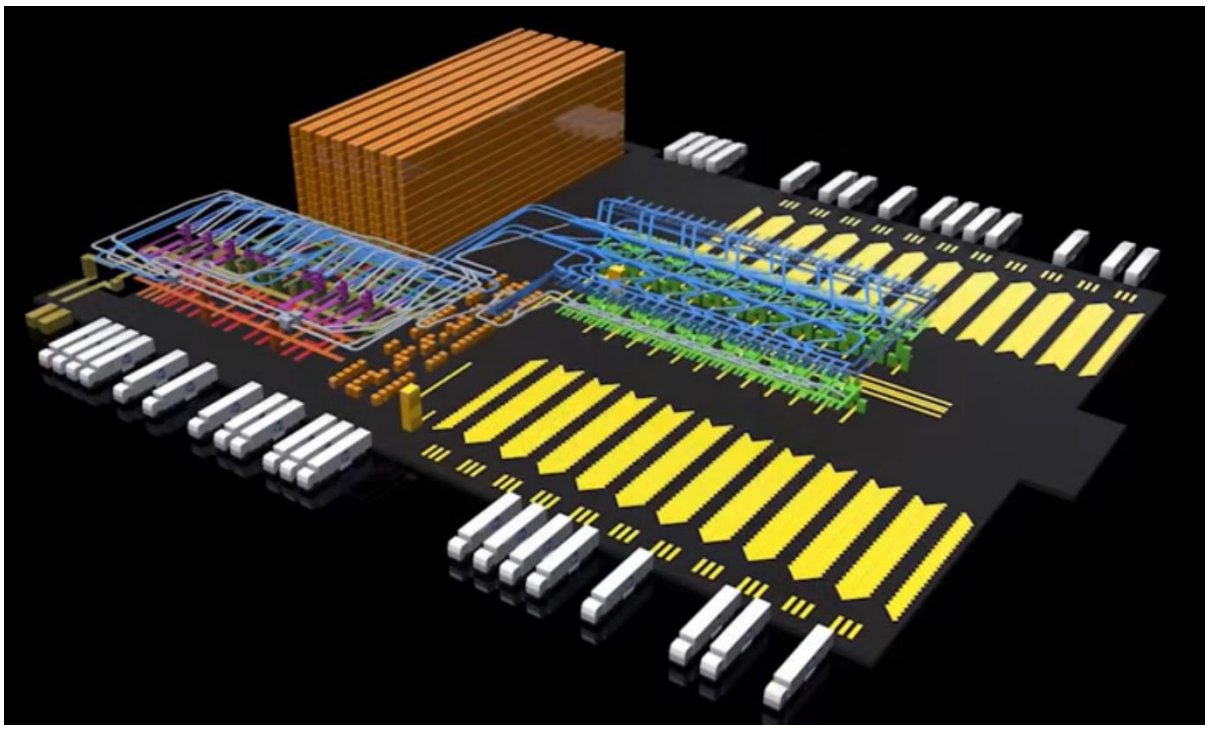
Automating the capacity planning process for a distribution centre

Comparing heuristic and integer programming approaches



Public version

November 2019

R. van Manen
Master thesis
University of Twente
Industrial Engineering & Management
Production & Logistic Management

Vanderlande
Supervisor: ir. Rob Burgers

University of Twente
Supervisor 1: Dr. ir. J.M.J. Schutten
Supervisor 2: Dr. P.C. Schuur

Preface

This thesis marks the end of my Master degree in Industrial engineering and Management at the University of Twente. Throughout the course of working on this thesis, I've received tremendous support from my supervisors and colleagues.

First of all, I would like to thank Rob Burgers for his excellence guidance at Vanderlande and providing excellent discussion and feedback. Also, I would like to thank the ACP team and more specifically Zowi for being a sparring partner in the early phase of this research. I would like to thank Paul Snijder from Albert Heijn especially for the guided tour and the insights in the planning process.

I thank Marco Schutten and Peter Schuur at the University of Twente. Their feedback and critical eye for detail has been valuable and contributed greatly especially in the last phase of my research.

Rik van Manen

Summary

This research considers the planning process of resource capacity in highly automated distribution centres. Vanderlande provides a planning application that planners can use to allocate the workstations and operators in distribution centres. Currently, this is a manual process that uses iteration between a change in the plan and feedback from a scheduling algorithm that simulates production. This process is time-consuming and often leads to suboptimal workstation allocations. We focus on reducing the time required for planning as well as reducing the number of weighed human operator time required to fulfil weekly demand for the distribution centre. The following research question is developed:

"How can effective capacity plans for distribution centres be made in reasonable time in the tactical planning phase?"

We find two approaches to make capacity plans: an iterative and an integrated strategy.

The iterative strategy covers a two-phase heuristic approach that iterates between a planning alteration and the scheduling function to control feasibility. The iterative strategy starts with an initial phase based on an adaptive search heuristic that removes shifts from a full capacity allocation iteratively based on the slack lead time of orders. Finally, these allocations are improved by a simulated annealing metaheuristic.

The integrated strategy uses an integer linear programming approach, covering both the planning and scheduling aspects of the problem. Due to the sheer size of the problem, the input parameters must be adapted slightly in order to be able to find feasible solutions. We find that while we adapt the input parameters, validity is maintained for all test instances.

For the focus distribution centre of this research, that of Albert Heijn in Zaandam, both the iterative and integrated strategy show significant improvements in cost compared to a manually created capacity plan. The integrated and iterative strategy show a mean reduction in operator cost of -16.9% and -14.9% respectively compared to a manually created plan. The reduction in operator hours including indirect personnel is -11.9% and -10.9% respectively.

Next to increased utilisation and lowered operator cost, the time required for planning is reduced significantly. These low-cost allocations are obtained within minutes. The reduction in time for planning and cost for fulfilling demand provide a solid basis for the adaptation of an automatic planner for the distribution centre in practice. The automatic planner could act as a basis for the planner in creating allocations.

We find that generally, the iterative strategy is more versatile and fit to practical problems. The integrated strategy requires several adaptations that happen to fit to our problem, for instance in the configuration of shifts and breaks. In practice, there may be many more shifts as well as variation in the length of breaks. These are hard to model in the integrated strategy. Finally, we find that the iterative strategy is more suitable for Vanderlande as it generally provides more reliable results and is more adaptable to differing input parameters and other distribution centres.

Table of contents

List of Figures

List of Tables

Abbreviations

ACP	**Automated case picking**
NC	Non conveyable
LFL	Load forming logic
TOP	Tactical and operational planning
LC	Load carrier
KPI	Key performance indicator
ICPA	Incremental capacity planning algorithm
NP-hard	Non-deterministic polynomial time hard
SO	Store order
TS	Tabu search
SA	Simulated annealing
GRASP	Greedy randomised adaptive search heuristic
ILP	Integer linear programming
LP	Linear programming
RCCP	Rough-cut capacity planning
RLP	Resource loading problem
DC	Distribution Centre
JIT	Just-in-time
RCPS	Resource-constrained project scheduling
AS	Adaptive search

Glossary

Term	*Description*	*Page introduced*
(internal) Order lead time	The estimated time from the start of outbound production until order departure time, determined by the scheduling function	15
Scheduling function	Priority rule used to assess capacity plan performance	29
Production area	A customer store order may demand load carriers from up to three different production areas	20
Production process	To fulfil demand for a production area, several processes in that area must be passed	20
Workstation	At each production process, a finite number of workstations can be activated to fulfil demand. Some of these workstations require (multiple) operators.	25
Capacity plan	The capacity plan indicates the availability of workstations over time in the planning week. The capacity plan enables the distribution centre to obtain an estimation of the number of operators required to fulfil demand	27
Operator shift	Some workstations require human operators. Human operators work for the duration of an operator shift.	26

1. Introduction

This thesis describes a master research conducted at Vanderlande, a material handling and logistics automation company based in Veghel, the Netherlands. Vanderlande focuses on three main domains: airports, warehousing and parcel. The focus of this research is on warehousing and more specifically distribution centres. The warehousing systems extend across a wide range of processes. For example, automated storage and retrieval system (AS/RS) and automated order-picking systems. Through a high level of automation in these distribution centres, Vanderlande customers can fulfil many picking orders with few human resources. While there are fully automated workstations at processes, numerous processes require human resources for operation.

Currently, Vanderlande equips customer distribution centres with a planning application that allows the customer distribution centres to plan the capacity by allocating workstations in ten-minute timeslots over a weekly horizon. With these capacity plans, they seek to fulfil all of their customer store orders with as few human operators as possible. However, the current planning process is time-consuming and tends to result in satisfactory rather than optimal capacity plans. This results in the inefficient use of human resources. Therefore, the goal of this research is to improve this planning process through automation.

This chapter introduces Vanderlande, the focus system of this research as well as the research problem and questions. Section 1.1 introduces the focus system of this research. Section 1.2 introduces the core problem. Section 1.3 considers the research design.

1.1. System description

Vanderlande systems arrange logistic process automation for distribution centres of retailers. These distribution centres produce load carriers that are ready for filling shelves in retailer stores from inbound pallets containing cases. Within these distribution centres, there is a high level of automation. One of these automation systems for retailers is referred to as Automated Case Picking (ACP).

Vanderlande's ACP systems provide the tools to outperform traditional order fulfilment methods. The ACP systems integrate bulk storage of pallets, depalletizing and tray loading, tray storage and roll cage building. ACP provides several novel solutions, including integrated automatic depalletizing, storage, palletising, internal transport and optimal pallet stacking solutions.

The system regards interaction between three main acting parties:

1. *(Planning) Users* – Planners or supervisors create capacity plans by allocating workstations as available to fulfil demand from customer stores and process inbound suppliers;
2. *Suppliers* – Inbound shipments arrive from suppliers. These shipments must be transformed by processes into units that are demanded by customer stores;
3. *Customer stores* – Customer stores demand load carriers from the distribution centre that must be ready by a truck's departure time. The distribution centre supplies these load carriers by processing inbound supply using the available workstations allocated by the planning application users.

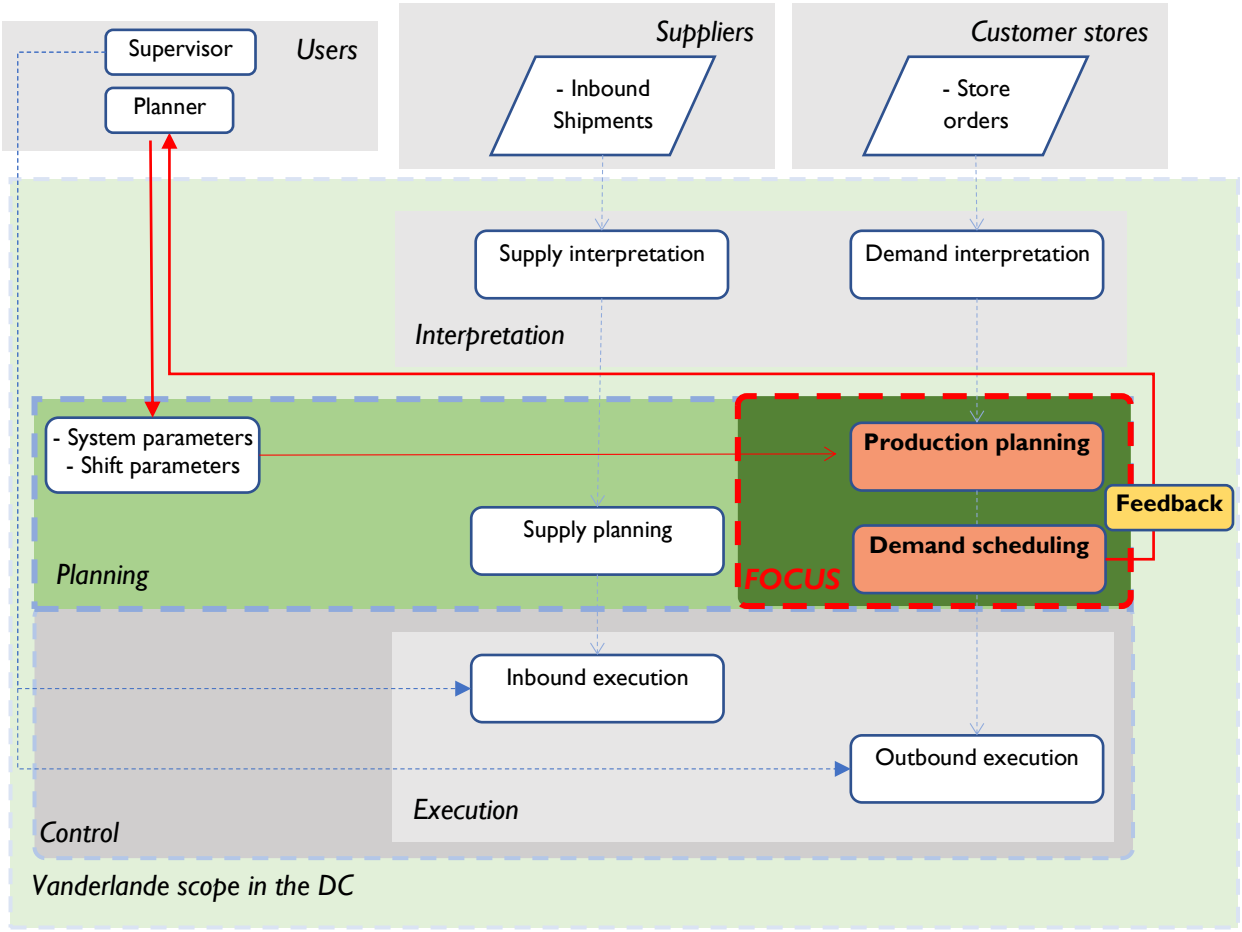


Figure 1 – Vanderlande scope in mechanised DC

Figure 1 distinguishes two main features in the Vanderlande scope in the distribution center: Planning and control. Planning refers to the processes that involve allocating capacity and more specifically workstations. Control refers to the processes involved in the execution of production on the day of production itself. This research focuses on the planning component. Section 1.1.1 introduces the elements of the system in more detail. Section 1.1.2 elaborates on the planning module in the distribution centre.

1.1.1. Planning and control

Planning and control in the mechanised distribution centres is required to fulfil outbound load carrier demand from stores before outbound trucks are expected to depart. In other words, all load carriers for store orders must be filled before the truck shipping them will depart. Figure 1 shows the Vanderlande scope in the mechanised DC. We distinguish three main components:

1. *Interpretation* – Inbound supply shipments as well as outbound demand must be interpreted so that they can be planned for. Customer stores may consist of hundreds of products spread over numerous load carriers. However, planning is not done for these products individually as it would complicate the process. Therefore, the demand for these products is interpreted and

translated to demand for load carriers from a select number of areas. Chapter 2 covers these areas in more detail.

2. *Planning* – Now that supply and demand have been interpreted to a format that can be handled for planning capacity, the planning application users can allocate workstations over different processes. By allocating these workstations, capacity is added to their respective timeslots in which they are assigned. The planning users create plans using shift parameters that indicate human resource availability as well as system parameters that contain information on the resources in the distribution centre. These plans can be evaluated using a scheduling function after which plans may be amended based on the feedback of the scheduling function.
3. *Execution* – Planners allocate workstation capacity over time with limited detail. For example, the planners state which workstations should be available at what moment, but they do not dedicate workstations to specific workstations. Also, planners plan on the level of aggregate processing units such as load carriers. The execution phase considers the control phase in the distribution centre with a higher level of specificity. The execution phase dedicates constituent cases to specific workstations on the day of production outside the planning scope.

Essentially, the interpretation phase generalises the demand so that it can be planned for. Then, the execution phase specifies demand again on the day of production to arrange the actual processing. The Vanderlande module responsible for planning is referred to as the Tactical and Operational planning module (TOP). The interpretation supply and demand are used in the TOP module for planning the resources used for producing orders. The output resource allocations from TOP are used for in- and outbound execution. The next section covers this module in more detail.

1.1.2. Planning module

The goal of the planning module is to plan the capacity that is required to cope with the customer store demand on the system at minimum cost. The capacity is determined by the amount of resources planned to be available over time. These resources include system resources (such as automated palletiser modules) and human resources (operators required for the use of several system resources). The demand for capacity includes both inbound (supply) and outbound (store) demand. Inbound demand consists of forecasted and actual shipping notifications for inbound goods into the warehouse. The outbound demand consists of the forecasted and actual customer store orders that the must fulfil.

Planning in the TOP module consists of two phases: the tactical and the operational planning phase.

- The **tactical planning** phase considers the scope of several weeks into the future. During this phase, a production planner determines the amount of system capacity required during a particular period to meet forecasted demand for that period. The production planner determines this amount by iteratively updating the resource allocation and simulating the resource plan for the tactical phase through a what-if loop by using a demand scheduling algorithm. The tactical plan serves as input for the operational planning process. The tactical plan specifies the level of system capacity over time for the production processes (e.g. depalletizing or picking) in the distribution centre.

- The **operational planning** phase considers a shorter outlook: up to a number of days in the future. During this phase, a production supervisor monitors the live performance through KPIs. They monitor whether the system capacity as planned in the tactical plan continues to meet operational demand, as generated by actual and forecasted demand. The production supervisor attempts to balance planned capacity against operational demand. The supervisor either makes minor adjustments to the planned capacity (insofar possible with human resource capacity availability on short notice) or the operational demand. The former is done by updating and simulating the resource plan for the operational phase i.e. the operational plan, through a "what-if" loop. The latter is done by altering the store order pool, e.g., cancelling store orders or parts of store orders.

The operational planning process also determines the start times for each store order (order plan). The start times define when a store order is released for execution in each production area (e.g. non-conveyables picking, or automated case picking). The start times, which result from the demand scheduling function, depend on the available capacity as specified in the operational plan, on the demand generated by actual and forecasted store orders and the production progress.

The start times and performance indications (such as order lead times) that result from the resource allocation in the demand scheduling function (introduced in Chapter 2) are not definitive. They serve as a simulation, or an attempt to model real-life situations to study how the system works (Law & Kelton, 2000). Definitive resource assignment of orders to workstations and order execution happens in the execution phase on the day of production.

1.2. Problem description

Section 1.1 describes that the planning module is used for planning the resources in the distribution centre. A capacity plan indicates for timeslots of 10 minutes how many workstations are available for processing at which processes in order to fulfil demand timely. Deciding how many workstations to allocate when is a complex decision as the decision space is vast, and subject to numerous constraints. There are tens of workstations over several processes that could be required during each timeslot.

While we describe the planning process done by production planners in more detail in Chapter 2, we show that the planning problem for each production week is complex as planners have to take into account numerous aspects simultaneously. For example, the complexity stems from:

- Precedence relations between processes: creates lack of transparency in improvement potential, as it may be unclear which processes are causing infeasible allocations.
- Meeting store order lead time constraints: Each store order introduces different constraints on the lead time (time spent from the start of outbound production until departure). Identifying which orders cause infeasibility is cumbersome. Moreover, indications of timeslots exhibiting improvement potential through capacity reduction becomes ambiguous.
- Identifying per process for which timeslots capacity should be added or removed: lead times tend to cause demand to shift forward in time, so improving in a timeslot with great lead time may have no effect as the lead times stem from a lack of allocated capacity in later timeslots.
- Operator (shifts) and machine availability: some workstations require a minimum ratio of human resources to be available in relation to the total number of system resources planned. Furthermore, human resources must be allocated in the length of shifts.

To plan resources, planners or supervisors use an iterative process. In this process, they use what-if analysis with a demand scheduling function that simulates a production schedule on a just-in-time basis, and calculates KPIs based on this schedule to simulate performance on the day of production. The performance indicators from this simulation are used for future optimization of resource plans. If performance indicators are not considered satisfactory, the production planner or supervisor continues to repeat a loop modifying resource allocation and then scheduling demand until they are satisfied. Once satisfied, they consider activating a modified resource plan, transitioning to the operational phase.

Figure 2 shows an overview of the planning process flow for the distribution centre resource availability. To plan capacity, input parameters are used. These parameters include internal and external parameters. Internal (distribution centre) parameters include amongst others workstation productivity rates and the availability of human resources. These parameters can be set by the distribution centre itself. External actors such as customer stores provide external parameters in the form of store orders. These orders contain information on the departure times and request number of items per production area.

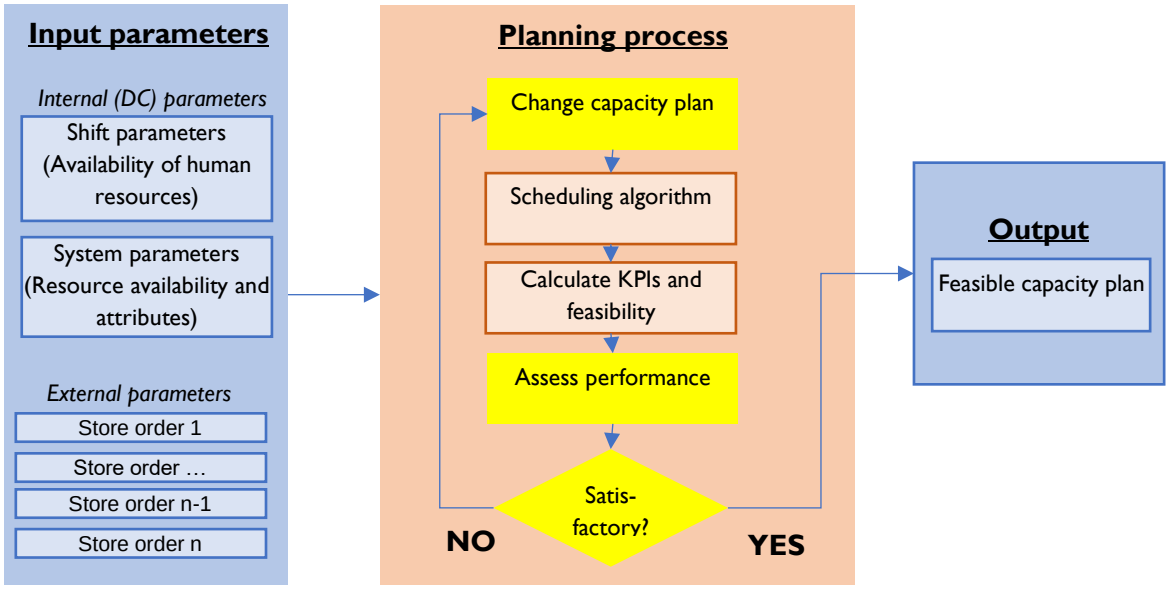


Figure 2 – Planning process flow

The goal of the planning process is to create a plan that indicates workstation available over a week that minimizes the use of resources while fulfilling all customer store orders. Figure 2 highlights human (planner) actions in yellow. The current (manual) planning process broadly includes the following steps:

1. The capacity over time is altered by the human planner. For instance, (multiple) workstations at a process may be removed or added for a time interval. This impacts the capacity availability to process store order demand.
2. To check the effect of a change in the capacity plan, the automatic scheduling algorithm is used. It uses a priority rule that schedules demand recursively from the store order departure times.
3. After scheduling, KPI values for the given capacity plan are generated based on the scheduling algorithm. Moreover, feasibility can be assessed.

