chain Management". In: *Supply Chain Management, Marketing and Advertising, and Global Management*, pp. 103–119. Ed. by H. Bidgoli. Vol. 2. The Handbook of Technology Management. Wiley, 2010.

[124] M. Mamei, F. Zambonelli, and L. Leonardi. "Co-Fields: A Physically Inspired Approach to Motion Coordination". *Pervasive Computing* 3(2), pp. 52–61, 2004.

[125] D. C. McFarlane and S. Bussmann. "Developments in Holonic Production Planning and Control". *International Journal of Production Planning and Control* 11(6), pp. 522–536, 2000.

[126] D. McFarlane, S. Sarma, J. L. Chirn, C. Y. Wong, and K. Ashton. "Auto ID systems and intelligent manufacturing control". *Engineering Applications of Artificial Intelligence* 16(4), pp. 365–376, 2003.

[127] K. M. McKay and V. C. S. Wiers. "Unifying the Theory and Practice of Production Scheduling". *Journal of Manufacturing Systems* 18(4). Special issue on scheduling: From Research Into Practice, pp. 241–255, 1999.

[128] D. L. McWilliams. "A dynamic load-balancing scheme for the parcel hub-scheduling problem". *Computers & Industrial Engineering* 57(3), pp. 958–962, 2009.

[129] D. L. McWilliams. "Genetic-based scheduling to solve the parcel hub scheduling problem". *Computers & Industrial Engineering* 56(4), pp. 1607–1616, 2009.

[130] D. L. McWilliams. "Iterative improvement to solve the parcel hub scheduling problem". *Computers & Industrial Engineering* 59(1), pp. 136–144, 2010.

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

[132] D. L. McWilliams, P. M. Stanfield, and C. D. Geiger. "Minimizing the completion time of the transfer operations in a central parcel consolidation terminal with unequal-batch-size inbound trailers". *Computers & Industrial Engineering* 54(4), pp. 709–720, 2008.

[133] G. G. Meyer, K. Främling, and J. Holmström. "Intelligent Products: A survey". *Computers in Industry* 60(3), pp. 137–148, 2009.

[134] Z. Miao, K. Fu, Q. Fei, and F. Wang. "Meta-heuristic Algorithm for the Transshipment Problem with Fixed Transportation Schedules". In: *New Frontiers in Applied Artificial Intelligence*, pp. 601–610. Lecture Notes in Artificial Intelligence 5027. Springer, 2008.

[135] Z. Miao, A. Lim, and H. Ma. "Truck dock assignment problem with operational time constraint within crossdocks". *European Journal of Operational Research* 192(1), pp. 105–115, 2009.

[136] L. Monostori, J. Váncza, and S. R. T. Kumara. "Agent-Based Systems for Manufacturing". *CIRP Annals - Manufacturing Technology* 55(2), pp. 697–720, 2006.

[137] T. Murata. "Petri Nets: Properties, Analysis and Applications". *Proceedings of the IEEE* 77(4), pp. 541–580, 1989.

[138] R. Musa, J.-P. Arnaout, and H. Jung. "Ant colony optimization algorithm to solve for the transportation problem of cross-docking network". *Computers & Industrial Engineering* 59(1), pp. 85–92, 2010.

[139] M. Napolitano. *Making the move to cross docking: A practical guide to planning, designing, and implementing a cross dock operation.* Warehousing Education and Research Council (WERC), 2000.

[140] M. Napolitano. "Cross dock fuels growth at Dots". *Logistics Management* 50(2), pp. 30–34, 2011.

[141] R. de Neufville, A. G. de Barros, and S. Belin. "Optimal Configuration of Airport Passenger Buildings for Travelers". *Journal of Transportation Engineering* 128(3), pp. 211–217, 2002.

[142] J. M. Novas and G. P. Henning. "Reactive scheduling framework based on domain knowledge and constraint programming". *Computers and Chemical Engineering* 34(12), pp. 2129–2148, 2010.

[143] Y. Oh, H. Hwang, C. N. Cha, and S. Lee. "A dock-door assignment problem for the Korean mail distribution center". *Computers & Industrial Engineering* 51(2), pp. 288–296, 2006.

[144] D. Ouelhadj and S. Petrovic. "A survey of dynamic scheduling in manufacturing systems". *Journal of Scheduling* 12(4), pp. 417–431, 2009.

[145] P. Paganelli, V. Charalampos, E. Cornelisse, A. Damentka, M. Forcolin, M. Jermol, R. Styczynski, W. Szczurek, and D. Vedovato. *The EURIDICE Project – EURopean Inter-Disciplinary research on Intelligent Cargo for Efficient, safe and environment-friendly logistics*. White paper. `http://www.euridice-project.eu/index.php/web/pubdocs/58`. 2009.

[146] H. V. D. Parunak. "Manufacturing Experience with the Contract Net". In: *Distributed Artificial Intelligence*, chap. 10, pp. 285–310. Ed. by M. N. Huhns. Pitman Publishing London, 1987.

[147] H. V. D. Parunak. ""Go to the ant": Engineering principles from natural multi-agent systems". *Annals of Operations Research* 75, pp. 69–101, 1997.

[148] S. Pearson Specter. "How to crossdock successfully". *Modern Materials Handling* 59(1), pp. 42–46, 2004.

[149] K. E. Peck. "Operational Analysis of Freight Terminals Handling Less Than Container Load Shipments". PhD thesis. University of Illinois at Urbana-Champaign, 1983.

[150] P. Peeters, H. Van Brussel, P. Valckenaers, J. Wyns, L. Bongaerts, M. Kollingbaum, and T. Heikkilä. "Pheromone based emergent shop floor control system for flexible flow shops". *Artificial Intelligence in Engineering* 15(4), pp. 343–352, 2001.

[151] J. Philips. "Holonic task execution control of multi-mobile-robot systems". PhD thesis. KU Leuven, 2012.

[152] J. Philips, P. Valckenaers, E. Aertbeliën, J. Van Belle, B. Saint Germain, H. Bruyninckx, and H. Van Brussel. "PROSA and Delegate MAS in Robotics". In: *Holonic and Multi-Agent Systems for Manufacturing*, pp. 195–204. Ed. by V. Mařík, P. Vrba, and P. Leitão. Lecture Notes in Artificial Intelligence 6867. Springer, 2011.

[153] A. Pokahr, L. Braubach, J. Sudeikat, W. Renz, and W. Lamersdorf. "Simulation and Implementation of Logistics Systems based on Agent Technology". In: *Hamburg International Conference on Logistics (HICL 2008): Logistics Networks and Nodes*. Ed. by T. Blecker, W. Kersten, and C. Gertz, pp. 291–308. Hamburg, Germany, 2008.

[154] R. G. Qiu and M. Zhou. "Mighty MESs; State-of-the-Art and Future Manufacturing Execution Systems". *IEEE Robotics & Automation Magazine* 11(1), pp. 19–25, 40, 2004.

[155] A. S. Rao and M. P. Georgeff. “Modeling Rational Agents within a BDI-architecture”. In: *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR 1991)*. Ed. by J. Allen, R. J. Brachman, E. Sandewall, H. J. Levesque, R. Reiter, and R. Fikes, pp. 473–484. Cambridge, Massachusetts, USA, 1991.

[156] A. S. Rao and M. P. Georgeff. “BDI Agents: From Theory to Practice”. In: *Proceedings of the 1st International Conference on Multiagent Systems (ICMAS 1995)*, pp. 312–319. San Francisco, California, USA, 1995.

[157] H. L. Richardson. “Cross Docking: Information flow saves space”. *Transportation & Distribution* 40(11), pp. 51–54, 1999.

[158] *Road Freight Transport Vademecum 2010 Report – Market trends and structure of the road haulage sector in the EU in 2010*. European Commission, Directorate-General for Mobility and Transport, Unit D.3 – Land Transport. `http://ec.europa.eu/transport/modes/road/haulage/market_developments_en.htm`. 2011.

[159] M. Rohrer. “Simulation and cross docking”. In: *Proceedings of the 27th conference on Winter simulation*. Ed. by C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, pp. 846–849. Arlington, Virginia, USA, 1995.

[160] H. B. Roos. *Van marketentster tot logistiek netwerk: de militaire logistiek door de eeuwen heen*. Boom, 2002.

[161] C. R. Rosales, M. J. Fry, and R. Radhakrishnan. “Transfreight Reduces Costs and Balances Workload at Georgetown Crossdock”. *Interfaces* 39(4), pp. 316–328, 2009.

[162] A. Ross and V. Jayaraman. “An evaluation of new heuristics for the location of cross-docks distribution centers in supply chain network design”. *Computers & Industrial Engineering* 55(1), pp. 64–79, 2008.

[163] R. Sadykov. “Scheduling incoming and outgoing trucks at cross docking terminals to minimize the storage cost”. *Annals of Operations Research* 201(1), pp. 423–440, 2012.

[164] B. Saenz de Ugarte, A. Artiba, and R. Pellerin. “Manufacturing execution system – a literature review”. *Production Planning & Control: The Management of Operations* 20(6), pp. 525–539, 2009.

[165] B. Saint Germain. “Distributed coordination and control for networked production systems”. PhD thesis. Katholieke Universiteit Leuven, 2010.

[166] B. Saint Germain, P. Valckenaers, J. Van Belle, P. Verstraete, and H. Van Brussel. “Incorporating trust in networked production systems”. *Journal of Intelligent Manufacturing* 23(6), pp. 2635–2646, 2012.