4. The human planner assesses the performance of capacity plans given KPIs generated by the scheduling algorithm.
5. If the capacity plan is feasible (that is: all store orders handled within time constraints), then the human planner decides to either implement the current plan (if it is deemed satisfactory) or continue iterating in the loop when the performance is considered insufficient. This eventually creates a feasible capacity plan.

This what-if process is problematic for two main reasons: it is time consuming and unlikely to result in (near) optimal solutions in reasonable amount of iterations. We describe these problems briefly. The reason for the planning process being time consuming is that each modification in the allocation of workstations implies a change in performance. To check this performance, production for store orders must be simulated. Next to this, performance of a capacity plan is highly constrained, as we describe in Chapter 2. Furthermore, the optimal level of an allocation is currently hard to determine as analysis of plans is required and limited insight in improvement can be gained in the planning application. The result of these factors, combined with the fact that planning process is not trivial, is that a planner accepts a satisfactory performance level whereas they seek optimal rather than satisfactory performance.

1.3. Research Design

This section covers the research design. We describe the scope of the research as well as the problem and corresponding questions we answer to tackle the problem. Finally, we indicate which deliverables result from this research.

1.3.1. Scope

This thesis considers the planning module in the planning and control of Vanderlande's scope in distribution centres. This means that we only consider planning actions. Control actions such as cancelling or modifying of orders and order acceptance are not part of the scope of this research. We consider the tactical planning phase, as this is the phase wherein we can still have impact on the number of human resources to be allocated.

The planning module is used in several distribution centres, however the scope of this research considers the Albert Heijn DC. All examples given and experiments performed are done for the Albert Heijn DC. Therefore, the parameters such as resources and production flow are specific to this customer.

1.3.2. Research problem

In Section 1.2 we gave a description of the current problem. We found that the current tactical planning process is lacking as it is time-consuming and unlikely to be optimal. To solve this problem we consider the main research question:

How can effective capacity plans for the distribution centre be made in reasonable time in the tactical planning phase?

To answer the main question, we must first of all get insights in the way plans are created, and which (tacit) constraints and performance indicators are used to create and validate plans. Therefore, Chapter 2 serves to answer research question (1).

1. **How are capacity plans currently made for the distribution centre, and how is their performance assessed?**
    - 1.1 What does the process flow in the DC look like?
    - 1.2 What part of the distribution centre is subject to tactical capacity planning?
    - 1.3 What do the inputs and outputs for the planning look like (what information is used)?
    - 1.4 What does the planning process currently look like?
    - 1.5 Which performance indicators and constraints are used and how is performance assessed?
    - 1.6 How is performance assessed?

We seek to gain insight in the process flow, the processes to plan for and the inputs and outputs. The main research questions yields some ambiguity in the adjectives *optimal* and *reasonable*. We require definitions and trade-offs between performance indicators to be able to optimize for some variables.

To gain insight in capacity plan evaluation methodology, we consider questions (1.5) and (1.6). First of all, we want to determine how performance is calculated. As we described in Section 1.1.1, production is scheduled as a simulation of the production day, to evaluate the capacity plan. We seek to obtain information on how this scheduling is done. Furthermore, to determine the acceptable level of KPIs referred to in Chapter 2, we require information on which KPIs are used. Next to that, we want to find out which (tacit) constraints are used to determine acceptable KPIs so that we can model them in our approach. At last, the level of 'acceptable' performance in this context is ambiguous, so we seek to find a more concrete definition.

As we obtain insights in the performance of capacity plans, we require knowledge on how to create performant capacity plans automatically. As insights in process and performance are obtained through the previous research questions, we seek related problems in literature and answer question (2) in Chapter 3. We review capacity planning and scheduling positioning frameworks and problems (2.1) and approaches (2.2) to determine a frame of reference for our problem.

2 **What is known in literature on capacity planning and scheduling in distribution centres?**
    - 2.1 How are capacity planning and scheduling problems categorized in literature?
    - 2.2 What approaches are known to solve capacity planning and scheduling problems?

As we gain insights in the current planning and scheduling methods (1) and shape a frame of reference in theory (2), we consider question 3 in Chapter 4. First of all, we determine how we can model our problem as an optimization problem, with clear objectives, variables, parameters and constraints. Then, we consider approaches to solving capacity planning and scheduling problems found in literature that can be adapted to our problem.

3 **What strategies can we formulate to tackle our problem?**
    - 3.1 How can we model our problem as an optimization problem?

We seek to design solution strategies to apply to our problem, to answer question (4). Then, we seek methods available that could be applied to our problem to create and optimize our problem and

compare them in their performance over time. Chapters 5 and 6 cover alternative solution methods we design to tackle our problem.

4 How can we design solution methods suited to the strategies applicable to our problem?

4.1 How can we design solution methods fitting solution approaches?

As we find multiple ways to tackle our problem, we consider the performance of each on a set of real problem instances (5.1) and consider the trade-offs in performance and feasibility on problem instances of these strategies (5.2).

5 How do the alternative strategies perform?

5.1 How is performance over time of the methods considered?

5.2 What are the trade-offs in performance and feasibility of the methods considered?

1.4. Research deliverables

The research deliverables we provide in this thesis:

- A capacity planning method that is both efficient and optimal
  - Comparison of performance of different planning methods
  - Insights in the performance and accuracy trade-offs of proposed methods
  - Insights in the time-performance trade-off of proposed methods
  - Comparison between limitations/assumptions made
  - Recommendations regarding the implementation of capacity planning methods

2. Context analysis

This chapter considers the following research questions:

1.1. How are capacity plans currently made for the distribution centre?
1.2. How is performance currently measured, and what is considered optimal performance?

Section 2.1 describes the system under consideration, or the part of the logistics process in the distribution centre, and the process flow through this system in Section 2.2. After that we describe the parameters and variables used to create capacity plans in Section 2.3.

To gain insight in how performance is measured (question 1.2), we first describe how plans are created in Section 2.3 and 2.4, and which (tacit) knowledge is used to evaluate them. We seek to determine the performance indicators used and the constraints that bound solutions optimality. Section 2.6 describes the assessment of plans is done as to finding the level of satisfactory or optimal performance.

2.1. System definition

Figure 3 shows the schematic production flow through the distribution centre at Albert Heijn Zaandam. Every week, there hundreds of orders for load carriers units from customer stores. These orders can distinguish combinations of units in three types. To produce these units, different processes are required. Figure 3 shows the three areas with the respective processes used to produce these demand units. The production areas are:

- **ACP: Automated Case Picking units** – The final units in this area are load-carriers with items that can be picked automatically (without manual labour). However, there are some workstations in these processes that require human resources.
- **NC: Non-Conveyable units** – The final units in this areas are load-carriers with items that cannot be picked using automated processes.
- **RRP**: **Retail Ready Packaging units** – This area constitutes Retailed Ready Packaging units that arrive from inbound trucks.

Inbound supply refers to the processes between inbound truck delivery and the buffers where units are stored until a demand request arrives. Outbound production refers to the processing of inbound units (cases on pallets or load carriers) to outbound load carriers that are ready to be shipped to customer stores.

Capacity planning in the distribution centres focuses on planning for outbound demand. The rationale for this is that (forecasts of) outbound store order demand are known well in advance, contrary to inbound deliveries for which supply is less well-known. Therefore we can only effectively plan for outbound demand in the tactical phase. Section 2.2 describes the outbound processes in more detail with an example of process flow through the ACP area.

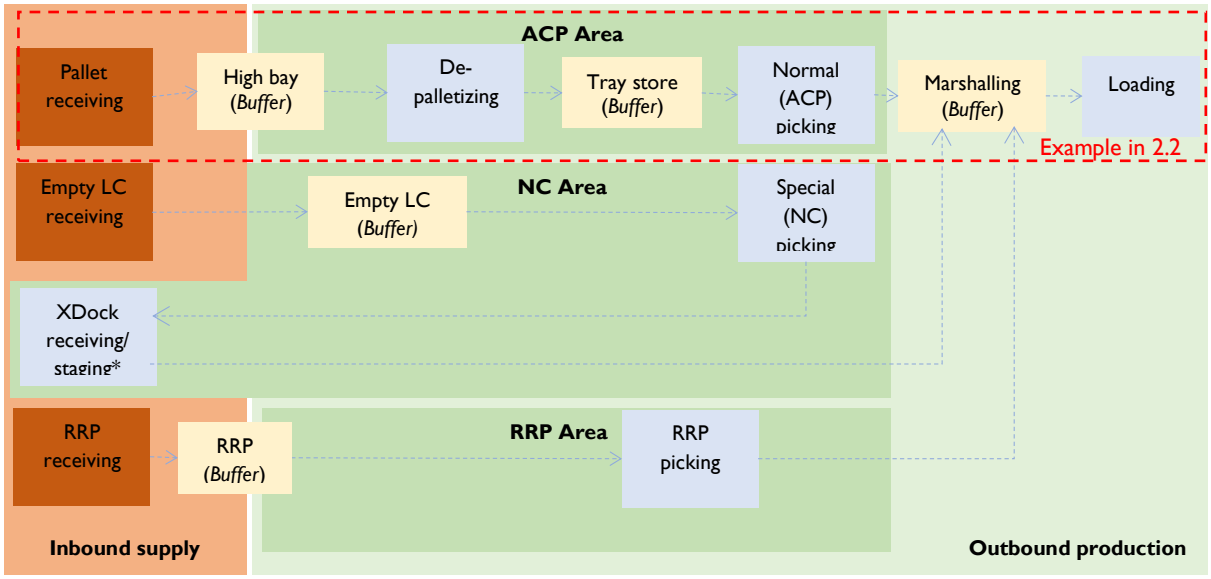


Figure 3 – Schematic production flow through the distribution centre

2.2. Process flow

In order to provide insight in how capacity plans are currently made for the distribution centre, we answer question *1.1: What does the process flow in the DC look like?* in this section.

This research considers outbound production in the distribution centre as the transformation from inbound units (cases, load carriers, or RRP displays used in the RRP area) in the buffer at the end of the inbound area to the combination of filled load carriers that are ready for outbound shipment to customer stores. Any process in the outbound area transforms these processing units somehow.

This section describes production flow through the processes from inbound decoupling point to outbound loading for the ACP production area (Figure 3) specifically. Figure 4 shows the inbound pallet receiving process cut-off point. This is the point from which outbound production and planning starts. Truck drivers manually transport full pallets in the pallet receiving area to a conveyor belt. From here they are transport to the High bay storage buffer, an ASRS system for pallets, shown in Figure 5. The high bay storage buffer serves as the decoupling point between the inbound and outbound processes in the ACP area as shown in Figure 3.

Outbound production is planned for customer store orders. As demand is expected, workstations will be made available to start processing. The retrieval of items from the high bay storage buffer marks the start of these outbound processes (Figure 5). This moment also marks the start of the internal lead-time for an order. The full pallets must eventually be transformed to load carriers used for filling shelves in stores. The first step to obtain these load carriers is to depalletize full pallets from the high bay storage into cases ready for handling.

Figure 4 – Inbound pallet receiving

Figure 5 – High bay storage buffer

Full pallets are conveyed to the depalletizing system resources. There are two types of workstations classes available to fulfil the depalletizing task. Figure 6 shows an automatic depalletizing system resource. This workstation does not require any human resource for operation. Figure 7 shows a semi-automatic depalletizing workstation where humans are guided in the depalletizing process by a system resource.

Figure 6 – Automatic depalletising

Figure 7 – Semi automatic depalletising

After depalletizing, cases are transported to the tray store buffer, an AS/RS system for trays, from which they can be moved the *Normal (ACP) picking* process, where they are stacked onto load carriers that are ready for stores. Again, two types of workstation classes are available to fulfil the picking orders. Figure 8 shows an automatic picking workstation not requiring any human operators. Figure 9 shows a semi-automatic picking workstation where a human is aided by a machine in the picking effort.

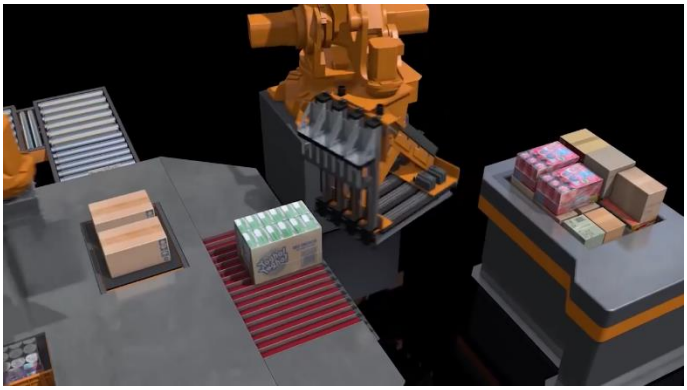


Figure 8 – Automatic picking

Figure 9 – Semi-automatic picking

While we describe process flow through the ACP area, each order may constitute demand for several production areas, these areas refer to the *RRP*, *NC (Non-conveyable)*, or *ACP (Automated case picking)* areas in Figure 3. For each of these areas, demand must be in the marshalling area in time so that the truck can depart at its predefined deadline departure time. The departure time marks the end of the internal order lead time.

Figure 3 shows an exception to the cut-off point between inbound and outbound processes, where there is no buffer between process *Special (NC) Picking* and process *XDock (Crossdock) Receiving*. In this chain of processes, load carriers (LCs) picked in the Non-conveyable (NC) process are transferred to the *Xdock* area where, they are consolidated and transported to the marshalling buffer. Therefore plans have to be made for *Special (NC) picking* in conjunction with *XDock Receiving*. The significance of this is that planners expect LCs from *XDock* to always arrive 8 hours prior to the departure of its respective truck, so that the working window for that part of the order is different than the outbound process.

2.3. Planning parameters and variables

Planning distribution centre capacity requires demand and supply information. Demand information for each week is found in the transport plan. The transport plan is generated from a combination of definitive and forecasted demand of store orders for separate production areas. We discuss the parameters for the transport plan in Section 2.3.1.

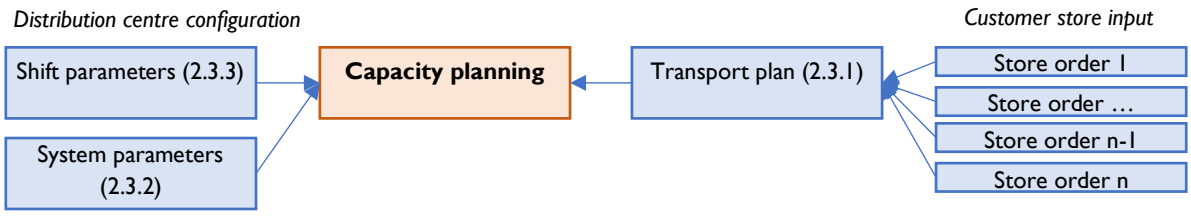


Figure 10 – Planning capacity using demand (customer stores) and supply (DC parameters)

On the distribution centre input, there are system and shift parameters. The system parameters (Section 2.3.2) cover the availability and production levels of process workstations as well as process precedence relations. Shift parameters (Section 2.3.3) are used to plan human resources over

workstations. The shift parameters describe time windows in which human resources can be allocated.

2.3.1. Transport plan parameters

On any production day, (hundreds of) store orders are to be fulfilled by the distribution centre. The plan containing a week's outbound store orders is referred to as the transport plan. This transport plan is input for the planning application as an XML file. Each store order has numerous attributes. We describe these orders by their attributes and function.

Attribute	Explanation
TruckDepartureDay	• Day at which store order is due
TruckDepartureTime	• Time at which truck departs on DepartureDay
Production order	• Production order, may consider several areas
ProductionAreaId	• Identifier of production area
Quantity of units	• Number of units required from production area
LoadingOffset	• Time required for loading
Cut-off offset	• Store order demand cutoff: demand is final only after (TruckDepartureTime – CutOffOffset), so outbound production cannot start before this offset.

Table 1- Store order atttributes,

Table 1 shows attributes of a single store order. The departure day and time together determine the moment in time at which outbound production order must depart to the store. This moment in time is marked as a deadline, it cannot be exceeded. All units for a store order must be in the Marshalling buffer (see Figure 3) at *TruckDepartureTime – LoadingOffset* on their respective day of departure.

Production for a store order is further constrained by the cut-off offset attribute. Stores may alter their orders until *TruckDepartureTime – CutOffOffset*. This implies that demand is not definitive until that time. Therefore, we consider the available working window for an order in the distribution centre as the interval from *TruckDepartureTime-CutOffOffset* to *TruckDepartureTime-LoadingOffset*. At the same time, the internal lead time may not exceed the cutoff offset.

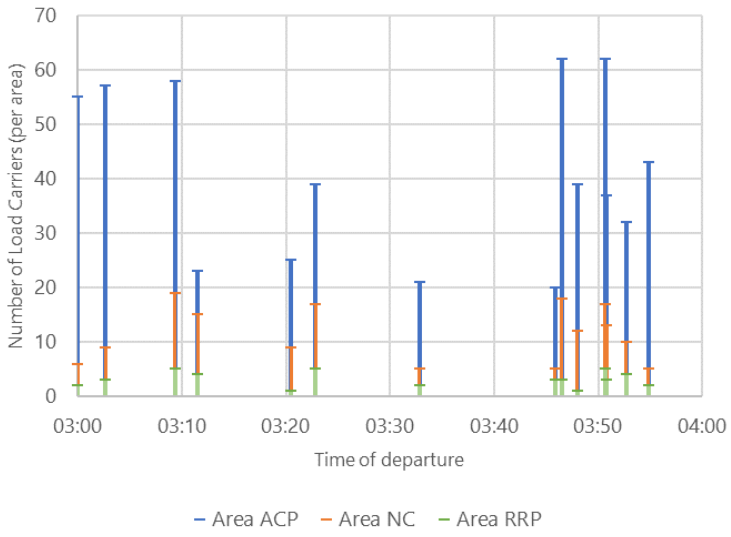


Figure 11 – Sample transport plan time window

Figure 11 shows a one hour time window of a transport plan. Here, there are several production area orders, where each order contains load carrier demand for three distinct production areas that must be ready at the time of departure specified in the interval. This means that at the times specified, all demand load carrier demand for each area in that order must be finished. This time is offset by the loading offset. Furthermore, scheduled production for these orders cannot start before the cut-off offset as stores can alter their orders until that time.

2.3.2. System parameters

Each production area is composed of production processes. Orders for these areas must follow the areas constituent process in a fixed order. Table 2 shows the attributes of a production process. By linking the *ProductionAreaId* from the store orders in the transport plan in Table 1, we retrieve the corresponding processes required in an area from Table 2. Each process with *ProductionAreaRef* equal to the *ProductionAreaId* in the store order attributes must be passed for a store order with demand for that area. The relations between processes are given by the incoming and outgoing buffers.

Attribute	Explanation
Name	• Name of process
ProductionAreaRef	• Reference area
InBuffer(s)	• Incoming buffer for process
OutBuffer(s)	• Outgoing buffer for process
Offset before	• Time before process (used in scheduling)
Offset after	• Time after process (used in scheduling)

Table 2 – Process attributes

In order to fulfil the demand specified in a store order's *QuantityOfUnitsToProduce*, there must be sufficient capacity at processes in their working window. There are numerous system workstations available at each process. The main differentiating factor for workstations in a process is their handling type. Table 3 shows these workstation classes. E.g. in the process *Depalletising* there is a maximum of 10 workstations with *WorkstationType* = *Depalletising*, of which 6 have *HandlingType = Automatic* and 4 have *HandlingType* = *SemiAutomatic*, acting as two different workstation classes for one area. Among these classes, workstations can be considered uniform as their productivity and cost are considered equal. The *HandlingType* attribute determines if one or more operators are required to fulfil production. Automatic system workstations do not require any operators apart from supervisors, whereas both semi-automatic and manual system workstations do require operators.

Attribute	Explanation
Name	• Name of workstation class
Process ref	• Reference process
WorkstationType	• Type of workstation
HandlingType	• How cases are handled: automatic or manual/semiautomatic
Capacity	• Production capacity per hour of corresponding workstations.
NumberOfOperators	• Number of operators required at workstation in the workstation class

Table 3 – WorkstationClass attributes

Table 3 introduces handling types for workstations. There is only a limited number of workstations at each process that have handling type *Automatic*, additional workstations require handling type being either *Manual* or *SemiAutomatic*. Workstations with handling type ≠ *Automatic* require human

resources for operation. These human resources can be allocated for timeslots of 10 minutes, but their employment is constrained in the duration of their shift working window. For instance, an operator may be available in the shift from 07:00-15:00 on Monday. In that case, cost is incurred for all 48 ten-minute timeslots in that time window.

2.3.3. Shift parameters

Many system resources require human resources. These human resources can only be employed in the length of shifts. The properties of these shifts are defined in the shift parameters file. The shifts are defined in windows of starting and ending times. For the distribution centre in our scope, we consider three adjacent full-time shifts and a shorter part-time evening shift. Human resources incur cost for the entire duration of the shift. They cannot be employed for shorter time intervals than any of these windows.

Shift	Start time (h:mm)	End time (h:mm)	Total time (h:mm)
Day shift	7:00	15:00	8:00
Evening shift	15:00	23:00	8:00
Night shift	23:00	7:00	8:00
Flex evening shift	15:00	19:00	4:00

Table 4 – Shift settings

For the distribution centre under consideration, the shifts available are categorised following Table 4. There are three 8-hour shifts spanning full days. Next to that, there is an overlapping flex evening shift. Each of these shifts has a predefined number of breaks. During these breaks, they cannot work at any workstation.

2.4. Current planning method

This section provides insight in the current planning process to answer question *1.4: "What does the planning process currently look like?"*.

From the information found in store orders, planners have to create workstation allocations so that sufficient capacity is available to meet demand requirements at minimum use of resources. Planners can define the availability of a workstation in timeslots of 10 minutes. If they define a workstation as available in a timeslot, this workstation is considered by the control application in the operational phase so that units (cases, pallets or load carriers) may flow to that workstation for processing. Figure 12 shows an example of an allocation of workstations at the NC picking process.

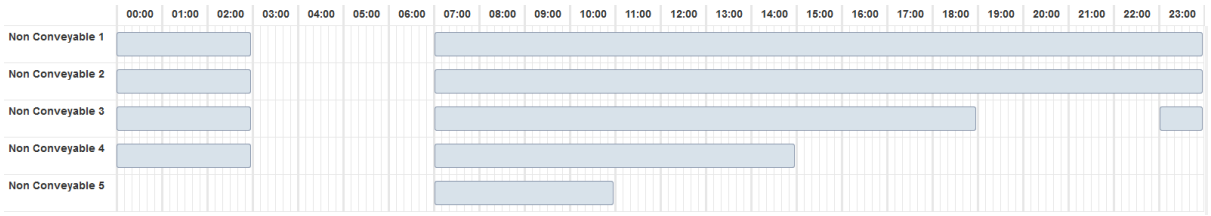


Figure 12 – Illustration of a sample allocation of workstations at a process.

In practice, planners start planning for a weeks forecasted demand with a default allocation of system resources. This default allocation of resources already allocates many workstations requiring human

resources throughout the week. The planner decides to either modify allocations using the feedback found from the demand scheduler (we describe the scheduler in more detail in Section 2.6), or deem the current allocation satisfactory. They decide to add or remove (multiple) workstations (and possibly human resources) for a range of timeslots and schedule again. They continue this iterative process until they find no more errors and performance is considered satisfactory.

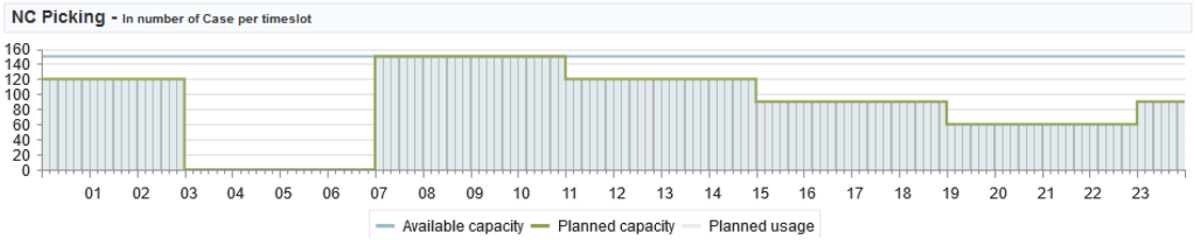


Figure 13 – Workstations translated to capacity

Figure 13 shows how the availability of workstations is translated to capacity. Note that in this case, demand has been scheduled for all available capacity, the workstations are fully utilised as all workstations are in use during all timeslots in the figure. Figure 14 shows a capacity allocation over the duration of a full week. Here, we note that capacity varies significantly and that not all capacity is used all the time.

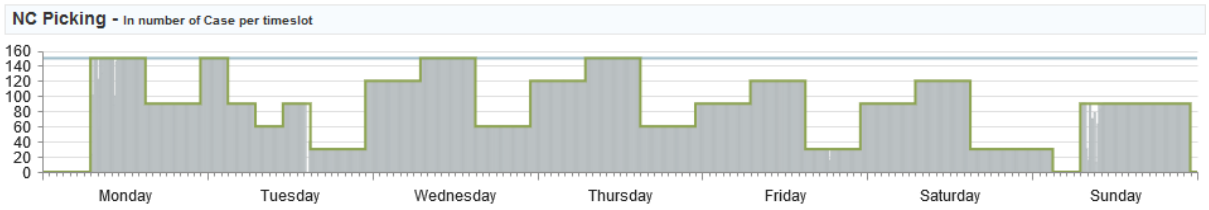


Figure 14 – Capacity allocation over week

2.4.1. Planning window overlap

Planners create transport plans in the tactical phase that considers store orders for a single week from Monday to Sunday. The planning process does not take into account either capacity or demand from preceding or succeeding weeks. This decision leads to problems as the distribution centre does normally expect demand in adjacent weeks whereas this is not planned for. To resolve this problem, the scheduling algorithm uses the capacity and demand of the current week to simulate the capacity and demand of the adjacent weeks.

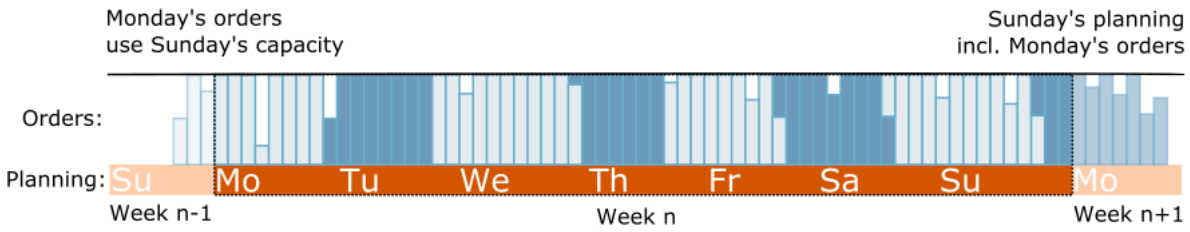


Figure 15 – Extending the planning window using current week

The extension of the planning window aims to solve two problems that arise if demand from adjacent weeks is not considered:

1. *Undercapacity at the end of the planning week following no expected demand in the succeeding week* – when the demand from a succeeding week is not taken into account, then capacity at the end of the planning week will typically be allocated to a very low level. For instance, there may be an order that must depart at 01:00 on the Monday succeeding the current planning week. The majority of processing for this order must be done on the planning weeks Sunday. However, this is not taken into account if the planning week is not extended. To resolve this, the demand and capacity of the planning week's Monday is appended to the end of the current planning window. As weeks demand does typically not vary greatly, an estimate of true demand is obtained rather than creating undercapacity.
2. *Overdemand at the beginning of the planning week not being accounted for leading to practically infeasible allocations* – Overdemand may exist at the beginning of the planning week

Figure 15 shows how overlap in planning weeks is 'simulated' as an extension of the current week. Demand from the succeeding Monday (from week n) is appended to Sunday (in week n, the week considered for planning) to simulate demand from week $n+1$. If this would not be done, then the capacity at the end of the Sunday in week n would be too low as no demand is expected in the succeeding week.

Next to that, Sunday's capacity is added to precede the current planning week. The capacity on Sunday in week $n-1$ is equal to the capacity on Sunday in week n. This is done so that the added lead times can be calculated more reliably. Section 2.5 covers this aspect in more detail.

2.4.2. Dealing with infeasible plans

If we allocate all system resource and specify processes for each human resource during their availability, there should be a schedule that satisfies all constraints (Table 5 introduces these specific constraints), otherwise there is no feasible solution for that particular plan. Hans (2001) describes 4 options available to a production planner if the available capacity in a plan is insufficient.

1. Shift jobs in time. Or to split jobs over two or more periods
2. To increase the lead time of some customer orders (by decreasing their start time, or by increasing their due date), and then to reschedule;
3. To expand operator capacity in some weeks by hiring staff
4. To subcontract jobs or entire orders

Outside the scope of our research, other options for creating feasible plans are available. If no feasible solution can be created by assigning maximum available capacity in each timeslot in a planning window, production planners may decide to alter the input transport plan. For instance, they can communicate with stores asking them to change their cut-off times and thus spreading demand to mitigate infeasible schedules. They can then provide the altered transport plan to the TOP planning module to be used as input for capacity planning.

In the scope of our research, we only consider (3) as an option for an automated capacity planner. The other three options are available outside the scope of this research (1.3.1) as they require communication with other outside actor, such as retail stores or inbound suppliers that is hard to do automatically.

2.5. Scheduling store orders

The demand scheduling function automatically schedules forecasted and/or actual store orders within resource availability allocations introduced in the tactical or operational plan. The main purpose of the demand scheduler is to first validate whether a proposed tactical or operational plan provides sufficient capacity to meet demand for the warehouse, and second aid in optimising the production plan. Section 2.6 describes these performance indicators and constraints.

During the Tactical and the Operational capacity planning phases, the demand scheduling function can be used to evaluate a resource plan on user request. This effectively enables the planner to perform a so called "what-if" analysis to evaluate alternative planning scenarios.

The following main objective applies to the demand scheduling function: a store order needs to be scheduled as late as possible, i.e. following a just in time (JIT) method. This objective is introduced for a number of reasons:

- To minimize the order lead time;
- To minimize the amount of shortages due to late inbound deliveries;
- To minimize the occupation level in the outbound marshalling buffer;
- To maximize the time for the transport planning to reassign load carriers over shipments without impact on the logistics flow within the warehouse.

Orders are scheduled based on the capacity plan created by the planners workstation allocations. The scheduling of these orders is a simulation of production as the true production parameters of store orders are not regarded as known in the planning phase, but are only regarded in the control phase (see Figure 1) which controls product-by-product flow and production times. These true production parameters may include the exact production composition for orders areas (e.g. volume and weight of cases). In Section 1.2 we explained how the planning module creates a simulated environment of the DC. On the day of production, the production may have slightly shifted or orders may be composed differently: the adaptations to the scheduling of orders is then done in the control phase. In other words, the output from the planning module, the scope of this thesis, is a resource allocation per discrete timeslot

The scheduling, or simulation of order processing in the DC, is done using a scheduling algorithm in the planning module. After the outbound transport plan (result of forecasted orders and store orders) and the available capacity plan (described in Section 2.4) have been determined, the algorithm schedules store orders from the transport plan on the workstations available.

We briefly describe how the scheduling algorithm works on a just-in-time basis using illustrative examples. The algorithm starts by identifying timeslots with insufficient production capacity (overdemand) from the earliest available timeslot onwards. When a timeslot with overdemand is found, the algorithm attempts to reassign the production demand to the first earlier timeslot with available production capacity. The scheduler is considered finished only when there is no overdemand left in any timeslot at any process.

1. Allocate all store order demand to the timeslots in which they are due at the final process in each area corresponding to the order.

2. Recursively shift excess demand from the last demand timeslots due so that no excess demand exists.

Scheduling example – toy problem

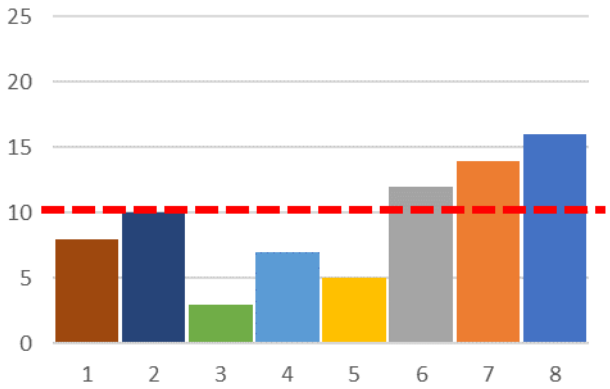


Figure 16 – Scheduling example, initial demand and constant capacity = 10

We consider a toy scheduling problem for a sample process. Figure 16 shows initial demand for a planning window of 8 timeslots in length. The figure illustrates the final demand for orders at this process that have a deadline in each timeslot of the timeslots, separated by color. The vertical axis shows the number of units due at the process we consider at each timeslot. There is a red line representing the production capacity during each timeslot. This example considers uniform production capacity of 10 units per timeslots. In reality, the production capacity frequently differs.

The distribution centre cannot process more than 10 cases per timeslot, otherwise more workstations must be allocated to add capacity. We note that there is excess demand. Demand that is due in timeslots $t \in \{6,7,8\}$ cannot be processed immediately. Therefore, they must be processed in another timeslot. The orders cannot be shifted later in time as they have a departure deadline. They are shifted recursively following the scheduling algorithm.

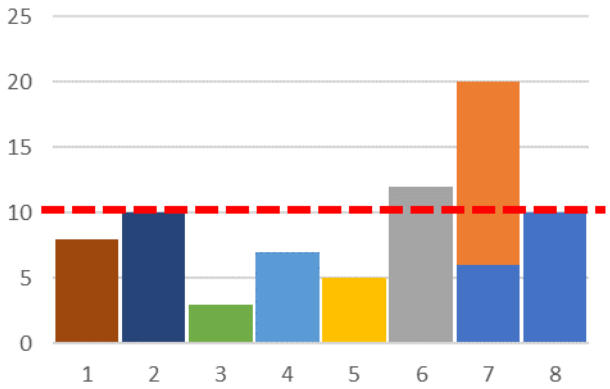


Figure 17 – First shift in schedule

Figure 17 shows the first shift in the schedule in an attempt to mitigate overdemand. Demand in timeslot 8 is shifted forward. Overdemand remains in timeslots 6 and 7.

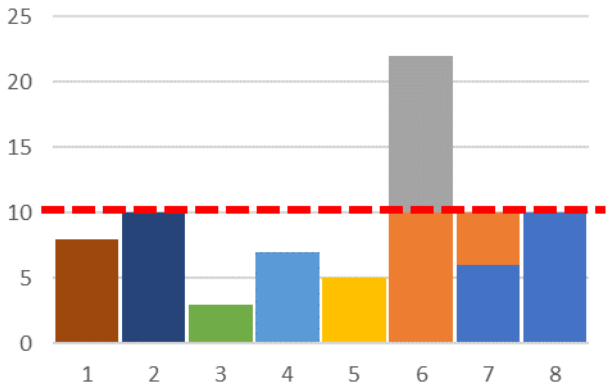


Figure 18 – Second shift in demand, mitigating overdemand in timeslot 7

Figure 18 shows how overdemand in timeslot 7 is mitigated. The scheduling algorithm shifts the demand from the order set due in timeslot 7 forward.

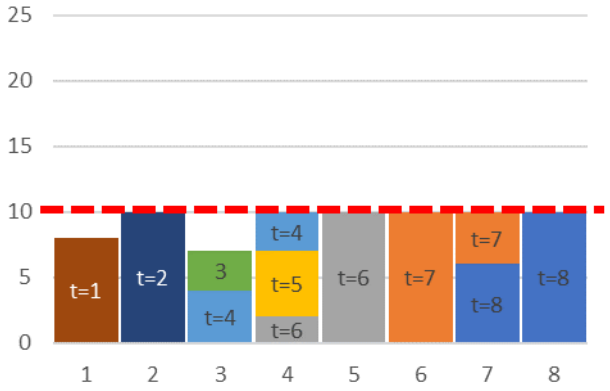


Figure 19 – Final scheduling attempt

For this allocation, the scheduling algorithm continues to seek and remove peaks from timeslot 6 following the priority rule. Figure 19 shows the final schedule once all demand has shifted and no overcapacity remains.

Note that each recursive shift adds lead time of one timeslots for the orders. Figure 19 shows that for instance demand with a deadline t=6 is now processed even in timeslot 4. Remember that this scheduling function is an example of a single process, where scheduling is done recursively from the deadline time following a JIT rule. Note that there may be processes before this process. If we take the analogy of the ACP area, the process in the picture may be *ACP Picking*. The preceding *Depalletizing* process that must be finished before *ACP Picking* can start.

Therefore, the demand at t=6 can only be added as demand at *Depalletizing* at t=4 or earlier. This way, an offset of at least (t=6) – (t=4) = 2 time slots is added. Next to the offsets added by under capacity in timeslots, there are fixed offsets between processes.

2.5.1. Precedence in scheduling production flow

The scheduling algorithm example describes how demand is scheduled for an allocation over a single process. In reality, demand for a single area considers one or more processes that must be processed

sequentially. Figure 20 shows the precedence relations in the process flow in the production and scheduling of orders. Each store order consists of demand for several areas, that may consist of several processes. The store orders are ranked by their departure time, which serves as indication of the order of processing.

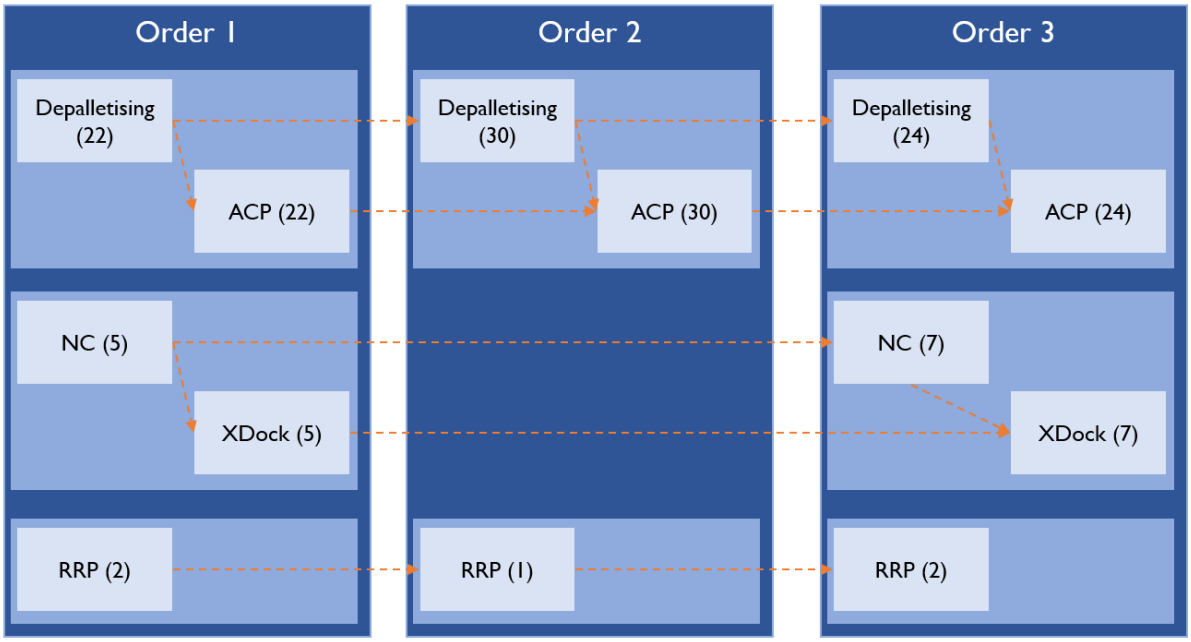


Figure 20 – Precedence relations in production

In this case, store order 1 constitutes demand for 3 areas, whereas order 2 contains no demand for the second area. There are strict precedence relations between the processes in these areas, indicated by orange dashed arrows. These arrows indicate that if and only if all predecessors to a process have finished processing, their common successor may start production.

Production at the *ACP* process for store order 2 cannot start before both depalletizing for order 2 and ACP for order 1 have finished processing. The time between starting at successors is further offset by the attribute's offsets preceding- and succeeding the processes.

2.5.2. Offsets in scheduling

The scheduling algorithm example in Figure 19 shows that that offsets in time are added for some orders that had overdemand in their respective timeslots as demand shifts back in time. These offsets contribute to the total lead time for an order. Each constituent process of an area contributes to the total lead time of a production area order in an order. The maximum of these production area order leadtimes is considered the lead time for an order. Next to these offsets that arise as a function of the capacity allocation, there are fixed offsets. These can be offsets before and after both processes and areas as well as time required for loading, as seen in Figure 21.

Any order on any resource must be finished fully before work on a succeeding order with later departure times can start. Furthermore, the processing sequence for any area is fixed in the departure time of the orders. Therefore, we can quickly schedule recurrently from the last timeslot in the planning window if we consider fixed capacity. The scheduling process is deterministic. That is to say

that it always creates the same schedule for one capacity plan. The scheduling problem is therefore not an optimization problem and can be considered trivial.

From this description of the scheduling algorithm, we find several interesting properties that may be of use in a categorization in the literature review. The first is that we do not allocate orders to workstations. We schedule the capacity required (by customer orders) for an order in a timeslot over the total capacity available (determined by planners) at a process in that area and timeslot iteratively.

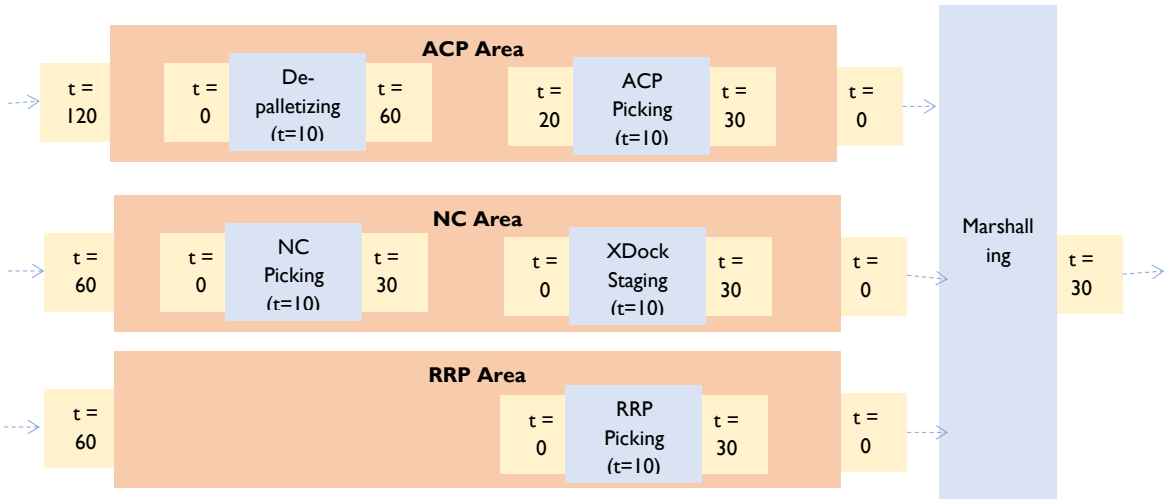


Figure 21 – Fixed offsets between processes and areas in the DC

The fraction of customer orders processed during a timeslot for a order is variable up to the finest level of granularity (1 case). This implies that the processing time is also variable, as order processing time is spread over preceding timeslots if their deadline timeslot does not suffice. The time required for processing cases corresponding to a customer order depends as a linear function on the amount of capacity allocated to it. Moreover, we observe hard precedence relations. Work for an order must always be completed entirely before processing on order with greater departure time can start. If two orders have a departure time in the same timeslot, we can group them as Chapter 4 will describe. Finally, we observe that pre-emption is possible. If an order is due in a timeslot and its preceding timeslot has no free capacity, production can be shifted to an earlier timeslot.