[167] B. Saint Germain, P. Valckenaers, H. Van Brussel, Hadeli, O. Bochmann, C. Zamfirescu, and P. Verstraete. "Multi-agent manufacturing control: an industrial case study". In: *Intelligent Manufacturing Systems 2003, A Proceedings volume from the 7th IFAC Workshop*. Ed. by L. Monostori, B. Kádár, and G. Morel, pp. 207–212. Budapest, Hungary, 2003.

[168] B. Saint Germain, P. Valckenaers, P. Verstraete, Hadeli, and H. Van Brussel. "A multi-agent supply network control framework". *Control Engineering Practice* 15(11). Special Issue on Manufacturing Plant Control: Challenges and Issues, pp. 1394–1402, 2007.

[169] B. Saint Germain, P. Valckenaers, P. Verstraete, J. Van Belle, and H. Van Brussel. "Decision making in the supply network". In: *Preprints of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2009)*, pp. 2079–2084. Moscow, Russia, 2009.

[170] S. Sandal. "Staging approaches to reduce overall cost in a crossdock environment". Master's thesis. University of Missouri, 2005.

[171] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. Brueckner. "Evolving Adaptive Pheromone Path Planning Mechanisms". In: *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pp. 434–440. Bologna, Italy, 2002.

[172] B. Schaffer. "Implementing a Successful Crossdocking Operation". *IIE Solutions* 29(10), pp. 34–36, 1997.

[173] J. J. Schneider and S. Kirkpatrick. *Stochastic Optimization*. Scientific Computation. Springer, 2006.

[174] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos. "Self-organization in multi-agent systems". *The Knowledge Engineering Review* 20(2), pp. 165–189, 2005.

[175] M. Shakeri, M. Y. H. Low, and Z. Li. "A Generic Model for Cross-dock Truck Scheduling and Truck-to-Door Assignment Problems". In: *Proceedings of the 6th IEEE International Conference on Industrial Informatics (INDIN 2008)*, pp. 857–864. Daejeon, Korea, 2008.

[176] D. E. Shobrys and D. C. White. "Planning, scheduling and control systems: why cannot they work together". *Computers & Chemical Engineering* 26(2), pp. 149–160, 2002.

[177] H. A. Simon. *The Sciences Of The Artificial*. MIT Press, 1969.

[178] R. Soltani and S. J. Sadjadi. "Scheduling trucks in cross-docking systems: A robust meta-heuristics approach". *Transportation Research Part E: Logistics and Transportation Review* 46(5), pp. 650–666, 2010.

[179] K. Sörensen and M. Sevaux. “A Practical Approach for Robust and Flexible Vehicle Routing Using Metaheuristics and Monte Carlo Sampling”. *Journal of Mathematical Modelling and Algorithms* 8(4), pp. 387–407, 2009.

[180] G. Stalk, P. Evans, and L. E. Shulman. “Competing on Capabilities: The New Rules of Corporate Strategy”. *Harvard Business Review* 70(2), pp. 57–69, 1992.

[181] J. Stelling, U. Sauer, Z. Szallasi, F. J. Doyle III, and J. Doyle. “Robustness of Cellular Functions”. *Cell* 118(6), pp. 675–685, 2004.

[182] M. Stickel. “Planung und Steuerung von Crossdocking-Zentren”. Dissertation. Institut für Fördertechnik und Logistiksysteme, Fakultät für Maschinenbau, Universität Karlsruhe, 2006.

[183] P. P. M. Stoop and V. C. S. Wiers. “The complexity of scheduling in practice”. *International Journal of Operations & Production Management* 16(10), pp. 37–53, 1996.

[184] C. S. Sung and S. H. Song. “Integrated service network design for a cross-docking supply chain network”. *Journal of the Operational Research Society* 54(12), pp. 1283–1295, 2003.

[185] C. S. Sung and W. Yang. “An exact algorithm for a cross-docking supply chain network design problem”. *Journal of the Operational Research Society* 59(1), pp. 119–136, 2008.

[186] I. Sungur, F. Ordonez, and M. Dessouky. “A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty”. *IIE Transactions* 40(5), pp. 509–523, 2008.

[187] S.-L. Tang and H. Yan. “Pre-distribution vs. post-distribution for cross-docking with transshipments”. *Omega* 38(3-4), pp. 192–202, 2010.

[188] L. Y. Tsui and C.-H. Chang. “A microcomputer based decision support tool for assigning dock doors in freight yards”. *Computers & Industrial Engineering* 19(1-4), pp. 309–312, 1990.

[189] L. Y. Tsui and C.-H. Chang. “An optimal solution to a dock door assignment problem”. *Computers & Industrial Engineering* 23(1-4), pp. 283–286, 1992.