The offsets added to an order through scheduling contribute to the order lead time. Two factors determine order lead time: fixed and variable offsets. The variable offsets are introduced by the scheduling algorithm. Figure 21 illustrates the fixed offsets. Therefore, there is a minimum time required to process every order. For each order, the lead time is constrained. It cannot increase beyond a certain value set by the store. Section 2.6 treats this constraint in more detail.

2.6. Capacity plan performance evaluation

Section 2.5 mentions that the function of order scheduling is to get insights in performance. In this case performance refers to hard- and soft constraints as well as KPIs. Table 5 shows performance constraints and indicators. Plans are considered infeasible when hard constraints (i.e. errors) persist. Hard constraints indicate that a capacity plan must be altered. Soft constraints are considered as

warnings, they give an indication of improvement potential in a plan, these warnings indicate that a capacity plan could likely be improved. Section 2.6.1 introduces these constraints in more detail. Section 2.6.2 describes the performance indicators.

2.6.1. Planning constraints

We describe each of these constraints. Figure 3 shows buffers between processes. These buffers are limited in size. The marshalling buffer can only store a limited number of load carriers similar to how the tray store can only handle a limited number of trays. When buffers are full, production at preceding processes must be stalled, adding lead time and idling workstations. The scheduling algorithm described in Section 2.5 does not take these buffers into account, as it continues scheduling recursively. A hard constraint (1) is that the number of units in a buffer cannot exceed its maximum, so that the scheduling algorithm mirrors expected production parameters.

Some cases cannot be processed by fully automatic workstations. However, the exact information on which cases are unsuitable for automated depalletizing is not known in the planning phase (only available in the control phase). Hard constraint (2) adds that the ratio of workstation capacity with handling type *Automatic* to *SemiAutomatic* in the depalletizing process cannot exceed 4. The reason for this is that on average, the fraction of cases requiring handling by *SemiAutomatic* workstations weighed with the production capacity of those workstations is expected to exceed 0.25. The expectation is that true performance becomes lacklustre if this ratio is not met: as production for cases that cannot be handled automatically would stall, creating queues and increasing lead times.

Hard constraints (errors)	**Soft constraints** (warnings)	**KPIs**
1. Maximum buffer capacity cannot be exceeded	1. Efficiency too low	1. Buffer utilisation
2. Handling type capacity ratio in depalletizing must be at least 4		2. Order Leadtime
3. Cutoff offsets cannot be exceeded (per order)		3. Production throughput
4. Fixed maximum leadtime cannot be exceeded (for all orders)		4. Utilisation
5. Overdemand at process		**5. Productivity (cases/operator hour)**

Table 5 – Capacity plan performance constraints and indicators

The third hard constraint (3) indicates that cut-off offsets cannot be exceeded, meaning that the maximum lead time for an order cannot be greater than *CutOffOffset*. In other words, all production for an order must occur in the window [*TruckDepartureTime – CutOffOffset, TruckDepartureTime*]. If this window is not adhered to, then stores cannot alter their demand anymore as production has already started. Figure 22 shows an illustration of the realized lead times that are created by shifting demand in the scheduling function.

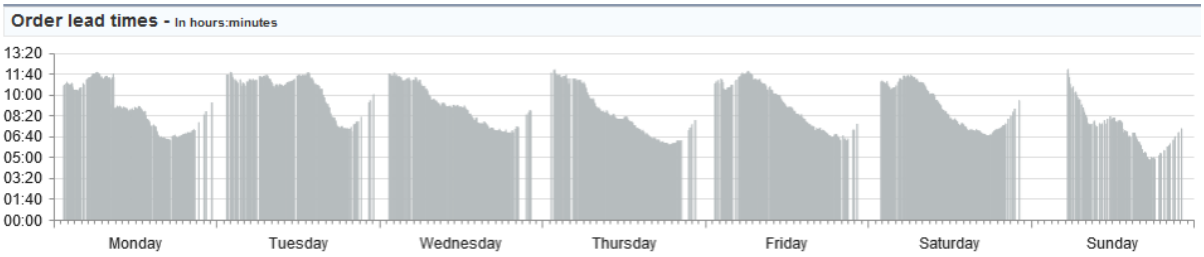


Figure 22 – Illustration of lead times per order plotted with the departure time of orders on the horizontal axis

Hard constraint (4) states that maximum lead time of an order cannot exceed a fixed time that is equal for all orders. This is an addition to the cut-off constraint. The *CutOffOffset* for any order may exceeded the maximum lead time over all orders. To satisfy this constraint we should constrain the allowed to work window of an order further, to: [*TruckDepartureTime* – min(*CutOffOffset*, *MaximumLeadTime*), *TruckDepartureTime*].

The final hard constraint (5) mitigates overdemand at a process. The term overdemand was introduced in Section 2.5.1. If demand cannot be shifted to the present due to undercapacity at workstations, then some demand cannot be fulfilled thus creating overdemand. This should not be possible as we have an objective of fulfilling all store order demand.

There is a soft constraint. This soft constraint serves as a warning rather than error (hard constraints). It aids aid planners in creating performant capacity plans. This soft constraint is that planners receive an indication when efficiency at a process is too low in a plan.

2.6.2. Performance indicators

Once hard constraints are satisfied and soft constraints are mitigated, performance can be optimised. The planning module shows the manual planner KPIs of a capacity plan in Table 5. In Section 1.2 we mention that the goal of planning for the distribution centre is to fulfil store orders with minimum cost. We describe these indicators and provide arguments for why we decide to optimise for productivity only, as optimising for productivity adds to the planning goal by itself while constraints are adhered to.

- *Buffer utilisation*: dominated by hard constraint: buffer capacity cannot be exceeded. A lower buffer utilisation is generally preferred, however it does not contribute directly to a lowered objective of obtaining maximum throughput for minimum cost. The buffer utilisation merely constrains production when buffers are full (or not full enough)
- *Order leadtime*: dominated by hard constraint cutoff times cannot be exceeded, maximum leadtime cannot be exceeded. Order lead times do not directly contribute to the objective, they offer an indication of production slack. If order lead times are low, we can expect to be able to reduce capacity for some timeslots.
- *Production throughput*: Dominated by cutoff times cannot be exceeded, implying that all store orders must be fulfilled and all orders are processed.
- *Utilisation*: indicates the capacity of workstations used as percentage of the maximum available of workstations, if utilisation > 1 a transport plan would be infeasible, requiring changes to the allocation. However, as there are limits to the number of workstations

allocated the changes must stem from decisions outside the scope of the planning module (e.g. arranging different departure times with stores)

2.7. Cost function

Given that a solution is feasible, (satisfying all constraints) its performance is evaluated using a cost function. The cost function measures the weighed cost of employing system- and human resources during shifts. There is cost differentiation over timeslots for human resources. Additional hourly compensation is incurred following the percentage multipliers in Figure 23. All timeslots in these hours incur an additional percentage cost equal to the amounts listed. We observe that there is a preference to work during the day.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Monday	55	55	55	55	55	55	55	0	0	0	0	0	0	0	0	0	0	0	37	37	37	37	37	39
Tuesday	39	39	39	39	39	39	39	0	0	0	0	0	0	0	0	0	0	0	37	37	37	37	37	39
Wednesday	39	39	39	39	39	39	39	0	0	0	0	0	0	0	0	0	0	0	37	37	37	37	37	39
Thursday	39	39	39	39	39	39	39	0	0	0	0	0	0	0	0	0	0	0	37	37	37	37	37	39
Friday	39	39	39	39	39	39	39	0	0	0	0	0	0	0	0	0	0	0	55	55	55	55	55	60
Saturday	60	60	60	60	60	60	60	35	35	35	35	35	35	35	35	35	35	35	70	70	70	70	70	75
Sunday	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	55	55	55	55

Figure 23 – Hourly cost matrix during week

Capacity plans yield an indication of the required human resources employment on system resources over the length of a shift. The capacity plans do not indicate which exact human resource is required on a specific system resource. In fact, there is a preference among warehouse employees to work at workstations in different processes during their shift. Therefore, we do not have to allocate system resources requiring human resources for intervals of human resource shift length. However, from an optimization perspective it is implied that the sum of human resources over timeslots at all workstations during a shift is equal to the number of human resources working during that shift. Shift utilization may therefore be regarded as an implicit performance measure. A plan can be regarded increasingly inefficient as function of decreasing shift utilization as human resources idling increases and productivity decreases. Normally, maximizing the single performance indicator productivity implies a maximum utilization of resources.

Another implicit constraint considers the workstation allocations early on Sunday. There is a strong preference for not working from 00:00 to 07:00 on Sunday. We must take this into account when modelling the problem: allocations in that timeframe should only be considered if plans are otherwise infeasible.

2.8. Conclusion

This chapter provides insight in the method of capacity planning for the distribution centres. The current manual planning process involves an iterative process. For each process in the distribution centre, the availability of resources is indicated for 10-minute timeslots per week. Planners allocate

resource to cope with store order demand. The allocation of these resources must be feasible. That means that the capacity plans should not violate constraints.

The evaluation of the capacity plans is done using a scheduling function that is based on a Just-In-Time priority rule. The store order demand can be scheduled using the function. Then, the planning application provides feedback to the planner. It indicates if and which constraints are violated. Next to that, performance measures are included to the planner. Using these indicators, the planner decides whether the current capacity plan is satisfactory.

The current planning process follows a 'what-if' loop iteratively. In practice, planners find themselves creating many iterations until a satisfactory capacity plan is obtained. A satisfactory plan does not violate any constraints and performs well in the pre-defined performance indicators.

3. Theoretical framework

This chapter considers research question 2: *What is known in literature on capacity planning and scheduling in distribution centres?* To answer this question, we first consider question 2.1: *How are capacity planning and scheduling problems categorized in literature?* We consider the hierarchical planning framework in section 3.1, and find strategies available to tackle problems in the tactical/operational phase. Section 3.2 reviews planning and scheduling problem formulations in literature.

Then, we consider research question 2.2: *What approaches are known to solve capacity planning and scheduling problems?* Section 3.3 discusses solution approaches to the problem. We describe exact methods in Section 3.4 and approximate methods in Section 3.5.

3.1. Hierarchical planning framework

In this section we review the hierarchical planning framework for hierarchical planning in (semi-) project-driven organizations of De Boer (1998). Figure 24 shows this hierarchical planning framework. It distinguishes four levels of (horizontal) planning activities in order to break down multi-project planning into more manageable parts: strategic resource planning; rough-cut capacity planning; resource-constrained project scheduling and detailed scheduling. Strategic resource planning considers long-term decisions regarding space, staffing levels, layouts and the number of resources.

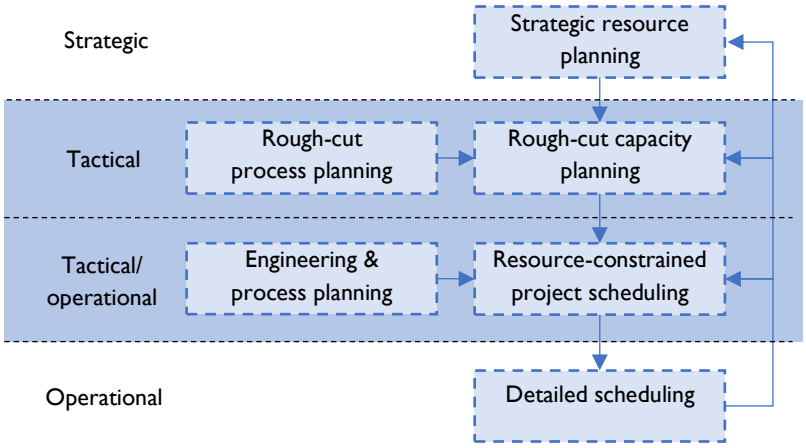


Figure 24 – Hierarchical planning framework in (semi-) project-driven organisations (De Boer, 1998)

Rough-cut capacity planning in the tactical phase considers a medium-term horizon, where the question is how sufficient resources can be allocated to cope with demand (projects) as effectively as possible (Hans, 2001). In the tactical level, the problem to be solved is the allocation of resources such as machines and workforce. In this phase, the regular capacity is considered fixed due to e.g. workstation or machine availability, but medium term decisions can be made on hiring additional personnel (Leus, 2003). The tactical planning phase window is most often regarded in the order of weeks to months. In order to generate input for the RCCP, some rough-cut process planning has to take place. Based on customer specifications, this rough-cut process planning process should result in a network of work packages with rough estimates of resource requirements.

Capacity planning and scheduling problems are often distinguished in being time-driven or resource driven (Möhring, 1984). This refers to the objective and constraints of the optimisation problem. If a problem is resource driven, its objective is to minimize tardiness or makespan, constrained by fixed capacity levels. In a time-driven problem, the optimisation objective is to minimize additional resource usage, while being constrained by due dates of components. There are hybrids of the two, where the objective function minimizes a tradeoff between makespan and resource usage.

The resource-constrained project scheduling (RCPS) phase determines when activities are performed as capacity levels are more or less given by the rough cut capacity plan. A more detailed activity network can be drawn where work packages of the RCCP level are broken down into smaller activities with constant duration and resource rates. However, RCPS does not indicate which persons or machines of a group are assigned to each activity.

Detailed scheduling specifies which machines or persons should work at what moments in time. In Chapter 4 we argue that our problem considers the tactical planning as well as the tactical/operational phase. Therefore, we require interaction between planning and scheduling phases.

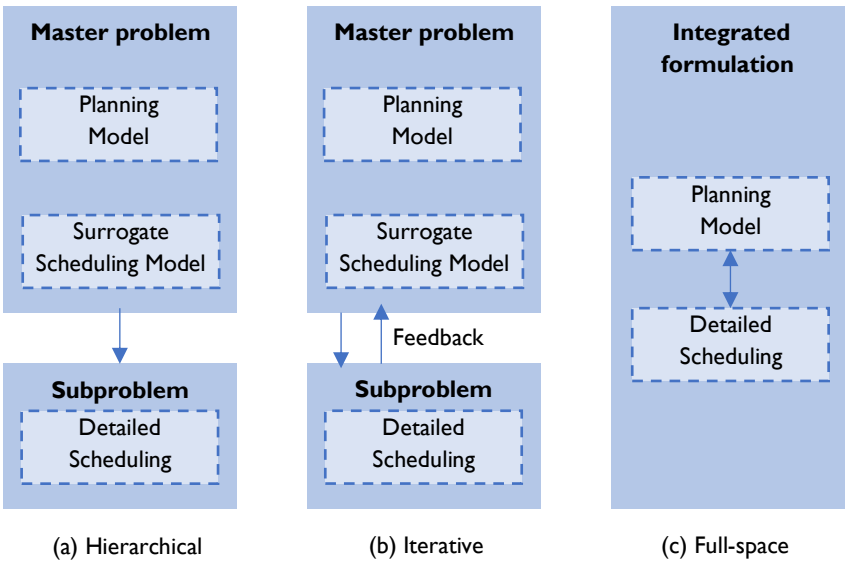


Figure 25 – Solution strategies for integrated production planning and scheduling (Maravelias & Sung, 2009)

Figure 25 shows a classification of solution strategies for the integrated planning- scheduling problem in three categories (Maravelias & Sung, 2009). If the flow of information is only top-down from the planning model to detailed scheduling without intermediate planning, then hierarchical strategies can be considered. If there is a feedback loop from the scheduling models back to the master or tactical planning problem, then the methods can be considered iterative. If the problem considers a full formulation with detailed scheduling information for each planning period, then its solution should provide all the necessary information. These strategies are referred to as full-space or integrated methods.

In Chapter 4 we argue that we can approach our problem using the either the iterative sequential (b) or the full-space integrated (c) strategy as iterative or concurrent communication is required between planning and scheduling functions. In the coming sections we review solution methods based on both these strategies. Section 3.2 covers planning and scheduling problems than span the tactical and operational phase. Section 3.3 introduces the approaches that can be used to tackle these problems.

Section 3.4 covers exact methods in more detail. Section 3.5 covers approximate methods in more detail.

3.2. Planning and scheduling problems spanning the tactical and operational phase

This section examines capacity planning and scheduling approaches to combinatorial optimisation problems found in literature.

3.2.1. Capacity planning problems

Hans (2001) state that resource loading concerns loading a given set of orders and determining the resource capacity levels that are needed to process these orders and their constituting jobs. Resource loading considers a set of jobs that must be scheduled within a given time horizon that is organized into consecutive time periods, where each period is associated with a fixed number of available workers. The objective is to find a feasible schedule that minimizes the number of additional workers needed to execute all the jobs. The resource loading problem (RLP) solely allows for the modelling of linear precedence relations. The RLP with generalized precedence constraints is known as the Rough-Cut Capacity Planning (RCCP) problem (Hans, 2001).

In Chapter 2 we find that the main objective of our planning problem is to minimize the cost of using non-regular capacity. A planning problem exhibiting similar objectives is the time-driven variant of the RCCP-problem in the tactical phase (De Boer, 1998).

Hans (2001) formulates a mixed time-driven and resource-driven RCCP problem for a job shop in which aggregate orders containing jobs with precedence constraints are loaded onto resources. For this, they consider fixed machine capacity and operators capacity. Operators are distinguished between regular and non-regular capacities. They consider no overlap between connected activities.

Gademann & Schutten (2001) study the time-driven RCCP problem in a multi-project environment with precedence relations. In their description, the problem contains work packages to be planned on resources from several projects. They have to be planned so that due- and release dates are met. They also include a minimum working fraction for any work package in a timeslot. De Boer (1998) extends the time-driven RCCP model with suggestions for cost differentiation and capacity constraints.

3.2.2. Project scheduling problems

Kis (2005) study a resource constrained project scheduling problem in which the resource usage of each activity may vary over time proportionally to its varying intensity where overlap of preceding activities is allowed. These generalized precedence constraints allow for modelling minimum and maximum time-lags between activities (Neumann, Schwindt, & Zimmermann, 2002). However, the number of resources in Kis (2005) is considered fixed. Kis (2005) consider the time-driven resource constrained project scheduling problem (RCPSP) with nonregular hiring and variable intensities, with feeding precedence relations, here there can be overlap between activities connected by precedence relations.

3.3. Solution approaches

In Chapter 4 we describe that the integrated problem can be viewed as a combinatorial optimisation problem "Combinatorial optimisation is the mathematical study of finding an optimal arrangement, grouping, ordering or selection of discrete objects usually finite in numbers." (Lawler, 1976).

A combinatorial optimisation problem is referred to as easy if we can develop an algorithm that solves every problem instance in a polynomial-time complexity bounded by the size of a problem instance to optimality (Osman & Kelly, 1996). Such an algorithm is referred to as being efficient. A problem is called hard if efficient algorithms for solving it do not exist.

Figure 26 shows two main method categories: exact methods and approximate methods specifically for the vehicle routing problem. Many of these methods can be applied to other combinatorial problems. Exact methods obtain optimal solutions and guarantee their optimality (Talbi, 2009). Approximate methods find good solutions on large-size problem instances (Talbi, 2009).

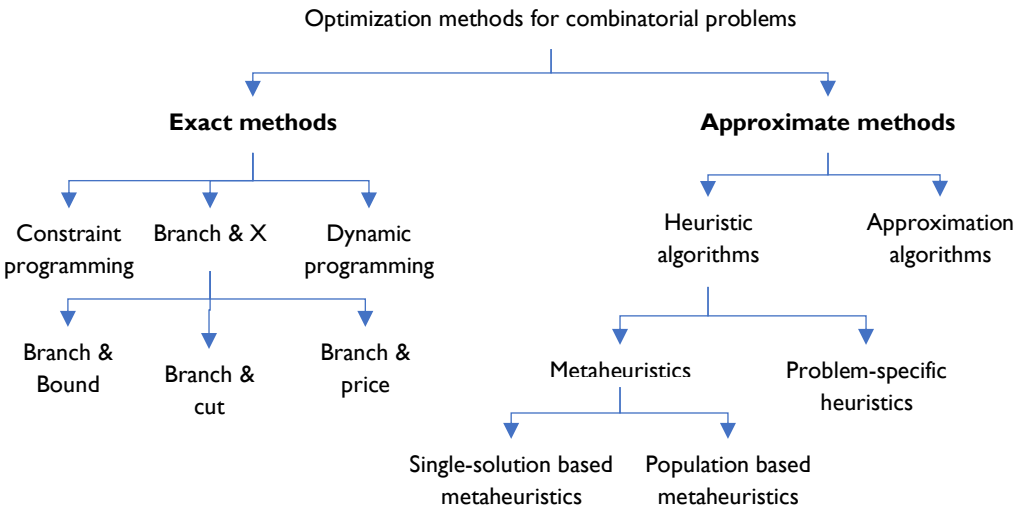


Figure 26 – Adaptation of categorisation of optimization methods (Talbi, 2009)

Chapter 4 describes how our problems is similar to the time-driven RCCP problem. The time-driven RCCP problem is an NP-hard problem (Kis, 2005). An optimal algorithm for a problem in this class would require a number of computational steps that grows exponentially with the problem size. Hence, solving the RCCP problem to optimality may be difficult for large problem instances (Hans, 2001).

As we find that NP-hard problems cannot be solved to optimality in polynomial time if P ≠ NP, then we may have to consider heuristic methods are they are more likely to find feasible solutions within limited time. Therefore, we review exact and heuristic approaches to similar problems in the following sections. Section 3.4 covers exact methods. Section 3.5 covers approximate methods.

3.4. Exact methods

Exact methods are often applied to small-size instances. The set of exact techniques covers a broad set of methods.

Figure 26 shows a categorisation of optimization problems. For exact combinatorial optimization problems, we distinguish Branch & X (Branch & Bound, Branch & Cut, Branch & Price), constraint programming, dynamic programming and algorithms based on artificial intelligence such as A* (Masmoudi, 2011). However, the latter is specifically fit for the vehicle routing problem.

The time-driven RCCP problem is formulated by De Boer (1998) as a mixed integer linear program (MILP). That is to say that is a generalization of a linear program, where some variables are integer. Formulating and solving ILP problems is called integer linear programming. There are numerous combinatorial optimization algorithms for finding feasible or optimal solutions to these programs. These methods use well-known algorithms such as branch and bound or branch and cut. Numerous solvers are available for ILP problem formulations that are based on these algorithms. These solvers are often based on similar solution methods based on enumerative approaches or cutting plane methods (Hans, 2001). We expect that completely enumerative approaches are unlikely to yield promising results as the solution space is too large to enumerate due to the combinatorial nature of the problem. Therefore, we focus on implicit enumerative approaches. The most commonly used enumerative approach is called branch-and-bound where branching refers to enumerating parts of the solution and bounding refers to fathoming or pruning possible solutions through a comparison with known lower- or upper bounds.

Branch-and-cut methods combine cutting plane algorithms with branch and bound results. Cutting plane methods improve the relaxation of the problem to more closely approximate the integer programming problem, and branch-and-bound algorithms proceed by a sophisticated divide and conquer approach to solve problems (Mitchell, 2002). Many solvers are available that are most often based on branch-cut methods (Mittelman, 2019).

Hans (2001) proposes an exact branch-and-price algorithm to solve the RCCP problem modelled as a MILP. Branch-and-price combines branch-and-bound and column generation methods. In branch-and-price, the ILP problem is relaxed after which column generation is done at every branch-and-bound node, solving the LP relaxation.

3.5. Approximate methods

Given that the problem is NP-hard and problem instances are large-sized, we are unlikely to find optimal solutions within polynomial time, or even reasonable solutions within reasonable time. When optimizing such complex problems, there is a trade-off between computation time and the quality of solutions obtained. For the above reasons, we may have to resort to heuristic methods, or combinations of heuristic- and exact methods for finding solutions to our problem, that do not guarantee optimal solutions given sufficient runtime.

Figure 26 distinguishes two classes of heuristic algorithms. The first, specific heuristics, are used to solve specific problems or instances (Masmoudi, 2011). The second class of metaheuristics considers generic heuristics that can be applied to a vast number of problems (Osman & Kelly, 1996).

3.5.1. Constructive heuristics

Constructive heuristics build candidate solutions to optimization or decision problems iteratively, starting from empty solutions. They normally use heuristic functions that estimate for each solution component the benefit of including it into a partial candidate solution (Thomas & Ruiz, 2018). Several of these problem-specific constructive heuristics are proposed in literature. We evaluate them in relation to the adaptations to our problem.

The Incremental Capacity Planning Algorithm (ICPA) is a constructive heuristic fitting the RCCP problem proposed by De Boer (1998). ICPA incrementally adds additional (nonregular) capacity to timeslots until a solution is feasible. The ICPA heuristic is less suitable for extensions such as maximum capacity constraints and cost differentiation (De Boer, 1998). Our problem features capacity constraints: increasing non-regular capacity is not always possible as the number of system resources (i.e. workstations) is limited. Next to that we have cost differentiation for non-regular capacity. Capacity from workstations not requiring human resources is less costly than workstations that do require human resources.

Greedy heuristics pick the best element from a set of potential solution elements at each iteration. Kolisch & Drexl (1996) propose an adaptive search heuristic for hard scheduling problems, using a hybrid of priority rules and random search techniques. Greedy randomized adaptive search procedure (GRASP) is a constructive metaheuristic. It attempts to overcome the drawback of greedy approaches by introducing randomness in the solution construction process (Sevaux, Sörensen, & Pillay, 2018). GRASP creates a restricted candidate list of best elements and randomly picks one element from that list, rather than iterating on the best candidate. This way, iterations end in different solutions.

3.5.2. Metaheuristics

Meta-heuristics are general-purpose algorithms that can be applied to solve almost any optimisation problem (Caceres-Cruz & Arias, 2015). Meta-heuristics are a class of approximate methods designed to attack hard combinatorial optimization problems where classical heuristics have failed to be effective and efficient (Osman & Kelly, 1996).

Local search methods improve incumbent solutions. Local search methods start from feasible solutions and iteratively improve them until a local optimum is found. After finding a solution, the neighbourhood of a solution is the set of all feasible solutions in the vicinity of the solution that follows some distance measure. After each iteration the neighbourhood of a candidate is explored and the current solution is replaced with a better solution from their neighbourhood, if one exists.

Metaheuristics are solution methods that “orchestrate an interaction between local improvement procedures and higher levels strategies to create process capable of escaping from local optima and performing robust search of a solution space.” (Glover & Kochenberger, 2003). We describe some well-known metaheuristics, their applicability and expected performance to our problem. The metaheuristics we discuss: tabu search (TS) and simulated annealing (SA).

The key to success for a local search algorithm consists of the suitable choice of a neighbourhood structure, efficient neighbourhood search techniques, and the starting solution (Festa & Resende, 1995). Due to the large solution space of our problem (both in the number of timeslots and available

workstations), we have to make effective decisions in the neighbour structures, search techniques and starting solutions considered for local search methods.

We can change capacity levels by searching the neighbourhood structure of the incumbent solution, and change some variables that neighbour the incumbent solution variables. If we seek optimal solutions, we must define a neighbourhood structure so that all possible allocations can be reached.

Simulated annealing

Simulated annealing (SA) is a metaheuristic that combines local search and probabilistic methods introduced by Kirkpatrick et al. (1983). Simulated annealing mimics the cooling process of metals, which follows a progressive reduction in the atomic movements that reduce the density until a low-energy state is reached.

In each iteration of SA, neighbours of incumbent solutions are generated randomly, a probabilistic move is made to the neighbouring solution depending on its objective function and the incumbent solution value. If the neighbour solution is better than the incumbent solution, it is always accepted. If it is worse, it is accepted with a certain probability that is updated over iterations in the search. The acceptance of worse solution value neighbours allows for escaping of local optima. The cooling scheme determines the rate of cooling, and influences the acceptance rates through the annealing procedure. The cooling scheme determines the rate of cooling. Quick cooling may cause irregular solutions whereas slow cooling may cause stalling in local optima.

Masmoudi (2011) considers a simulated annealing adaptation for the RCCP problem under uncertainty where project plans and project schedules are modified successively.

Tabu search

Tabu search (TS) is another well-known metaheuristic able to avoid local optima first mentioned by Glover (2003). Contrasting SA, TS' method for escaping local optima is deterministic. TS uses memory, or a tabu list to avoid trapping in local optima. The tabu list contains information on the most recently visited neighbours to avoid them in (short-term) future iterations. TS explores a solution space by moving to the best solution space in the candidate list. To avoid traversing recent solutions, moves or solutions performed recently are added to the tabu list. When this solution or move is included in the tabu list, it cannot be chosen as the next solution.

3.5.3. Hybrid approaches

Until now, we distinguished between exact and approximate approaches to our problem. However, hybrid approaches could be considered. Matheuristics consider the combination of mathematical programming with heuristics (Fischetti & Fischetti, 2016).

De Boer (1998) proposes an LP-based algorithm to solver the time-driven RCCP problem. The precedence relations in the problem introduce integer variables, causing a hard problem. However, relaxing the precedence relations leads to an LP-formulation. Therefore, De Boer (1998) proposes an idea that uses LP iteratively, repairing broken precedence relations if necessary. They propose multiple ratios that can be used to narrow the time windows of work packages iteratively.

Gademann and Schutten (2001) propose an LP-based heuristics. Their improvement heuristic starts from a feasible solution and tries to improve the solution iteratively by changing the start- and

completion times of the work packages in the solution to mitigate nonregular resource usage. They also propose repairing heuristic similar to De Boer (1998). However, they differ in the strategy of repairing violated precedence constraints.

3.6. Conclusion

This chapter provides a theoretical framework for our problem. We find that the capacity planning problem integrated with the scheduling function can be regarded as a combinatorial problem that is in many regards similar to the time driven rough-cut capacity planning problem.

Alternatively, this chapter finds that the problem can be regarded as an iterative problem where there is feedback from the scheduling function the master problem of capacity planning. As the scheduling function can be considered trivial, we find that we can apply heuristics to obtain solutions that are evaluated using the scheduling function and a cost parameter.

4. Problem approach

Chapter 3 introduces two solution strategies for planning and scheduling problems in general: a strategy based on fully integrating the planning- and scheduling problems and a strategy based on iterating between planning and scheduling. We consider research question 3: *What strategies can we formulate to tackle our problem?* in this chapter.

Section 4.1 describes the characteristics of our problem so that we can compare it to methods described in literature. Section 4.2 positions our problem in an adaptation of the hierarchical planning framework from De Boer (1998). Section 4.2 finds two means of approaching our problem: either through integration or iteration. Finally, Section 4.3 and 4.4 considers solution space and time for our problem as important factors that may determine capacity plan quality.

4.1. Problem characteristics

We describe the properties of our problem and its components so that we can compare it to existing formulations of problems in literature and finally formulate our problem as an optimization problem.

In Chapter 2 we found that we can optimize for a single performance indicator given that hard constraints are met. The allocation of resources over timeslots creates a finite solution space. For each timeslot in the planning window, a finite number of decisions can be made on the number of workstations employed at processes for each workstation class. Therefore, our problem can be considered a combinatorial, rather than a continuous optimisation problem.

In Chapter 2 we provide a rationale for our research to consider deterministic demand and supply. We do not consider stochasticity or uncertainty in processing times and capacity levels nor in offsets and resource availability, neither do we consider uncertainty in the arrival/departure of trucks or demand. In other words: the transport plan with store orders or forecasts of these orders is the best estimate of true production we have and therefore use it to plan capacity with predetermined human- and system resource availability.

Our problem has properties similar to time-driven problems as we consider the departure times of trucks as hard constraints that must be adhered to, where we have deadlines rather than due dates.

We cannot employ unlimited capacity as we are constrained by the availability of both human resources and system resources. Some system resources require human resources for operation, both of these resources are available up to a finite number: the number of persons available in a shift and the number of (semi-)manual workstations at a process. Furthermore, there is a link between the human resources hired over both adjacent timeslots and processes, as their cost is incurred in shifts. As we described in the process of planning in Section 2, cost is incurred for an interval of timeslots rather than distinct timeslots for human resources. Next to that, these resources may be employed over system resources at different processes during their employment shift.

Demand for any production area in a store order constitutes (many) production units (cases/LCs). There is no fixed production time for these orders. We can state that the processing time for an order is variable as it is dependent on the number of workstations handling the order. The processing rate in one timeslot depends on the available capacity.

Furthermore, pre-emption within processes of store orders is allowed. I.e. the execution of any component process may be spread over different not necessarily adjacent timeslots. We can stall production for any order and continue producing in a later timeslot, without violating any constraint.

Chapter 2 describes the process flow through the distribution centre. If a store order constitutes demand for a production area, the flow sequence of production through that area is deterministic. There are clear precedence relations between processes in these areas. For instance, (automated) case picking cannot start before depalletizing has finished.

4.2. Positioning & strategy

This section discusses how our problem can be positioned in the hierarchical planning framework of De Boer (1998), and how solution strategies can be defined for that position. Chapter 3 reviews the hierarchical planning framework. We argue that we can position our problem in the tactical and tactical/operational phase.

Planning in the distribution centre is done for store orders for small timeslots of 10 minutes, normally seen in operational planning. However, we argue that our main problem of allocating capacity fits a tactical decision similar to the time-driven RCCP problem. We seek to allocate sufficient resources to cope with store orders within working windows (lead time and deadline constrained). There is maximum fixed capacity in the number of workstations available per area, meaning that non-regular capacity cannot be hired infinitely. Moreover, we deal with work packages rather than activities as exact order components (such as a detailed description of the items on a load carrier) are not available in the scope of the planning module as described in Chapter 2 (only available in the execution phase in the distribution centre).

Figure 27 shows the hierarchical planning framework adapted to the planning problem at the distribution centre. Initially, the planner plans resource using system and shift parameters. Then, the plan is iterated on using the scheduling algorithm. This scheduling algorithm uses the transport plan, the priority rule and the system parameters (offsets and routing between processes) to create feedback for the planning phase.

Currently scheduling is performed iteratively after planning in a what-if loop, serving as feedback to the planner. This is done by the demand scheduling algorithm described in Chapter 2, which functions as a performance validation method for the capacity plan. The demand scheduler considers the tactical/operational level of scheduling, sequencing and assigning jobs as it does not yet definitively indicate which persons or machines are assigned to which activity. Once the capacity plan is satisfactory, the next step is detailed scheduling or execution on the day of production where orders are allocated to workstations rather than a group of workstations. However this is not part of the planning process and scope.

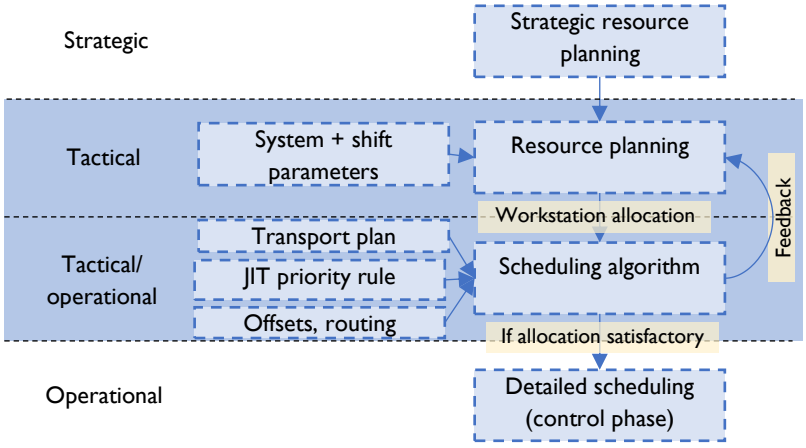


Figure 27 – Hierarchical planning framework (Adapted from De Boer (1998))

In the current situation, production planners plan and schedule demand iteratively to find feasible solutions. In order to find a feasible solution to our problem, we require feedback or integration between planning and scheduling, so we must either integrate or iterate between tactical planning and demand scheduling to find feasible solutions.

Chapter 3 discusses solution strategies for production planning and scheduling. We argue that we cannot consider the first hierarchical strategy, as we require feedback from the demand scheduling to planning to determine feasibility and performance of the schedule as Figure 27 shows.

The second, iterative approach is applicable and similar to the current (manual) process of planning. The third integrated strategy considers the full formulation of planning models and detailed scheduling. This approach integrates planning and scheduling approaches, which could be applicable to our problem as demand scheduling feedback is available in the planning phase. However, then we would have to consider the system and shift parameters, the transport plan as well as the routing and priority rules as a combinatorial problem, where we must adhere to the scheduling rules while minimizing capacity.

We conclude that we can approach the required link between planning and scheduling in two ways: either by iteration or integration. Either we can create capacity plans and evaluate performance by scheduling demand iteratively, or we could attempt to integrate these planning and scheduling phases by creating larger processing units, or work packages, and model these so that they reflect the performance indicators and constraints in the demand scheduler so that we can continue meeting the performance constraints described in Table 5.

4.3. Solution space and time considerations

This section describes the practical balance between time available for planning and the solution space of the planning problem.

Optimally, the best capacity plan can be obtained by enumerating all possible workstation allocations and picking the one that is feasible at the lowest cost. That would mean that for each workstation over every ten-minute timeslot, we could consider if using a workstation is optimal or not. To illustrate the magnitude such an enumerative approach, let us consider a total of 1008 timeslots over a planning

weeks horizon. With the binary availability of around 50 workstations (a workstation is either planner or not), the solution space will be too large to consider fully.

Note that in order to determine the optimal combination of workstation allocations, a means of validation is required. An allocation can only be optimal if it is feasible. The rules of the scheduling algorithm must be adhered to in order to assess feasibility. We observe that on average, the scheduling algorithm can be performed explicitly approximately 10 times per second on a regular laptop.

Typically, the input parameters for a planning week change frequently as a result of changes in the demand by customer order stores. A change in the input parameters suggests a change in the optimal capacity plan. Therefore, over the time period until the planning week, the capacity plan may require numerous alterations. For this reason, it is impractical to allow a means of planning to run in the order of hours. Preferable would be to allow a planning method to run in the order of minutes. Given the allowed runtime in the order of minutes and the observation that the scheduling algorithm can be performed approximately 20 times per second, a typical planning procedure given the input parameters can consider only several thousands of iterations, vastly smaller than the total enumeration of workstation allocations. This implies that in advance of assessing feasibility of allocations, informed decisions should be made on where the capacity plan can be improved.

4.4. Aggregating demand

By using the properties of the demand scheduling algorithm, orders can be aggregated, thus decreasing the number of orders to plan for as well as the planning complexity. Recall that the scheduling algorithm schedules in discrete intervals of ten minutes. This feature causes there to be no practical discrepancy between orders with common 10-minute departure time intervals. The reason for this is that scheduling is indicated in intervals of 10 minutes.

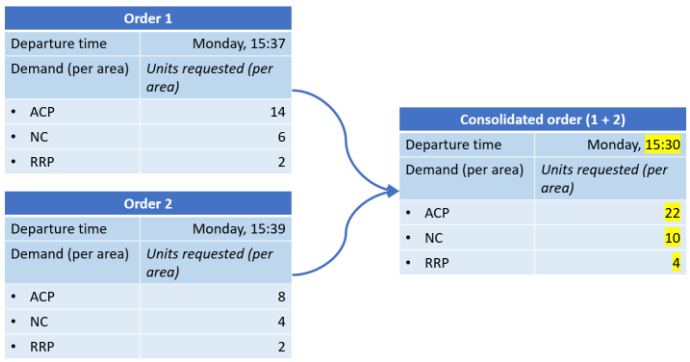


Figure 28 – Aggregating order demand

Figure 28 shows an aggregation example with two orders. The orders are both due in the same 10-minute time window. The scheduling algorithm can only indicate finished production in 10-minute intervals, e.g. finishing at 15:20 or 15:30. Next to the departure time, the differentiating attributes of these orders are the number of production units requested per area. Recall that the production time per unit as well as the production flow is uniform. This allows for the aggregation of orders. For this example, if the number of units requested per area by the consolidated order (1+2) is finished by 15:30, then demand for its constituent orders is fulfilled as well. We use this means of aggregating demand for both the integrated and iterative strategy.

5. Solution design: Iterative strategy

This chapter approaches the problem of planning and scheduling for the distribution centre through the iterative strategy. We propose using an iterative strategy that creates initial solutions with modified adaptive search procedure and improves them with a simulated annealing improvement metaheuristic. Section 5.1 considers the initial phase and Section 5.2 considers the improvement phase. This chapter motivates the decision for these optimisation methods and their implementation.

The iterative strategy alternates between a planning phase where workstation allocations are changed and a scheduling phase where customer store order demand is scheduled over these allocations. Performance and feasibility of the plan can be assessed given the feedback obtained from the scheduling algorithm. Chapter 2 finds that the scheduling aspect of the iterative strategy is trivial as it considers an algorithm that always amounts to the same outcome given a certain input. This aspect of the scheduling problem advocates the use of an iterative strategy to our problem as the effect on feasibility of a change in a capacity plan can be evaluated through the feedback from the scheduling algorithm.

Recall that the iterative strategy is based on rapid iteration between planning and scheduling phases. This means that each time the capacity plan is changed, the change must be evaluated using the scheduling function to assess feasibility and performance. While the goal of planning is to minimize the cost of operators, running time must be considered. Chapter 4 discusses that the input parameters to capacity planning can change frequently. These directly impact the best capacity plans that can be created. This results in the requirement that the planning algorithm should not be too time intensive as it may be performed many times. Therefore, we focus on creating capacity plans in the order of minutes.

Chapter 4 describes the vast solution space if we were to consider all combinations of workstation allocations. If a planning algorithm were to consider an enumerative approach where all combinations of workstation allocations over timeslots are considered, then the scheduling algorithm would have to be performed in a magnitude far greater than time allows. The scarcity of time and the limited number of iterations that can be performed prohibit using an enumerative approach.

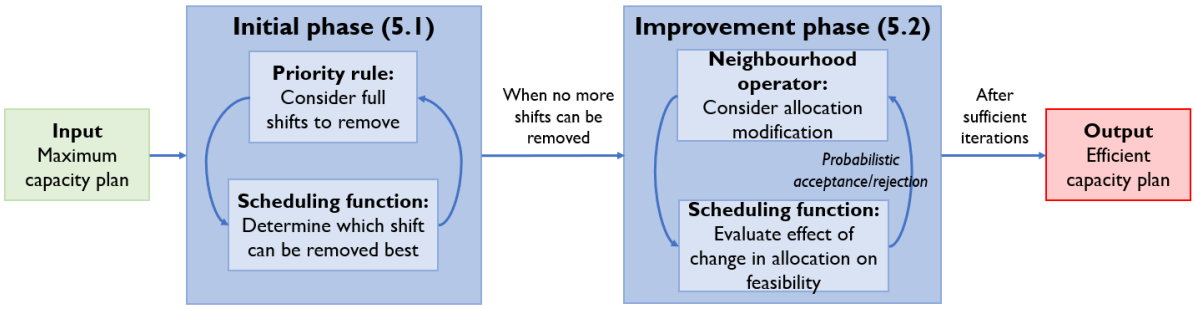


Figure 29 – Iterative strategy conceptual illustration

As the number of planning iterations will be limited to the order of thousands given a time limit in the order of minutes, decisions must be made on the neighbouring solutions to be considered during each iteration. Therefore, to minimize the time used for obtaining efficient capacity plans, the iterative strategy is split into two phases. Figure 29 illustrates these phases. The initial phase considers a limited a single neighbourhood operator only allowing operator shifts to be remove from a process so that

capacity plans with reasonable cost can be obtained quickly. The improvement phase considers more neighbourhood operators, allowing the exploration of a larger fraction of the solution space.