[190] B. Vahdani, R. Soltani, and M. Zandieh. “Scheduling the truck holdover recurrent dock cross-dock problem using robust meta-heuristics”. *The International Journal of Advanced Manufacturing Technology* 46(5), pp. 769–783, 2010.

[191] B. Vahdani and M. Zandieh. “Scheduling trucks in cross-docking systems: Robust meta-heuristics”. *Computers & Industrial Engineering* 58(1), pp. 12–24, 2010.

[192] P. Valckenaers, F. Bonneville, H. Van Brussel, L. Bongaerts, and J. Wyns. "Results of the Holonic Control System Benchmark at KULeuven". In: *Proceedings of the 4th International Conference on Computer Integrated Manufacturing and Automation Technology*, pp. 128–133. Troy, New York, USA, 1994.

[193] P. Valckenaers and H. Van Brussel. "Holonic Manufacturing Execution Systems". *CIRP Annals - Manufacturing Technology* 54(1), pp. 427–432, 2005.

[194] P. Valckenaers, Hadeli, B. Saint Germain, P. Verstraete, J. Van Belle, and H. Van Brussel. "From Intelligent Agents to Intelligent Beings". In: *Holonic and Multi-Agent Systems for Manufacturing*, pp. 17–26. Ed. by V. Mařík, V. Vyatkin, and A. W. Colombo. Lecture Notes in Artificial Intelligence 4659. Springer, 2007.

[195] P. Valckenaers, Hadeli, B. Saint Germain, P. Verstraete, and H. Van Brussel. "Emergent short-term forecasting through ant colony engineering in coordination and control systems". *Advanced Engineering Informatics* 20(3), pp. 261–278, 2006.

[196] P. Valckenaers, Hadeli, B. Saint Germain, P. Verstraete, and H. Van Brussel. "MAS coordination and control based on stigmergy". *Computers in Industry* 58(7), pp. 621–629, 2007.

[197] P. Valckenaers, T. Heikkilä, H. Baumgaertel, D. McFarlane, and J.-P. Courtois. "Towards a Novel Manufacturing Control Principle". In: *Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*. Ed. by H. Van Brussel and P. Valckenaers, pp. 871–875. Leuven, Belgium, 1999.

[198] P. Valckenaers and T. Holvoet. "The Environment: An Essential Abstraction for Managing Complexity in MAS-Based Manufacturing Control". In: *Environments for Multi-Agent Systems II*, pp. 205–217. Ed. by D. Weyns, H. V. D. Parunak, and F. Michel. Lecture Notes in Artificial Intelligence 3830. Springer, 2006.

[199] P. Valckenaers, B. Saint Germain, P. Verstraete, J. Van Belle, Hadeli, and H. Van Brussel. "Intelligent products: Agere versus Essere". *Computers in Industry* 60(3), pp. 217–228, 2009.

[200] P. Valckenaers and H. Van Brussel. "Intelligent Products: Intelligent Beings or Agents?" In: *Innovation in Manufacturing Networks*, pp. 295–302. Ed. by A. Azevedo. IFIP Advances in Information and Communication Technology 266. Springer, 2008.

[201] P. Valckenaers, H. Van Brussel, H. Bruyninckx, B. Saint Germain, J. Van Belle, and J. Philips. "Predicting the unexpected". *Computers in Industry* 62(6). Special Issue: Grand Challenges for Discrete Event Logistics Systems, pp. 623–637, 2011.

[202] P. Valckenaers, H. Van Brussel, Hadeli, O. Bochmann, B. Saint Germain, and Zamfirescu. "On the design of emergent systems: an investigation of integration and interoperability issues". *Engineering Applications of Artificial Intelligence* 16(4), pp. 377–393, 2003.

[203] P. Valckenaers, H. Van Brussel, P. Verstraete, B. Saint Germain, and Hadeli. "Schedule execution in autonomic manufacturing execution systems". *Journal of Manufacturing Systems* 26(2), pp. 75–84, 2007.

[204] J. E. Van Aken. "On the control of complex industrial organizations". PhD thesis. Technische Hogeschool Eindhoven, 1978.

[205] J. Van Belle, J. Philips, O. Ali, B. Saint Germain, H. Van Brussel, and P. Valckenaers. "A Service-Oriented Approach for Holonic Manufacturing Control and Beyond". In: *Service Orientation in Holonic and Multi-Agent Manufacturing Control*, chap. 1, pp. 1–20. Ed. by T. Borangiu, A. Thomas, and D. Trentesaux. Studies in Computational Intelligence 402. Springer, 2012.

[206] J. Van Belle, B. Saint Germain, P. Verstraete, P. Valckenaers, O. Ali, H. Van Brussel, and D. Cattrysse. "A Holonic Chain Conveyor Control System: An Application". In: *Holonic and Multi-Agent Systems for Manufacturing*, pp. 234–243. Ed. by V. Mařík, T. Strasser, and A. Zoitl. Lecture Notes in Artificial Intelligence 5696. Springer, 2009.