5.1. Initial phase

This section considers the initial phase of the iterative strategy. The initial phase considers iterative removal of shifts from an initial capacity plan that considers all workstations allocated over the entire planning week. This section provides a rationale for the decision to start with a maximum capacity allocation. Section 5.1.1 describes the prioritization of shifts to remove. The procedure of shift removal in the initial phase is based on the adaptive search heuristic as Section 5.2 describes.

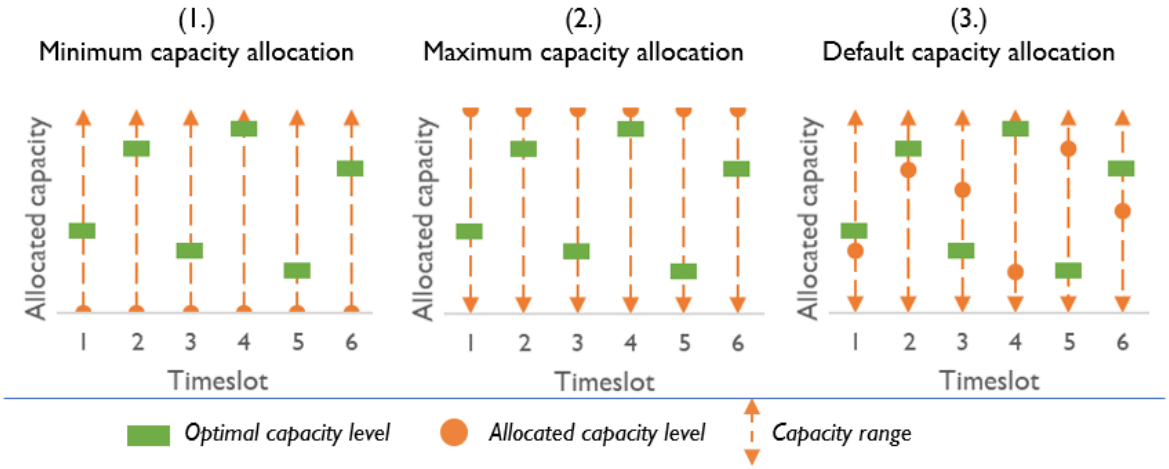


Figure 30 – Possible starting allocations for any plan.

First, we consider the starting allocation of the initial phase. From any plan, we can incrementally add or remove workstation allocations after which we apply the scheduling algorithm to determine the performance of the altered capacity plan. There are various allocations we can start from. Figure 30 illustrates the three main options and approaches. Table 6 discusses these initial capacity plans.

Initial capacity plan	**(1) Allocate no capacity at all processes over all timeslots**	**(2) Allocate maximum capacity at all processes over all timeslots**	**(3) Start with default allocation**
Initial feasibility	Never feasible initially as there is no capacity to process orders at all	Always feasible initially	Depends on order set instance
Required actions	Add capacity	Remove capacity	Adding or removing capacity depending on order set instance
Feedback from scheduling algorithm	Indication of overdemand and indication of which orders exhibit excessive lead times. Capacity must be added to mitigate these excessive lead times	Lead time of store orders: If the lead time obtained by scheduling is less than the allowed lead time, we can more likely remove capacity	Combination of (1) and (2)

Table 6 – Discussion of initial capacity plans

As the capacity plan changes, it must be evaluated by the scheduling algorithm to determine feasibility. The scheduling algorithm yields feedback stating which orders have lead times greater than

allowed (exceeding cut-off offsets) or cannot be processed due to under capacity at workstations. The feedback from the scheduling algorithm can thus be used to further improve the capacity plan.

We consider the three methods displayed in Table 6. If the first method of allocating no initial capacity is chosen, then the plan is never feasibility initially. Therefore, capacity must be added until it becomes feasibility. To add capacity, feedback from the scheduling algorithm can be used as the scheduling algorithm yields indications of infeasibility. For instance, if an order is scheduled recursively too often, the lead time may increase beyond a level that is permitted. If an order that departs at timeslot 200 shows a lead time of 150 timeslots given a capacity plan, whereas only 60 timeslots are allowed, then the lead time must be reduced to 60 timeslots or lower. To do this, capacity must be added in the range of timeslots 50 to 150. However, we do not know where exactly this capacity must be added nor at which process. We conclude that recovering from infeasibility is an intensive task that may require many iterations between planning and scheduling.

On the other hand, consider the second option of allocating maximum capacity initially. Capacity can be removed to improve the solution while maintaining feasibility. The scheduling feedback provides feedback that can aid in choosing where to remove capacity. If for instance an order realizes a lead time of 10 whereas a lead time of 70 is allowed, then we can more safely attempt to remove capacity than when the discrepancy between realized and permitted order lead time is lower as demand can be shifted further in the former case. Using this method, we mitigate infeasibility while rapidly iterating towards low cost solutions.

The last option of using a default allocation may require addition or removal operators depending on infeasibility. We decide that this option is unfavourable as it may require the same time intensive process as for the first initial allocation option. Therefore, we decide to use the maximum capacity allocation initially, where we prioritize on greater discrepancies between permitted and realized internal store order lead times.

5.1.1. Priority rule: determining regret

We introduced how the discrepancy between permitted and realized order lead times may be indicative of capacity removals that retain feasibility while lowering cost. This section describes how the initial phase prioritizes removing capacity at timeslots wherein orders that are to departure at that timeslot show greater lead time slack.

The initial phase plans for areas separately as human resources are not yet assigned over multiple processes during their shift to mitigate complexity. Initially, we assign all workstations at their maximum capacity: any schedule should be feasible at this point. If schedules are not feasible using maximum capacity, changes in customer order demand must be made that are available outside the scope of this thesis (Chapter 2).

In this initial procedure, human resources are dedicated to processes using multiples of the three available full-time shifts spanning an adjacent 24-hour time window, not yet considering other overlapping or part-time shifts. We use a priority rule to decide on promising shifts that could be removed from the allocation. Shifts with high priority are shifts with high minimum shift slack. Each timeslot wherein one or more orders are to depart has a deviation from store-quoted maximum lead time. We refer to this deviation as timeslot slack. This section introduces slack, how it is determined and why it is used for prioritization.

Figure 31 shows how slack per timeslot is computed given a capacity plan. Highlighted in red is the minimum permitted lead time of orders with their departure in a time window corresponding to the timeslot as found in the input parameters. The minimum of these permitted lead times is taken so that the lead time for all of the orders that depart in a timeslot is feasible. The scheduled lead time that is obtained by the scheduling algorithm must always be below this amount, otherwise infeasible plans are obtained (reconsider Appendix E for an example). The area lead time obtained through scheduling is highlighted in blue. The difference between the permitted and scheduled lead time is highlighted in green and referred to as lead time slack for orders that depart in a timeslot.

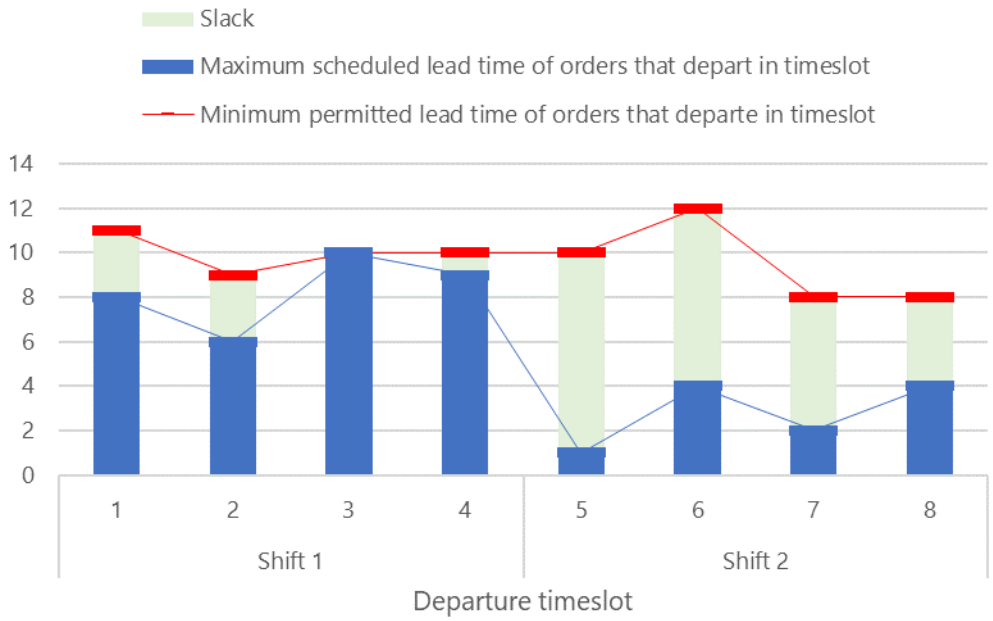


Figure 31 – Slack per timeslot illustrated

In the initial phase we consider capacity removal decisions. When capacity is removed during a timeslot, the scheduled lead time of order processing spanning this timeslot will either be stable or increase due to offsets created by scheduling. As the permitted lead time is fixed in the input parameters, the slack will either remain stable or decrease as a result of the removal decision. Accordingly, a greater slack will most likely decrease the chance of a capacity removal decision to be infeasible as demand of orders can be scheduled forward further. If the slack is 0 in a timeslot, a removal decision is far more likely to violate the permitted lead time than if the slack is a greater number.

Following the above arguments, it might be wise to prioritise on removing workstation allocations in timeslot where timeslot slack is high as we are more likely to retain feasible solutions while lowering cost, thus requiring less time for obtaining low-cost capacity plans.

As noted in Chapter 4, the likely number of total iterations that can be performed is in the order of thousands. If the capacity removal decisions would be considered for single timeslots, then a great number of iterations would have to be performed. Therefore, the removal of allocations in the full length of operator shifts must be considered. As we consider removing entire human resource shift working windows (the length of their employment) we must remove capacity for the entirety of their shift.

For this reason, the priority is adapted from timeslot slack to the minimum slack over the length of an operator shift. In the case of Figure 31, the minimum slack for the first shift is 0, while it is 4 for the second shift. In this case, it is probably wiser to attempt removing an allocation over the timeslots spanning the second shift.

If minimum shift slack is zero for any timeslot in a shift, we cannot expect to remove substantial amounts of capacity. Removing capacity likely induces increased lead time beyond feasible levels when slack approaches zero. We prioritize on the highest minimum slack in a shift time window as we expect the lowest probability of violating lead time constraints.

Adaptive search is a combination of priority rules and random search techniques which relies on an adaptation of the solution space. Kolisch & Drexl (1996) show that it is effective for the NP-hard resource-constrained project scheduling problem compared to other heuristics. The adaptive search procedure described by Kolisch & Drexl (1996) is a constructive heuristic differing from ours that is a local search procedure. However, we can apply the regret-based random sampling method based on the slack lead times of orders. Section 5.1.1 demonstrates how determining regret works. We could iteratively remove human resources allocated in their shift and their corresponding workstation if the resulting capacity plans remain feasible. Section 5.1.2 describes this procedure.

Shift	Departure Timeslot	Scheduled lead time	Permitted lead time	Timeslot slack z_t	Shift slack $\boldsymbol{z_s}$
1	1	8	11	3	**0**
	2	6	9	3	
	3	10	10	0	
	4	9	10	1	
2	5	1	10	9	**4**
	6	4	12	8	
	7	2	8	6	
	8	4	8	4	

Table 7 – Regret calculation for a sample plan

The minimum area slack in a shift determines the area regret factor r_s for shift s. Regret factor $r_s = z_s - \min_x z_x$ where z_s is the minimum slack of all orders with departure times in the timeslots corresponding to the operator shift. Table 7 shows a slack calculation for a sample plan. In this case, the shift regret for shift 1 is 0, the shift regret for shift 2 is 4. Therefore, we aim to prioritise on removing capacity for timeslots in shift 2.

As high maximum area lead times indicate low improvement potential, we use a priority rule that is based on the shift regret factor r_s. P_s denotes the probability of iterating on shift s. The shift probability P_s is determined by:

$$P_s = \frac{(r_s + 1)^\alpha}{C}, \qquad C = \sum_s (r_s + 1)^\alpha$$

Where C is the normalizing constant and α the bias factor. The bias factor regulates the degree of stochasticity in constructed solutions. When $\alpha = 0$, sampling is uniform. When $\alpha = \infty$, sampling is deterministic; in that case the shift with the highest regret factor will always be iterated on.

5.1.2. Adaptive search removal procedure

We remove operator shifts and their workstations iteratively based on their priority as long as resulting allocations are feasible. The initial phase does not yet consider employment of human resources over system resources at different processes during their shift. For this reason, the heuristic can be performed for processing areas separately. Figure 32 shows a flowchart of the initial phase. We describe the procedure for a single area.

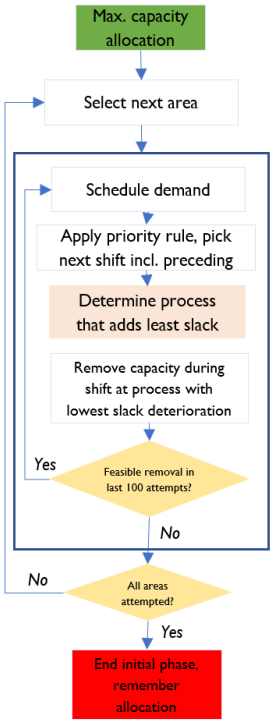


Figure 32 – Initial phase procedure

The heuristic chooses shifts iteratively based on the priority rule. The heuristic evaluates removing 2 shifts at the same process that precede the current shift in time. The preceding shifts are considered because lead time for an order may stem from offsets in timeslots prior to the shift during which the order departs. Only two preceding shifts are considered as they constitute 16 hours prior to the departure time of the order departure, where in practice the maximum lead time is 12 hours. This way, all shifts that cause lead time can always be considered. For instance, if maximum workstation capacity is allocated in a shift window, its succeeding shift may exhibit low lead times even if there are zero workstations planned in that shift time window. In that case, it is best to remove workstation allocations from the preceding shift, rather than the shift that was sampled initially.

After picking a shift based on the calculated probability, we add two of its preceding shifts to be evaluated as well because of the lead time phenomenon described in the previous paragraph. We attempt to remove capacity in any shift window in the allocation evaluation list separately. For any of these time windows, lead time increases due to the removal of allocations may stem from any process in the area. Therefore, we consider removing workstation allocations separately in all processes in the area.

We expect that we can remove multiple workstations in a class per process early in the constructive phase as the distribution centre typically has a sufficient number of spare workstations. Therefore we consider a rule for the quantity of workstations and their respective operators to be removed for a selected shift in each iteration. Furthermore, we expect greater computational effort in fully scheduling an allocation than in finding indications of infeasible allocations, advocating the multiple workstation removal decision. The removal rule is expected to significantly reduce the number of required iterations in the early phase of the constructive heuristic. To determine the number of workstations d_{ks} to be removed, a function based on parameters w_{ks} (the number of workstations currently allocated for a process k in during timeslots in shift s) and i (the number of iterations for which no feasible removal was made) is used.

The number of removed workstations d_{ks} at a process k in a shift s can only be in the interval $[w_{ks}, M_{ks}]$ where M_{ks} is the maximum number of workstations at a process k in shift s. Therefore, a maximum of $M_{ks} - w_{ks}$ workstations can be removed during a shift. The value for parameter i increases as feasible allocations are not found and resets to 0 when a feasible removal is made. With

an increase in i, the likelihood of a feasible high number of workstations removal decreases. To conclude, the probability of successful sampling from removing a greater number of workstation allocations decreases with the probability of finding a feasible solution.

$$d_{ks} \in U\left[1, \frac{M_{ks} - w_{ks}}{i}\right] \cap \mathbb{Z}$$

The constructive heuristic continues until a stopping criterion is satisfied. This stopping criterion i_{stop} is based on the same parameter i used for the removal multiple rule. The heuristic is considered finished when it fails to obtain any feasible solution improvement for i_{stop} iterations. i_{stop} is chosen so that probability of finding an improved solution in future same area iterations is considered sufficiently small. The procedure is finished when the implied probability of removing a shift allocation is considered sufficiently small. We set the rule so that the adaptive search procedure stops iterating for an area after not finding improvements for 100 iterations. This number is chosen as for higher values of i_{stop} the gain in solution cost is limited while the running time increases significantly (Appendix B). Furthermore, the value for the bias factor α used in determining priorities is set to 1 as it provides the best balance between solution time and cost.

Parameter	Description	Value
α	Bias factor priority rule (see Appendix B)	1
i_{stop}	Stopping criterion, stop initial phase after i_{stop} iterations without feasible workstation removal	100

Table 8 – Parameters for constructive adaptive search procedure

An exception in the removal of system resources requiring human resources is introduced for shifts in process *Depalletising*. Chapter 2 introduces a constraint on the ratio of semiautomatic to automatic workstations in this process. Therefore, we must remove several automatic workstations equivalent to the ratio amount if an attempted removal in *Depalletising* is expected to violate the constraint.

5.2. Improvement phase

This section regards the improvement phase that uses a simulated annealing metaheuristic to improve the solution obtained in the initial phase.

However, considering only the removal neighbourhood operator in the initial phase has an effect on the solution space: the solution space is not fully connected, thus not all possible workstation allocation combinations can be evaluated. For example, the initial phase only considers removing allocations workstations over the entire length of a shift. Other changes that could be considered are moving a human resource over workstations or having them occupying workstations at different processes during their shift. These changes could ultimately lead to less operators being required to fulfil demand following more optimal utilization of workstations.

Plans generated by constructive heuristics such as adaptive search are not guaranteed to be locally optimal, it is almost always beneficial to apply a local search to attempt to improve each constructed solution (Festa & Resende, 1995). Therefore, we expect that solution quality can still be improved significantly after the initial phase mainly by considering other operators that allows us to connect to larger portions of the solution space.

The solutions obtained from the greedy constructive heuristic can be improved using metaheuristics. Simulated annealing is a metaheuristic capable of escaping local optima such as those obtained by the initial phase. Simulated annealing explores neighbouring solutions and initially accepts many worse neighbouring solutions, eventually cooling down and decreasing the probability of accepting neighbouring solutions, causing the solution values to converge. Simulated annealing has been applied to many combinatorial optimisation problems (Blum & Roli, 2003).

Tabu search creates a candidate list of neighbours to the current solution, then picking the best solution from this list. To determine the best neighbour, all of the possible candidates must be evaluated by the scheduling function, as only feasible neighbours can be considered. This would mean that at each iteration, many scheduling function iterations would have to be made. Chapter 4 described that the solution space is vast and that the number of available iterations is limited. At the same time we note that, especially early on, many quick wins can be made as some operators were previously neglected. Therefore, we decide to use simulated annealing rather than tabu search to improve the initial phase as the tabu search procedure would likely take far more time to improve.

Section 5.2.1 describes how we design the simulated annealing algorithm for our problem. Section 5.2.2 describes how the algorithm is implemented.

5.2.1. Simulated annealing design

Simulated annealing optimizes the cost function by exploring the neighbourhood of the current point in the solution space. The choice of neighbourhood operators is often critical in the success of the simulated annealing heuristic (Aarts & Korst, 2005). Several operators are available for constructing neighbouring solutions. We consider the allocation of human resources to workstations as subjects to the neighbouring operators. Several operations are possible:

1. Move operator shift allocation at a workstation to another shift time window
2. Exchange operator allocated to workstation with overlapping shift (e.g. part-time shifts)
3. Remove entire operator shift and workstation allocation
4. Add entire operator shift and workstation allocation

The choice of the above neighbourhood operators makes the solution space connected as all enumerations of capacity can be achieved from incumbent solutions by using a multiple of these operators. However, we decide to neglect the fourth operator, or adding an entire operator shift.

The shift adding operator has been tested in practice, however we found that the algorithm had trouble converging to better solutions when it was included. Despite neglecting the addition of entire operator shifts, move and swap operators promote the exploration of the majority of the solution space.

We decide that the neighbour operators can only be performed if they result in feasible solutions. We neglect the infeasible solution space and penalty parameters for the same reasons introduced in Section 5.1: recovering from infeasible solutions is expected to take many iterations whereas exploring the infeasible domain is unlikely to provide far greater solutions as most of the solution space can be explored by considering the above operators 1-4 only in the feasible solution space.

5.2.2. Simulated annealing algorithm

Neighbouring allocation s' feasibility is evaluated using the scheduling algorithm. If neighbouring solutions are feasible (adhering to constraints introduced in Chapter 2, then a cost $f(s')$ for the allocation as introduced in Chapter 4 is incurred. The acceptance probability p_{accept} of a neighbouring solution is determined by the cost of incumbent solution s, the cost of neighbouring solution s' and current temperature T. Solutions are evaluated by the cost function $f(s)$ based on the hourly cost matrix described in Chapter 4. If cost $f(s')$ for a neighbour s' is better (lower) than cost $f(s)$ for incumbent solution s then s' is always accepted. If the neighbouring solution value is worse (higher), then it is accepted with a probability dependent on the current temperature and difference in cost with the incumbent solution.

$$p_{accept}(T, s, s') = \begin{cases} 1, & \text{if } f(s') \leq f(s) \\ e^{-\frac{f(s')-f(s)}{T}}, & \text{otherwise} \end{cases}$$

The choice of parameters influences solution progression and runtime. Table 9 lists the parameters used in simulated annealing. The pseudocode in Figure 33 shows how these parameters influence performance. Current temperature c regulates the probability of accepting worse solutions in the current markov chain. High temperatures imply greater probabilities of accepting worse solutions. Often, the initial temperature c_0 is chosen so that the acceptance ratio $\chi(c) = \frac{\text{number of accepted solutions}}{\text{number of proposed solutions}}$ is close to 1 initially (Aarts & Korst, 2005). When $\chi(c) \approx 1$, simulated annealing acts as a global optimisation method as any neighbouring solution can be accepted to become the incumbent solution. As temperature decreases, simulated annealing converges to a local search method. The probability of accepting worsening solutions decreases incrementally, thus decreasing the probability of escaping local optima as annealing progresses.

Appendix B finds that $X(c) \approx 1$ for c_0 > 10000, therefore we use 10000 as the initial temperature. Also, $X(c) \approx 0$ when $c < 1$, so 1 is set as the stopping temperature. Appendix B explains that we can evaluate around 300 chains given the runtime constraints (set to 10 minutes to compare strategies). As we stop the algorithm after 10 minutes for a fair comparison with the integrated strategy, the temperature cooling parameter α is set to 0,97 so that we expect to finish in around 10 minutes.

Parameter		Value
k	Initial markov chain length; finite length of markov chain	Starting at k=5, incrementing by one after every 10 chains
c_0	Starting temperature; initial value of control parameter	10000
c_{stop}	Stopping temperature; final value of control parameter	1
α	Temperature cooling parameter	0,97

Table 9 – Simulated annealing cooling schedule parameters

A dynamic cooling schedule may be used to regulate the search for acceptable solutions and runtime. Initially, many neighbouring solutions may provide improvements as the constructive heuristic proposed in Section 3.5.1 does not take into account hourly cost differences nor a second shift set. As annealing continues, the acceptance ratio $\chi(c)$ continues to decrease. As we expect many neighbouring solutions to be improving solutions, we consider starting with a low value for k, iteratively increasing k with a decreasing probability p_{accept} of improving the incumbent solution.

Appendix B tests two scenarios: one where the markov chain length is constant at k = 20, and an incremental strategy where the markov chain length start at a low number and gradually increases. We test several configurations (Appendix B), and find that a start at k = 5 and increases every 10 iterations performs best for our test instances. This incremental markov chain length parameter outperforms the constant option (Appendix B). We use the incremental markov chain length parameter.

Algorithm Simulated Annealing

input: Problem instance I
$s \leftarrow$ Initial solution from constructive heuristic
$c \leftarrow c_0$;
$k \leftarrow 5$;
while $c < c_{stop}$ **do**
  for i **in** k **do**
    $s' \leftarrow$ Neighbour(s, P_n) pick neighbour of incumbent with probability P_n
    if $f(s') \leq f(s)$ **then**
      $s \leftarrow s'$;
      if $f(s) < f(s_{best})$ **then**
        $s_{best} \leftarrow s$;
    else
      $s \leftarrow s'$ with probability $P(c, s, s') = e^{-\frac{f(s')-f(s)}{c}}$
    end if
  end for
  $c \leftarrow \alpha c$;
  $k \leftarrow k + 1$ every 10 iterations
  end if
end while
output: Optimized solution s_{best} for I

Figure 33 – Simulated annealing pseudocode

Some neighbourhood operators may become decreasingly able to find feasible solutions. For instance, a shift removal decision will less likely result in feasible solutions as the heuristic progresses. However, the scheduling algorithm is performed nonetheless at computational cost. Therefore, we want to decrease the frequency of picking operators that result in infeasible allocations as the simulated annealing algorithm progresses.

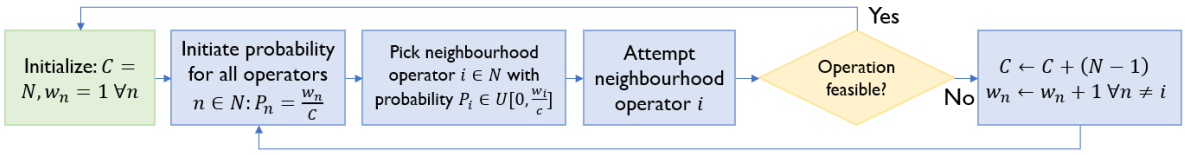


Figure 34 – Probability of choosing a neighbourhood operator decreases as infeasible solutions are found

In order to lower the number of iterations, we propose a neighbourhood operator decision rule based on a probability factor that is determined by the rate of success of that neighbourhood operator. At

the start of each markov chain the probability of picking an operator is equal for all four neighbourhood operators. If an infeasible allocation is created by the operator, we decrease the probability of choosing that operator in the current markov chain.

To regulate the choices of neighbourhood operators, we propose a scheduling rule that updates the frequency of picking a certain operator in Figure 34. With the proposed decision rule, we expect to lower the number of required iterations. We cannot expect (especially in the later stages) that the probability of a successful change by a neighbourhood operator is equal over all operators. For instance, we expect that few shift removal decisions can be made. Considering the shift removal operator as often as other neighbourhood operators would require many costly iterations that are unlikely to optimize the solution.

5.3. Conclusion

This chapter explains how an iterative strategy can be performed using a two-stage heuristic approach. We find that turning infeasible solutions to feasible solutions is complex, therefore we want to keep the created plans feasible at any time. Initially, we can start from a solution that assigns operators of all workstations during the entire planning horizon. Then, we iteratively remove entire operator shifts and their corresponding workstations intelligently based on a feedback metric from the scheduling algorithm (the slack lead time). The probability of removing a shift is then based on the regret-based sample of this slack metric for entire shifts.

We find that the initial heuristic yields decent solutions. However, it neglects several aspects of the planning problem. For instance, the initial phase does not consider hourly cost rate differences, part-time shifts and employing operators over multiple processes during their shift. To explore a larger portion of the combinations that can be made, we propose using a simulated annealing heuristic.

The simulated annealing heuristic uses more operators that allow exploring a greater portion of the solution space. However, due to the relatively time-consuming scheduling algorithm, intelligent decisions must be made as the number of iterations over time is relatively scarce. We find that, despite not being able to explore the entire solution space due to neglect a shift-adding operator, we are able to find significant improvements from the initial phase.

6. Solution design: Integrated strategy

This chapter regards the integrated strategy approach to the problem in the distribution centre. Through integration, we seek to find the right combination of scheduling and planning variables that minimizes the cost of operators. This way, the problem can be viewed as a combinatorial optimisation problem.

Combinatorial optimisation involves a set of a discrete set of feasible solutions over which the goal is to find the best solution. But how can we model our problem so that we can find feasible solutions using combinations of planning and scheduling? We resort to literature where we find that our integrated problem is in many regards similar to the time-driven RCCP problem, but that there are several key differences that hinder the solution methods proposed in literature. We describe this analogy and its difference in Section 6.1.

The number of combinations for the variables yields a vast solution space for the NP-hard problem that is hard to optimise for through conventional methods based on branch-and-cut. Therefore, we aggregate or simplify several aspects of the integrated problem to reduce the problem size, while attempting to maintain feasibility. Section 6.2 describes these changes. Finally, section 6.3 describes how the reduced problem can be modelled as an ILP by introducing scheduling and planning buckets.

6.1. Comparison to time-driven RCCP

This section covers combinatorial optimisation problems related to planning and scheduling. We discuss the properties of these problems and how they align to the integrated problem for the distribution centre. Our problem requires generic precedence relations, as we described in Chapter 2. We require the ability of hiring additional (nonregular) capacity in the tactical phase. The properties of the formulation of the time-driven RCCP problem in Gademann & Schutten (2001) are similar to the general properties of our problem, albeit with some adaptations that we describe in the forthcoming section.

De Boer (1998) provides a comparison between activities (as found in RCPSP problems) and work packages (as found in RCCP problems). Table 10 shows the characteristics of the smallest processing units found in scheduling problems in comparison to the formulation of aggregated work packages in the RCCP formulation.

Characteristic	Activity (scheduling)	Work package(planning)	**Our problem**
Modes	Multiple	Not predetermined	Single mode
Duration	Fixed for each mode	Minimum duration with variable duration	Variable duration
Pre-emption	Not allowed	Pre-empt resume	Pre-empt resume
Resource rate	Constant	May vary during execution	May vary during execution
Capacity requirements per resource	Fixed number of units for each mode (e.g. 3 men)	Total of res. time units (e.g. 120 man hrs.)	Total of. units (e.g. 120 cases)

Table 10 – Work-packages and activities compared, from De Boer (1998), extended with our problem characteristics

If we consider work packages as the demand for production areas of store orders in our problem, the characteristics of our problems work packages are similar to those found in RCCP formulations, rather than those found in the RCPSP (Table 10) as our comparison shows. We describe each characteristic. We have no minimum or fixed duration for processing any store order. If more capacity (workstations) can be allocated to an order in a timeslot by using more workstations an order can be processed in any variable duration. Processing rate is a continuous function of available resource capacity in any timeslot.

In Chapter 2 we also found that pre-emption should be possible, as order processing can be stalled in a queue prior to workstations, or in their respective buffers. The resource rate can vary during execution with the capacity allocated in a single timeslot. If more workstations are opened, more capacity is available and the processing rate can increase.

We find that our problem is in many regards similar to the time-driven RCCP problem. However significant differences exist. We highlight the differences in this section.

The time-driven formulation of De Boer (1998) assumes no limit on the amount of non-regular available capacity. In our problem, we have limited capacity available as there is a limited number of system resources available for each workstation class, and an upper limit on the number of human resource per available shift time window. This means that our problem can be infeasible due to not enough capacity being available in a certain time window, which indicates that we cannot plan for those store orders or must adapt input to the planning, for which the options available (outside the scope of this thesis) are described in Chapter 2.

The time-driven RCCP considers the hiring of additional capacity as a continuous variable where for example 10 extra hours can be hired for a week of production. In our problem, extra capacity is hired in integer multiples, where the resulting added capacity is considered continuous. For instance, employing one extra workstation in a workstation class in a (range of) timeslots incurs extra capacity as a linear function of that added unit depending on the workstation production rate for that time window. We have additional constraints for intermediate production levels. Buffer utilization cannot (virtually) be greater than 1 as the demand scheduler cannot handle this (Chapter 2)

In our problem there is no regular- or free capacity. Although automated workstations are considered far cheaper than non- fully automated resources, they still incur some cost due to for instance degradation. In other words: automated resources are considered nonregular as there is a significant preference for not using them over using them.

In none of the time-driven RCCP formulations found in literature, we find indications of employment of human resources over multiple adjacent time units. In our problem, cost for workstations is incurred for the duration of shifts if those workstations require human resources. If workstations use human resources, then cost is incurred for the entire shift duration. Next to that, we have cost differentiation during the week: some timeslots may be favoured for employing semi-automated or manual workstations as they may be less expensive during daytime. These differences are highlighted in Chapter 2.

To the best of our knowledge, no RCCP formulation in literature considers adding nonregular resources (employment of human resources) that are eligible for multiple resources during their shift. We mentioned that cost for employing workstations in classes with manual or semi-automatic

resources is incurred for the duration of shifts. During these shifts, human resources may switch between system resources supporting human operation in multiple processes. E.g. during one shift they may be active at both a semi-automatic depalletizing station and a semi-automatic ACP station.

Generally, RCCP problems consider time periods of weeks to years with time buckets in that smallest order, and a number of time slots usually in the order of tens (De Boer, 1998), (Hans, 2001). If we consider timeslot sizes used in the scheduling algorithm (10 minutes), we require 6*24*7 = 1008 timeslots in a week. Next to that, if we state that production area orders in store orders are the smallest unit of processing, we typically require thousands of orders in a single week.

This section finds that we can model the problem similar to the RCCP problem, but that there are some significant differences that must be accounted for. Section 6.2 introduces how some of the input parameters of our problem must be altered if we wish to use an approach similar to that of the time-driven RCCP.

6.2. Processing input parameters for ILP formulation

This section describes how the store orders and system parameters can be translated through pre-processing steps in order to reduce the problem size and number of computational steps required for solving the ILP.

Typically, formulations such as for the time-driven RCCP problem entail a planning horizon with at maximum several dozen of planning timeslots. However, in our problem there are 1008 ten-minute timeslots every week. Combining with 47 total workstations that can be allocated, the solution space is large. For this reason it is very likely that a branch-and-cut approach through integer programming is unlikely to find obtain efficient capacity plans. Therefore, we take several steps to aggregate and process the original parameters so that we can find solutions through integer programming.

With these processing steps we deviate slightly from the real scheduling function feedback. We do not fully mirror the scheduling function when modelling the ILP through these pre-processing steps. For instance, we decide to aggregate time from 10 minutes to scheduling buckets consisting of multiple timeslots. This has effects on the scheduling function as for instance offsets may be defined in multiples of 10 minutes smaller than 1 hour in total. While we provide upper bounds, these decisions may not translate well into the true scheduling performance. In other words, we may create allocations that seem to not violate constraints in the ILP model but do provide errors or warnings in Vanderlande's TOP planning application. We test the effect of incorporating these processing steps on validity in Chapter 7.

We consider the following steps in this section: (Appendix A provides a more detailed version of why these steps are likely retain validity)

- Aggregating demand over time for uniform demand areas – We schedule demand in the order of scheduling buckets that consists of multiple timeslots rather than individual 10 minute timeslots. This is possible because orders must be processed in their order of departure.
- Creating an upper bound on the number of changeovers operators can make during their shift – If we allow an operator to change processes every 10 minutes, then the solution space is far greater than if the number of changeovers would be limited. Therefore, we introduce planning buckets that allow a limit number of changeovers per operator shift

- Constraining order working windows using cut-off times – If the maximum permitted lead time for an order is 10 hours, then we should only consider allocating demand from 10 hours prior to the departure time of an order until its departure as timeslots outside that interval would create an infeasibile capacity plan.
- Altering workstation productivity to account for breaks – In reality, operators can take breaks in the order of timeslots. However, as we create planning buckets we can no longer model these specifically. Therefore, adjustments in productivity can be made to account for these changes.

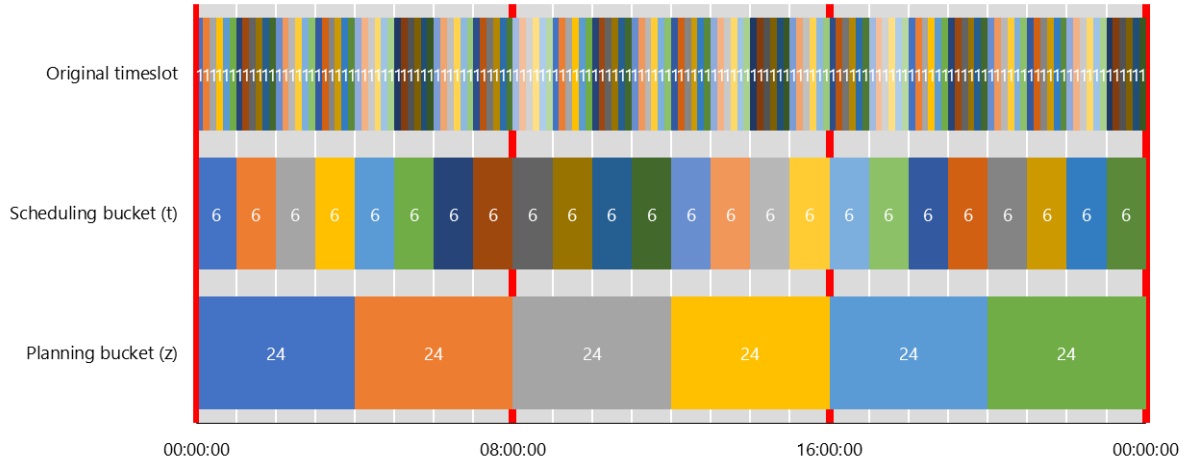


Figure 35 – Demonstration of time buckets and allocation buckets

Normally, the planning horizon is divided into 10 minute long timeslots, creating 1008 timeslots for a week. However, given the large multiple of combinations that can be made given such a long horizon, we decide to increase the size, scheduling in time buckets of α ten minute timeslots, in the case of Figure 35 this is 6 timeslots. Moreover, allowing human operators to change over workstations would create large number of combinations that is hard to explore. Therefore, we use allocation buckets that partition full-time shifts of 8 hours into β allocation buckets. In the case of Figure 35, an operator with a shift ranging from for instance 00:00 to 08:00 can be assigned two different processes during their shift.

6.3. Modeling the problem as an ILP

This section models the adapted problem as described in Section 6.2 as an ILP problem.

We are given a set of n work packages $J_1, J_2, \dots, J_n$ that must be planned on K processes. The work packages may belong to different demand areas. The work packages consider the workload for a single process after aggregation. We have a time horizon divided in T scheduling buckets that include α 10-minute timeslots. A work package J_j requires Q_{jk} production units at process R_k. The number of workstations that do not require operators available at a process R_k $(k = 1,2, \dots, K)$ in a time bucket is equal to $r_k$units. The number of workstations that do require operators at a process R_k in a time bucket is equal to n_k units. These workstations that at a process can process a maximum of TSR_k and TSN_k units per time bucket respectively at a process k.

x_{jkt} denotes the fraction of work package J_j performed at process R_k in time bucket t. Therefore, $x_{jkt} \cdot Q_{jk}$ units are processed for work package J_j in time bucket t on resource R_k. Section 6.2 introduces a working window for a work package. This working window is obtained using the departure times and minimum cut-off times of the orders that constitute the work package. Processing for work package J_j must be done in the time window $[R_j, D_j]$.

Chapter 2 introduces precedence relations between work packages. For each of the three final demand areas, there is a specific path of processes that all work follows. For instance, all work at *Depalletising* must be completed before work at *Palletising* can start. Therefore, there may be precedence relations between work package J_i and J_j denoted by $J_i \rightarrow J_j$.

The scheduling algorithm introduced in Chapter 2 prescribes two types of precedence relations. Jobs are scheduled recursively based on their departure time. A job with an earlier deadline must always be processed fully before a job with a later deadline can start. Furthermore, we noted that the process flow within areas must be strictly adhered to as described in the previous paragraph. Following the rules of the scheduling algorithm, work for work package J_i must be performed completely before we can start performing work for work package J_j.

We find that a solution $[x_{11}, x_{12}, \dots, x_{1T}, x_{21}, x_{22}, \dots, x_{2T}, \dots, x_{n1}, x_{n2}, \dots, x_{nT}]$ is feasible if all work packages are performed completely within their respective time windows if precedence relations are followed. These solutions may require the employment of workstations n_k that require human operators. In fact, some processes do not have any regular workstations available.

There are limits on the number of workstations that do and do not require human operators, as the number of workstations available per process is finite. We denote the maximum number of workstations that do not require human operators as $MaxR_{kt}$ workstations that require human operators as $MaxN_{kt}$. In practice, workstations that do not require human resources are expected to incur far less cost than workstations that do as they do not require human resources. We state that all workstations incur a base cost of 0.0001 per time bucket. This way, the objective function will minimize the number of workstations, however giving far greater priority to minimizing human operators as the next paragraph explains.

The cost for human resources is not incurred in regular time bucket intervals as they are not employed per time bucket. Instead, they are employed for the length of their shift. As the cost for workstation requiring operators stems mainly from the use of humans, we introduce a cost parameter C_s. Cost is incurred for S shifts $S_1, S_2, \dots, S_s$. These shifts correspond to the shift time windows introduced in Chapter 2. One workstation incurs a theoretical base cost of 1 per timeslot, increasing for more expensive time slots. We denote the number of human operators required for a workstation at process R_k as H_k. In each shift S_s with starting time bucket B_s and ending time bucket E_s, a total number of u_s human operators are employed. The window $[B_s, E_s]$ refers to the shift time window during which the human operators are allowed to perform work. Shift cost C_s differs per shift as incurred from the hourly cost matrix.

Human operators u_s are available for an entire shift. During their shift, they may change over processes. For instance, they may work at *Depalletising* in time bucket B_s whereas they work at *NC Picking* in time bucket $B_s + 1$. Section 6.2 introduces parameter β, through which the number of

changeovers is regulated. Therefore, rather than employing resources per time bucket, we change to planning buckets that constitute multiple time buckets.

Set and indices

t = Scheduling buckets, $i = 1,2,3, \dots, \frac{1008}{\alpha}$

j = Jobs (= store orders for a process with a deadline in (aggregated) time window$[t, t + \alpha)$ $j = 1,2,3, \dots, J$

$k \in K$ = Production processes {Depalletising, ACP, NC Picking, Xdock Staging, RRP Pick}

$s \in S$ = Shifts, used for indicating when operators can operate workstations.

$z \in Z$ = Planning buckets = Assignment length of operators, 8 hour shifts are partitioned in buckets of $\frac{8}{\beta}$ hours or $8\frac{\alpha}{\beta}$ timeslots

Work package parameters:

K_j = Process corresponding to production for work package j

Q_j = Demand of work package j in unit equivalents (case or LC)

R_j = Earliest possible scheduling bucket during which work for work package j can be performed

D_j = Latest possible scheduling bucket during which work for work package j can be performed

P_j = Work packages preceding work package j

O_j = Offsets corresponding to work package j

Process parameters:

TSR_k = Maximum production of a single workstation not requiring human resources for process k in a single scheduling bucket t in unit equivalent

TSN_k = Maximum production of a single resource workstation requiring additional human resources for process k in a single scheduling bucket t in unit equivalent

$MaxR_{kt}$ = Maximum number of workstations not requiring operators at process k in scheduling bucket t

$MaxN_{kt}$ = Maximum number of workstations requiring operators at process k in scheduling bucket t

H_k = Number of workers required to perform work at workstation that requires operators at process k

ρ = Minimum ratio in demand between semi-automatic and automatic depalletising

Shift parameters

A_t = Number of workers available to perform work in time bucket t

LB_s = Starting time bucket of shift s

UB_s = Ending time bucket of shift s

C_s = Cost of shift s, i.e. mean cost accounting for daily/ hourly cost variation

I_s = $\begin{cases} 1 \text{ if cost to be incurred for shift s (in target planning week)} \\ 0 \text{ if no cost to be incurred for shift s (outside target planning week)} \end{cases}$

LM = Lower bound scheduling bucket for work window with strong preference to mitigate

UM = Upper bound scheduling bucket for work window with strong preference to mitigate

V_z = Planning bucket in target week corresponding to z in the preceding week (empty for buckets in target planning week), so planning buckets in preceding week have $V_z \neq \emptyset$

W_z = Planning bucket in target week corresponding to z in the succeeding week (empty for buckets in target planning week), so planning buckets in succeeding week have $W_z \neq \emptyset$

E_s = $\begin{cases} 1 \text{ if shift S in target planning week} \\ 0 \text{ if shift S not in target planning week} \end{cases}$

F = Fixed cost parameter

Variables

x_{jt} = Amount of work for work package j done in scheduling bucket t (fraction of total work)

r_{kz} = Number of workstations that do not required operators planned at process k in scheduling bucket t

n_{kz} = Number of workstations that require operators planned at process k in scheduling bucket t

u_s = Number of human resources planned in shift s

f_t = $\begin{cases} 1 \text{ if any capacity used in scheduling bucket t} \\ 0 \text{ if no capacity used in scheduling bucket t} \end{cases}$

y_{jt} = Binary variable for precedence relations (indicating earliest possible start for work package j)

Objective function

$$Min\ c = \sum_{k \in K} \sum_{z \in Z} 0.001 r_{kz} + \sum_{s \in S\ if\ E_s = 1} C_s u_s + \sum_{z \in Z\ if\ L_z \in X} F f_z$$

The multiobjective function considers three main objectives. The first is to minimize the cost of workstations. Note that the coefficient is small, the priority is given towards minimizing the use of operators. The second is to minimize the use of human resources that are in the planning week. The third is to mitigate time buckets during which there is a preference to not perform any work in the distribution center at all.