[207] J. Van Belle, P. Valckenaers, and D. Cattrysse. "Cross-docking: State of the art". *Omega* 40(6). Special Issue on Forecasting in Management Science, pp. 827–846, 2012.

[208] J. Van Belle, P. Valckenaers, B. Saint Germain, R. Bahtiar, and D. Cattrysse. "Bio-Inspired Coordination and Control in Self-Organizing Logistic Execution Systems". In: *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN 2011)*, pp. 713–718. Caparica, Lisbon, Portugal, 2011.

[209] J. Van Belle, P. Valckenaers, G. Vanden Berghe, and D. Cattrysse. "A Tabu Search Approach to the Truck Scheduling Problem with Multiple Docks and Time Windows". *Computers & Industrial Engineering*. Forthcoming.

[210] H. Van Brussel, L. Bongaerts, J. Wyns, P. Valckenaers, and T. Van Ginderachter. "A Conceptual Framework for Holonic Manufacturing: Identification of Manufacturing Holons". *Journal of Manufacturing Systems* 18(1), pp. 35–52, 1999.

[211] H. Van Brussel, P. Valckenaers, J. Wyns, P. Peeters, and L. Bongaerts. "Holonic manufacturing systems: architectural and manufacturing control issues". In: *Proceedings of 2nd CIRP International Seminar on Intelligent Computation in Manufacturing Engineering (ICME 2000)*, pp. 19–29. Capri, Italy, 2000.

[212] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. "Reference Architecture for Holonic Manufacturing Systems: PROSA". *Computers in Industry* 37(3). Special issue on intelligent manufacturing systems, pp. 255–274, 1998.

[213] T. Van Woensel, L. Kerbache, H. Peremans, and N. Vandaele. "Vehicle routing with dynamic travel times: A queueing approach". *European Journal of Operational Research* 186(3), pp. 990–1007, 2008.

[214] O. Ventä. *Intelligent Products and Systems*. Technology theme – Final report 635. VTT Technical Research Centre of Finland, 2007.

[215] P. Verstraete, P. Valckenaers, H. Van Brussel, B. Saint Germain, Hadeli, and J. Van Belle. "Towards robust and efficient planning execution". *Engineering Applications of Artificial Intelligence* 21(3), pp. 304–314, 2008.

[216] P. Verstraete. "Integrating existing scheduling techniques into the Holonic Manufacturing Execution System". PhD thesis. Katholieke Universiteit Leuven, 2009.

[217] P. Verstraete, B. Saint Germain, K. Hadeli, P. Valckenaers, and H. Van Brussel. "On applying the PROSA reference architecture in multi-agent manufacturing control applications". In: *Proceedings of the 5th European Workshop on Multi-Agent Systems (EUMAS 2007)*, pp. 126–143. Hammamet, Tunisia, 2007.

[218] P. Verstraete, B. Saint Germain, P. Valckenaers, H. Van Brussel, J. Van Belle, and Hadeli. "Engineering manufacturing control systems using PROSA and delegate MAS". *International Journal of Agent-Oriented Software Engineering* 2(1), pp. 62–89, 2008.

[219] I. F. Vis and K. J. Roodbergen. "Positioning of goods in a cross-docking environment". *Computers & Industrial Engineering* 54(3), pp. 677–689, 2008.

[220] K. Vitasek. *Supply Chain Management Terms and Glossary*. `http://cscmp.org/resources-research/glossary-terms`. 2010.

[221] M. A. Waller, C. R. Cassady, and J. Ozment. "Impact of cross-docking on inventory in a decentralized retail supply chain". *Transportation Research Part E: Logistics and Transportation Review* 42(5), pp. 359–382, 2006.

[222] J.-F. Wang and A. Regan. "Real-Time Trailer Scheduling for Crossdock Operations". *Transportation Journal* 47(2), pp. 5–20, 2008.

[223] M. Wen, J. Larsen, J. Clausen, J.-F. Cordeau, and G. Laporte. "Vehicle routing with cross-docking". *Journal of the Operational Research Society* 60(12), pp. 1708–1718, 2009.

[224] B. Werners and T. Wülfing. "Robust optimization of internal transports at a parcel sorting center operated by Deutsche Post World Net". *European Journal of Operational Research* 201(2), pp. 419–426, 2010.

[225] D. Weyns. "An Architecture-Centric Approach for Software Engineering with Situated Multiagent Systems". PhD thesis. Katholieke Universiteit Leuven, 2006.

[226] W. L. Winston. *Operations Research: Applications and Algorithms*. 4th ed. Duxbury Press, 2004.

[227] C. E. Witt. "Crossdocking: Concepts demand choice". *Material Handling Engineering* 53(7), pp. 44–49, 1998.

[228] C. Y. Wong, D. McFarlane, A. A. Zaharudin, and V. Agarwal. "The Intelligent Product Driven Supply Chain". In: *Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics*. Vol. 4. Hammamet, Tunisia, 2002.

[229] M. Wooldridge. *An Introduction to MultiAgent Systems*. 2nd ed. Wiley, 2009.

[230] M. Wooldridge and N. R. Jennings. "Intelligent agents: theory and practice". *The Knowledge Engineering Review* 10(2), pp. 115–152, 1995.

[231] J. Wyns. "Reference Architecture for Holonic Manufacturing Systems: the key to support evolution and reconfiguration". PhD thesis. Katholieke Universiteit Leuven, 1999.

[232] H. Yan and S.-l. Tang. "Pre-distribution and post-distribution cross-docking operations". *Transportation Research Part E: Logistics and Transportation Review* 45(6), pp. 843–859, 2009.

[233] V. F. Yu, D. Sharma, and K. G. Murty. "Door Allocations to Origins and Destinations at Less-than-Truckload Trucking Terminals". *Journal of Industrial and Systems Engineering* 2(1), pp. 1–15, 2008.

[234] W. Yu and P. J. Egbelu. "Scheduling of inbound and outbound trucks in cross docking systems with temporary storage". *European Journal of Operational Research* 184(1), pp. 377–396, 2008.

[235] C. Zamfirescu, P. Valckenaers, Hadeli, H. Van Brussel, and B. Saint Germain. "A Case Study for Modular Plant Control". In: *Holonic and Multi-Agent Systems for Manufacturing*, pp. 268–279. Ed. by V. Mařík, D. McFarlane, and P. Valckenaers. Lecture Notes in Artificial Intelligence 2744. Springer, 2003.

Curriculum vitae

Jan Van Belle was born in Assebroek (Bruges), Belgium on September 28, 1983. He received the master's degree of electrotechnical engineering (option ICT: Telecommunication and Telematics), cum laude from the KU Leuven, Belgium in July 2006.

In October 2006, he joined the Multi-Agent Coordination and Control (MACC) research group of the Mechanical Engineering Department of the KU Leuven. Until September 2008, he worked as a project engineer and was involved in the IOF project HFID (Holonische Fabrieksbesturingen - Industriële Demonstratie) and the IWT project ELC^2 (Egemin Logistic Chain Concept). From then on, he started his doctoral study under the supervision of prof. dr. ir. Dirk Cattrysse and dr. ir. Paul Valckenaers. His current research interests are multi-agent coordination and control in manufacturing and logistic systems.

During this period, he was daily supervisor and assessor of several master's theses. He also supervised student projects for the course 'Problem Solving and Engineering Design' ('Probleemoplossen en Ontwerpen') and was teaching assistant for 'Decision Making' ('Besliskunde').

In 2012, he was a member of the organizing committee of the bMES (benchmarking Manufacturing Execution Systems) workshop at the 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2012, Bucharest, Romania).

List of Publications

Articles in internationally reviewed academic journals

[1] **J. Van Belle**, P. Valckenaers, G. Vanden Berghe, and D. Cattrysse. "A Tabu Search Approach to the Truck Scheduling Problem with Multiple Docks and Time Windows". *Computers & Industrial Engineering*. Forthcoming.

[2] O. Ali, P. Valckenaers, **J. Van Belle**, B. Saint Germain, P. Verstraete, and D. Van Oudheusden. "Towards online planning for open-air engineering processes". *Computers in Industry* 64(3), pp. 242–251, 2013.

[3] B. Saint Germain, P. Valckenaers, **J. Van Belle**, P. Verstraete, and H. Van Brussel. "Incorporating trust in networked production systems". *Journal of Intelligent Manufacturing* 23(6), pp. 2635–2646, 2012.

[4] **J. Van Belle**, P. Valckenaers, and D. Cattrysse. "Cross-docking: State of the art". *Omega* 40(6). Special Issue on Forecasting in Management Science, pp. 827–846, 2012.

[5] B. Saint Germain, P. Valckenaers, H. Van Brussel, and **J. Van Belle**. "Networked Manufacturing Control: an Industrial Case". *CIRP Journal of Manufacturing Science and Technology* 4(3), pp. 324–326, 2011.

[6] P. Valckenaers, H. Van Brussel, H. Bruyninckx, B. Saint Germain, **J. Van Belle**, and J. Philips. "Predicting the unexpected". *Computers in Industry* 62(6). Special Issue: Grand Challenges for Discrete Event Logistics Systems, pp. 623–637, 2011.

[7] P. Valckenaers, B. Saint Germain, P. Verstraete, **J. Van Belle**, Hadeli, and H. Van Brussel. "Intelligent products: Agere versus Essere". *Computers in Industry* 60(3), pp. 217–228, 2009.

[8] P. Verstraete, P. Valckenaers, H. Van Brussel, B. Saint Germain, Hadeli, and **J. Van Belle**. "Towards robust and efficient planning execution". *Engineering Applications of Artificial Intelligence* 21(3), pp. 304–314, 2008.

[9] P. Verstraete, B. Saint Germain, P. Valckenaers, H. Van Brussel, **J. Van Belle**, and Hadeli. "Engineering manufacturing control systems using PROSA and delegate MAS". *International Journal of Agent-Oriented Software Engineering* 2(1), pp. 62–89, 2008.

Articles in academic books, published by internationally recognized scientific publishers

[10] J. M. Novas, **J. Van Belle**, B. Saint Germain, and P. Valckenaers. "A Collaborative Framework between a Scheduling System and a Holonic Manufacturing Execution System". In: *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*, chap. 1, pp. 3–17. Ed. by T. Borangiu, A. Thomas, and D. Trentesaux. Studies in Computational Intelligence 472. Springer, 2013.

[11] **J. Van Belle**, J. Philips, O. Ali, B. Saint Germain, H. Van Brussel, and P. Valckenaers. "A Service-Oriented Approach for Holonic Manufacturing Control and Beyond". In: *Service Orientation in Holonic and Multi-Agent Manufacturing Control*, chap. 1, pp. 1–20. Ed. by T. Borangiu, A. Thomas, and D. Trentesaux. Studies in Computational Intelligence 402. Springer, 2012.

Papers at international scientific conferences and symposia, published in full in proceedings

[12] J. Philips, B. Saint Germain, **J. Van Belle**, and P. Valckenaers. "Computational Complexity and Scalability Analysis of PROSA and Delegate MAS". In: *Proceedings of the 11th IFAC Workshop on Intelligent Manufacturing Systems (IMS 2013)*, pp. 41–46. São Paulo, Brazil, 2013.

[13] J. Philips, B. Saint Germain, **J. Van Belle**, and P. Valckenaers. "Traffic Radar: A Holonic Traffic Coordination System Using PROSA++ and D-MAS". In: *Industrial Applications of Holonic and Multi-Agent Systems. Proceedings of the 6th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS 2013)*. Ed. by V. Mařík,

J. L. M. Lastra, and P. Skobelev. Lecture Notes in Artificial Intelligence 8062, pp. 163–174. Prague, Czech Republic, 2013.

[14] **J. Van Belle**, B. Saint Germain, J. Philips, P. Valckenaers, and D. Cattrysse. "Cooperation between a Holonic Logistics Execution System and a Vehicle Routing Scheduling System". In: *Proceedings of the 11th IFAC Workshop on Intelligent Manufacturing Systems (IMS 2013)*, pp. 184–189. São Paulo, Brazil, 2013.

[15] J. M. Novas, R. Bahtiar, **J. Van Belle**, and P. Valckenaers. "An Approach for the Integration of a Scheduling System and a Multi-Agent Manufacturing Execution System. Towards a Collaborative Framework." In: *Preprints of the 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2012)*. Ed. by T. Borangiu and A. Dolgui, W258–W263. Bucharest, Romania, 2012.

[16] J. Philips, P. Valckenaers, E. Aertbeliën, **J. Van Belle**, B. Saint Germain, H. Bruyninckx, and H. Van Brussel. "PROSA and Delegate MAS in Robotics". In: *Holonic and Multi-Agent Systems for Manufacturing. Proceedings of the 5th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS 2011)*. Ed. by V. Mařík, P. Vrba, and P. Leitão. Lecture Notes in Artificial Intelligence 6867, pp. 195–204. Toulouse, France, 2011.

[17] P. Valckenaers, **J. Van Belle**, and O. Ali. "PROSA and Delegate MAS for Open-Air Engineering Processes". In: *Proceedings of the 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011)*. Ed. by Z. Mammeri. Toulouse, France, 2011.

[18] **J. Van Belle**, B. Saint Germain, P. Valckenaers, H. Van Brussel, R. Bahtiar, and D. Cattrysse. "Intelligent Products in the Supply Chain Are Merging Logistic and Manufacturing Operations". In: *Preprints of the 18th IFAC World Congress*. Ed. by S. Bittanti, A. Cenedese, and S. Zampieri, pp. 1596–1601. Milano, Italy, 2011.

[19] **J. Van Belle**, P. Valckenaers, B. Saint Germain, R. Bahtiar, and D. Cattrysse. "Bio-Inspired Coordination and Control in Self-Organizing Logistic Execution Systems". In: *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN 2011)*, pp. 713–718. Caparica, Lisbon, Portugal, 2011.

[20] O. Ali, B. Saint Germain, **J. Van Belle**, P. Valckenaers, H. Van Brussel, and J. Van Noten. "Multi-Agent Coordination and Control System for Multi-Vehicle Agricultural Operations". In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*. Ed. by W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen, pp. 1621–1622. Toronto, Canada, 2010.

[21] R. Bahtiar, **J. Van Belle**, B. Saint Germain, P. Verstraete, P. Valckenaers, H. Van Brussel, and D. Cattrysse. "Cooperation between a Holonic Schedule Execution System and the i2 Factory Planner". In: *10th IFAC Workshop on Intelligent Manufacturing Systems (IMS 2010), Preprints*. Ed. by P. Leitão, C. E. Pereira, and J. Barata, pp. 99–104. Lisbon, Portugal, 2010.

[22] P. Valckenaers, H. Van Brussel, B. Saint Germain, and **J. Van Belle**. "Networked Manufacturing Control: an Industrial Case". In: *Sustainable Production and Logistics in Global Networks, 43rd CIRP International Conference on Manufacturing Systems (ICMS 2010), Proceedings*. Ed. by W. Sihn and P. Kuhlang, pp. 129–136. Vienna, Austria, 2010.

[23] P. Verstraete, P. Valckenaers, H. Van Brussel, B. Saint Germain, **J. Van Belle**, and R. Bahtiar. "Schedule execution by a Holonic Manufacturing Execution System". In: *Proceedings of the International Conference on Competitive Manufacturing (COMA 2010)*. Ed. by D. Dimitrov, pp. 349–354. Stellenbosch, South Africa, 2010.

[24] B. Saint Germain, P. Valckenaers, P. Verstraete, **J. Van Belle**, and H. Van Brussel. "Decision making in the supply network". In: *Preprints of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2009)*, pp. 2079–2084. Moscow, Russia, 2009.

[25] **J. Van Belle**, B. Saint Germain, P. Verstraete, P. Valckenaers, O. Ali, H. Van Brussel, and D. Cattrysse. "A Holonic Chain Conveyor Control System: An Application". In: *Holonic and Multi-Agent Systems for Manufacturing. Proceedings of the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS 2009)*. Ed. by V. Mařík, T. Strasser, and A. Zoitl. Lecture Notes in Artificial Intelligence 5696, pp. 234–243. Linz, Austria, 2009.

[26] Hadeli, P. Valckenaers, H. Van Brussel, B. Saint Germain, P. Verstraete, and **J. Van Belle**. "Towards the Design of Autonomic Nervousness Handling in Holonic Manufacturing Execution Systems". In: *Proceedings of the 2007 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2883–2888. Montreal, Canada, 2007.

[27] Hadeli, P. Valckenaers, H. Van Brussel, P. Verstraete, B. Saint Germain, and **J. Van Belle**. "Production planning and control in bio-inspired holonic manufacturing execution systems". In: *Preprints of the 8th IFAC International Workshop on Intelligent Manufacturing Systems (IMS 2007)*, pp. 48–55. Alicante, Spain, 2007.

[28] P. Valckenaers, Hadeli, B. Saint Germain, P. Verstraete, **J. Van Belle**, and H. Van Brussel. "From Intelligent Agents to Intelligent Beings". In: *Holonic and Multi-Agent Systems for Manufacturing. Proceedings of the*

3rd International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS 2007). Ed. by V. Mařík, V. Vyatkin, and A. W. Colombo. Lecture Notes in Artificial Intelligence 4659, pp. 17–26. Regensburg, Germany, 2007.

[29] P. Verstraete, P. Valckenaers, H. Van Brussel, B. Saint Germain, Hadeli, and **J. Van Belle**. "On the performance of a holonic manufacturing execution system in case of an unfeasible planning due to machine breakdown". In: *Proceedings of the 40th CIRP International Manufacturing Systems Seminar*. Liverpool, UK, 2007.

Papers at other scientific conferences and symposia, published in full in proceedings

[30] **J. Van Belle**, B. Saint Germain, R. Bahtiar, P. Valckenaers, H. Van Brussel, and D. Cattrysse. "Towards holonic control for cross-docks". In: *Bijdragen vervoerslogistieke werkdagen 2010*. Ed. by F. J. A. Witlox and S. Weijers, pp. 319–332. Antwerp, Belgium, 2010.

[31] **J. Van Belle**, B. Saint Germain, P. Verstraete, P. Valckenaers, O. Ali, H. Van Brussel, and D. Cattrysse. "Holonische besturingen in vervoerslogistieke systemen". In: *Bijdragen vervoerslogistieke werkdagen 2009*. Ed. by F. J. A. Witlox and W. Ploos van Amstel, pp. 519–533. Deurne, The Netherlands, 2009.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF MECHANICAL ENGINEERING
CENTRE FOR INDUSTRIAL MANAGEMENT / TRAFFIC AND INFRASTRUCTURE
Celestijnenlaan 300A box 2422
3001 Leuven
http://www.mech.kuleuven.be