Constraints

$$\sum_{t=R_j}^{D_j} x_{jt} \geq 1 \qquad \forall j \in J \qquad (1)$$

$$r_{kz} \leq MaxR_k \qquad \forall k \in K, \forall z \in Z \qquad (2)$$

$$n_{kz} \leq MaxN_k \qquad \forall k \in K, \forall z \in Z \qquad (3)$$

$$\sum_{j\in J} Q_j x_{jt} \le TSN_k r_{kz} + TSR_k n_{kz} \qquad \forall k \in K, \forall z \in Z, \forall t \; if \; t \in [L_z, U_z] \qquad (4)$$

$$TSR_k r_{kz} \le \rho n_{kz} TSR_k \qquad k = Depal, \forall t \in T \qquad (5)$$

$$\sum_{\tau=R_i}^{t-O_i} x_{i\tau} \ge y_{jt} \qquad \forall j \in J, \forall t \in \left[R_j, D_j\right], \forall i \in P_j \qquad (6)$$

$$\sum_{\tau=R_j}^{t} x_{j\tau} \le y_{jt} \qquad \forall j \in J, \forall t \qquad (7)$$

$$u_s \ge \sum_{k\in K} n_{kz} H_k \qquad \forall s \in S, \forall z \in Z \; if \; L_z \in [B_s, E_s] \qquad (8)$$

$$\sum_{k\in K} MaxN_{kz} H_k \le A_z \qquad \forall z \in Z \qquad (9)$$

$$r_{kz} \ge r_{kW_z} \qquad \forall k \in K, \forall z \in Z \; if \; W_z \ne \emptyset \qquad (11)$$

$$n_{kz} \ge n_{kW_z} \qquad \forall k \in K, \forall z \in Z \; if \; W_z \ne \emptyset \qquad (12)$$

$$r_{kz} \le r_{kV_z} \qquad \forall k \in K, \forall z \in Z \; if \; V_z \ne \emptyset \qquad (13)$$

$$n_{kz} \le n_{kV_z} \qquad \forall k \in K, \forall z \in Z \; if \; V_z \ne \emptyset \qquad (14)$$

$$Mf_t \ge \sum_{k\in K} (r_{kt} + u_{kt}) \qquad \forall t \in [L, U] \qquad (15)$$

$x_{jt} \in [0,1]$

$y_{jt}, f_t \in \{0,1\}$

$r_{kt}, n_{kt}, u_s \in \mathbb{Z}$

$M =$ Large integer

Constraint (1) ensures that all work for work package j is performed within its available working window $\left[R_j, D_j\right]$. Constraints (2) and (3) guarantee that no more regular and nonregular workstations can be employed in any timeslot than the maximum number of system resources available for them respectively. Constraint (4) weighs demand for work packages with the planned amount of work performed for those work packages so that the sum of those products is smaller than the amount of capacity available for processing it in that timeslot. The required minimum ratio in capacity between system resource that do and do not require human operators at the depalletising process is formulated in constraint (5).

Constraints (6-7) define the precedence relations in production flow between work packages that are introduced in Chapter 2. Following these constraints, any work on work package J_j cannot start before all its predecessors $J \subset P_j$ have fully finished processing (5). Furthermore, work package J_j cannot start before work package J_{j-1} at the same process has fully finished processing (7).

Constraint (8) ensures that the number of human resources planned in shifts is greater than the number of planned workstations that require human resources (weighed with their respective number of operators). Constraint (9) guarantees that no more nonregular workstations, weighed with the number of human resources required for their operation, are used at all processes than the total number of human resources available in a shift time window.

Shifts may overlap. For instance, when we have a part-time evening and full-time evening shift. Therefore the sum of humans allocated in these overlapping shift timeslots must be greater than the

number of workstations in a process using human resources weighed with those workstation's respective number of required operators.

Constraints (11-14) mirror the extension of the planning window as described in Chapter 2. The amount of capacity planned in the appended preceding window must be smaller than the capacity planned in the days corresponding to the current planning week. The amount of capacity planned in the appended succeeding window must be smaller than or equal to those days in the current week. Because we minimize in the objective function, the capacity in the appended windows will be equal to the capacity in the week we plan for, similar to the process described in Chapter 2. In the objective function, we incur cost for shifts with their starting timeslot in range $[P, P + W]$ so that the appended timeslots are excluded in cost evaluation.

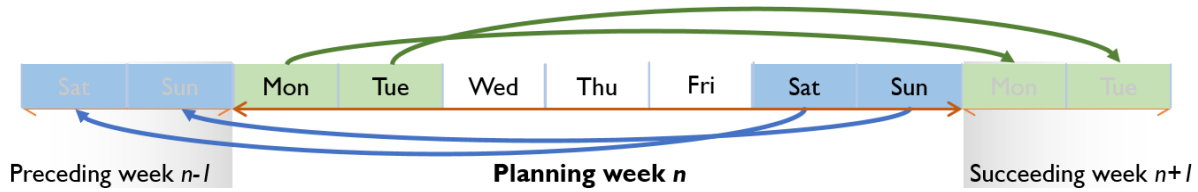


Figure 36 – Illustration of the extension of the planning window

Constraint (15) introduces the timeslot mitigation constraint for a dedicated time interval. We incur fixed cost in the objective function if any capacity is allocated in the window $\forall t \in [L, U]$. Variable f_t indicates whether any work is performed in timeslots $t \in [L, U]$. Parameter F in the objective function controls the cost of additional employment of timeslots in window $[L, U]$.

Relaxing precedence constraints

Following the flow relations introduced in Chapter 2, we argue that precedence constraining variables y_{jt} must be binary. The reason for this is that work at processes is not allowed to start before their predecessors have finished processing the preceding work package entirely. When we enlarge the planning resolution, (virtual) offsets between processes increase as a function of the resolution increase.

We can relax the constraints to $y_{jt} \in [0,1]$, creating a slight adaptation of the binary precedence flow. The continuous representation of y_{jt} incurs that work performed at successors must be greater than work performed at predecessors in any timeslot. Contrary to the binary constraint, the succeeding process no longer has to prolong starting until its predecessor has fully finished processing a work package, the constraints only indicate that the fraction of work performed for an order must be greater than for its successor.

We expect the relaxation decision to reduce runtime as it reduces the complexity of the program, at a minimum deterioration of feasibility as the majority of the scheduling functions flow relations are adhered to by relaxing. We test the effect of this relaxation decision on performance and feasibility in Appendix B. The relaxation constraints are shown to retain validity while providing better solutions in less time (Appendix B).

The values for $\alpha = 6$ (scheduling bucket) and $\beta = 2$ (planning bucket) are found through experimentation. Appendix B shows that this combination of the parameters yields the lowest cost capacity plans.

We model the problem using PuLP (Mitchel, O'Sullivan, & Dunning, 2011). PuLP is a linear programming toolkit for Python. Several solvers are available for ILP models. We distinguish commercial- and open-source solvers. We attempt to solve problem instances using CPLEX 12.9.0 and CBC 2.9.6. CBC (Coin-or branch-and-cut) is an open-source mixed-integer program solver that uses branch-and-cut methods (Forrest, 2019). CBC is among the most performant non-commercial solvers available today (Mittelman, 2019).

6.4. Conclusion

This chapter approaches the problem using the integrated strategy. With this strategy, we attempt to optimize the capacity plan and schedule orders over workstations at the same time. We model the problem as an integer linear program.

Given the size of the original combinatorial problem with 10-minute timeslots, we modify the input parameters so that we can solve for performant capacity plans. To do this, we introduce scheduling and planning buckets while relaxing some of the precedence constraints.

7. Performance

In this chapter we consider research question 5: How do the alternative strategies perform? Chapter 4 proposes two solution strategies to approach any problem instance. The problem instances constitute weekly store orders for the mechanised warehouse. This chapter compares the solution strategies for a set of problem instances.

To test performance, we define test instances in Section 7.2. We test the effect of input variables on solution cost. Section 7.3 covers these tests.

The problem instances we consider are real-life store order sets for the mechanised DC. We evaluate capacity plans using the weighed timeslot cost function described in Chapter 4. We only consider capacity plans that fulfil the constraints described in Chapter 2, validated in the TOP module using the procedure in Section 7.1.

7.1. Validation

We introduce two strategies to planning capacity. While they model the constraints to planning using (a simulation of) the priority rule in the scheduling function, the resulting capacity plans must be validated. The integrated strategy does not fully mirror the scheduling function in the distribution centre planning application. Therefore, we must validate if the scheduling function behaviour we model is valid.

Figure 37 shows the validation process. To validate, we load the explicit variables and parameters for planning (transport plan, shift parameters and system parameters) from the planning module. Then, we plan capacity automatically outside the planning module environment using either the integrated or the iterative strategy. We export the system resource levels (workstation allocations per timeslot) to the planning module and validate using the real scheduling function.

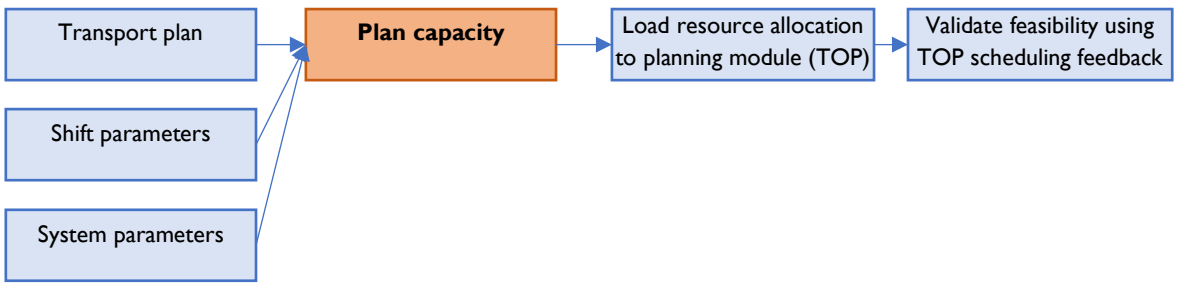


Figure 37 – Validation mechanism of the resource plan

If the scheduling function in the planning module reports warnings or errors normally used as feedback for manual planning, we know that our planning model is invalid in practice, as we modelled the problems to not violate constraints. All resulting capacity plans from the iterative as well as the integrated planning strategy were loaded into the planning module and found to report no errors, so all created plans are feasible. However, we cannot fully guarantee that created capacity plans are valid for any instance provided to the planning module.

7.2. Test instances

In order to test the performance of solution strategies, we consider multiple transport plan test instances. Figure 38 shows the minimum, mean, and maximum daily demand for 5 weekly transport plan test instances. These transport plans differ in their demand patterns. For instance, some transport plans show peak customer store demand in the weekend while others show little variance throughout the week. These factors may impact the performance of the solution strategies.

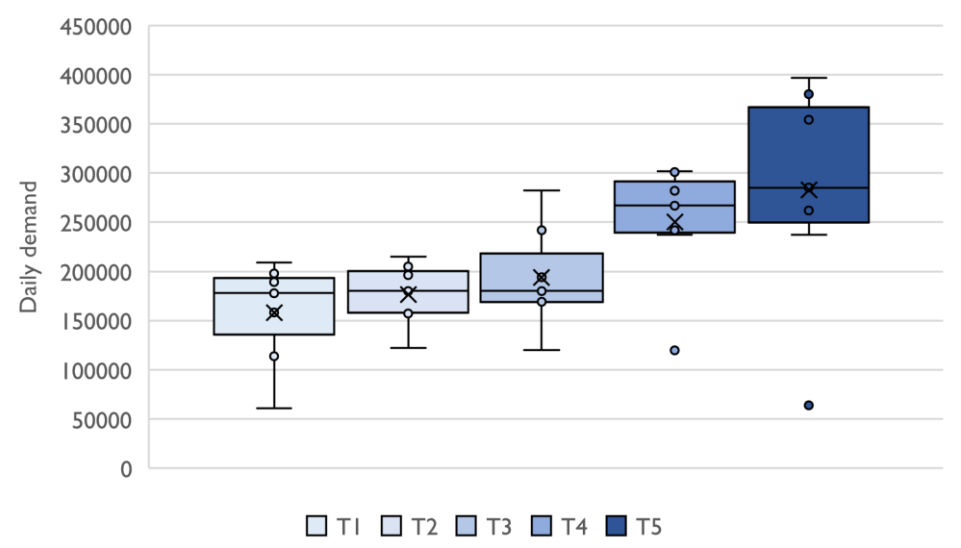


Figure 38 – Box plot of daily demand for 5 weekly transport plan test instances (inclusive median)

We create additional fictional test instances by multiplying demand for these test instances with factors 0.80, 1.20 as to test for a greater range of instances. Note that only test instances T1 and T2 include demand for the RRP area. Also, the permitted lead times are equal for all orders except for T2, which includes variance in the order lead time.

7.3. Experimentation

Capacity plans for the integrated and iterative strategy are obtained using input parameters other than the transport plan, model files and shift configuration. The input parameters reduce the complexity of the planning problem. We test various ranges of these values to determine their effect on the output cost function.

We define the scheduling bucket for the integrated strategy as the finest level of time we plan in. Normally, this is one timeslot (10 minutes). However, we can plan production and allocations in a coarser granularity therefore decreasing computational complexity. For the tests, we use a scheduling bucket of $\alpha = 6$ and a planning bucket of $\beta = 2$ as found to be optimal for our test instance (Appendix B).

Chapter 3 refers to differences in typical time convergence of exact versus approximate methods. To test this discrepancy, we consider the best solution cost value obtained after a time limit (maximum runtime). We consider maximum runtimes of $t \in \{30,60,300,600\}$.

Furthermore, we test the deviation over created workstation classes of both strategies over the same input transport plans. This way, we test stochasticity of both strategies. We extend the test instance set by multiplying store order demand with factors 0.8 and 1.2. We test if the performance differs for variations of demand. Lastly, we compare the performance of the strategies to a manually created plan by comparing several performance indicators over a single test instance.

7.3.1. Results

Both the integrated and iterative strategy use random numbers. The iterative strategy uses random numbers to calculate probabilities and the integrated strategy in CPLEX uses random numbers in some if its internal operations. For example, it has an influence on probing and on branching. Therefore, we require multiple runs for each instance to derive a statistical basis of the cost evaluation function and make conclusions on performance.

Runtime performance

A factor to consider in the performance of the strategies is time. A strategy that converges more quickly is more favourable in practice. Table 11 shows the performance over time of the integrated and iterative strategy with time limits of 30, 60, 300 and 600 seconds. We evaluate these runs using the cost function.

Initially, we provide a stopping temperature as a criterion for the simulated annealing. However, wedo not set a stopping temperature and use the best solution found after t seconds so that we can compare the methods somewhat fairly.

Generally, the integrated strategy outperforms the iterative strategy given sufficient time. We note that both strategies converge quickly. However, sometimes the integrated strategy fails to consider reasonable solutions initially (most often up to a minute) whereas the iterative strategy is generally quicker to converge. We presume that the reason for this is that the iterative strategy can easily remove shifts especially early on as many removals are feasible. On the other hand, the integrated approach includes the scheduling function orders as variables, which presumably takes more time to find a feasible configuration of both the planning and scheduling variables. We note that the iterative strategy performs better for the test instance with lead time variance (instance 2). In practice, input order set will have lead time variance, so the iterative strategy may be most performant.

	T=30		T = 60		T= 300		T = 600		
	Integrated	Iterative	Integrated	Iterative	Integrated	Iterative	Integrated	Iterative	*Integrated LP (integrality gap)*
1	8949	10848	8949	9116	8568	8602	8521	8512	*8400 (1.42%)*
2	33445	11876	11314	9788	9740	9654	9417	9238	*9192 (2.38%)*
3	31913	14506	10496	11720	10204	10309	10186	10287	*9827 (3.52%)*
4	43332	14258	11907	12672	11204	11423	10976	11242	*10653 (2.94)*
5	43332	19238	38240	16924	15220	15846	15143	15548	*14666 (3.15)*
%	123%		25%		-1%		-1%		

Table 11 – Performance over time of the integrated and iterative strategies

Table 11 also shows the relaxed LP solution for the integrated strategy as well as the integrality gap. Note that the LP acts as a lower bound on the solution value for the integrated strategy. The true optimal value for the integrated strategy is somewhere between the lower bound and the best

solution value obtained at 600 seconds. Note however that this lower bound is obtained after the aggregation decisions proposed in Chapter 6. The true lower bound may be slightly lower as the granularity becomes finer.

Performance over deviating demand

Table 12 shows the mean solution cost for the two solution strategies: iterative and integrated. The integrated strategy outperforms the iterative strategy when lead times are high and constant, and when demand is generally high. On the other hand, the iterative strategy outperforms the integrated strategy for plans with low demand and small or variant lead time windows (instance 2).

The scheduling bucket of 6 timeslots (1hour) leads to a loss in the scheduling part of the integrated strategy. Variables x_{jt} can be planned in multiples of the planning resolution. This creates larger time constraints on the solution space, thus increasing solution cost. The effect of this temporal aggregation and constraining decision seems to be more apparent as lead time windows and demand are smaller. On the other hand, as weekly demand increases, the performance of the integrated strategy improves.

	Demand factor = 0.8		Demand factor = 1		Demand factor = 1.2	
	Integrated	Iterative	Integrated	Iterative	Integrated	Iterative
1	6995	7212	8521	8512	10684	10724
2	7645	7566	9417	9238	11812	11943
3	7340	7603	10186	10287	15156	15569
4	7724	8004	10976	11242	16076	16412
5	9205	9667	15005	15548	INF	INF

Table 12 – Mean solution cost comparison of strategies and instances with planning bucket = 2 and max runtime = 600, with n=5 runs for each instance and strategy

7.3.2. Comparison to manually created plans

Whilst we show that we can obtain feasible capacity plans, we do not know how their performance relates to current (manual) allocations. We have a single reference plan for test instance 4. We use the performance indicators of this plan as a proxy for other test instances.

	Manual	**Automatic**	
		Integrated	Iterative
Operator timeslot cost (weighed hourly cost)	13208	10976 (-16.9%)	11242 (-14.9%)
Operator hours (incl indirect labor)	3059	2694 -(11.9%)	2724 (-10.9%)
Mean productivity (cases/operator hour)	572	649 (13.5%)	642 (12.2%)

Table 13 – Comparison of manual and automatic planning cost and utilization.

Table 13 shows a performance comparison for the manually created plan as well as the automatically created plans using the integrated and iterative strategy for a single test instance. We compare the main performance indicator: weighed operator cost. The integrated and iterative strategy respectively incur 20,6% and -18,6% less cost than the manual plan on average.

The operator hours performance indicator shows the number of operators found when importing the allocations into the planning module. This amount of operator hours includes indirect personnel hours. We observe a smaller percentual decline. We find two main reasons. The first is that we do not include indirect tasks in the weighed operator cost calculation, as this personnel is not explicitly planned for. Another explanation is that we optimize for weighed operator cost, rather than the unweighted number of operator hours.

7.4. Conclusion

This chapter finds that the capacity plans created by both the iterative and integrated strategy remain valid when tested in the planning module. Moreover, their results seem promising as the integrality gap for the integrated strategy is in the order of several percentages, indicating that the plans can most likely not be improved beyond those few percentages.

The integrated strategy generally outperforms the iterative strategy. However, the iterative strategy seems to perform well on the test instance that contains varying lead times. Practically, orders will contain varying lead times, so the iterative strategy may perform well on those test cases. On the other hand, we see that as the demand on the system increases, the outperformance of the integrated strategy increases as well.

To conclude, both the iterative and the integrated strategy outperform the manually created plan (for a single test case) for the Albert Heijn distribution centre.

8. Discussion & Conclusion

This chapter discusses the main research question: *How can effective capacity plans for the distribution centre be made in reasonable time in the tactical planning phase?*

Section 8.1 provides an overview of the benefits and drawbacks of both the integrated and iterative strategy. Section 8.2 discusses generalizability and applications to other distribution centres in more detail. Section 8.3 provides recommendations for further research.

8.1. Comparison integrated and iterative strategy

With a limited sample size, we find that the integrated and iterative strategy outperform manually created plans. Generally, their performance is similar, with the integrated strategy showing a slightly higher mean productivity and lower mean cost for most problems. We compare other aspects of both strategies. Table 14 provides an overview of arguments made for and against each strategy in previous chapters.

	Integrated	**Iterative**
Solution quality	~ For most test instances, the integrated strategy outperforms the iterative strategy	~ The integrated strategy outperforms the iterative strategy by ~2% given sufficient time
Solution convergence	- Generally takes more time to find (any) initial feasible solution	+ Instances converge quickly, mainly attributable to constructive heuristic
Solution deviation	+ Multiple runs for the same test instance show less deviation for the iterative strategy	- Multiple runs for the same test instance show greater deviation
Feasibility probability	- With the introduction of the planning resolution parameter α, we limit the solution space thus increasing the probability of not obtaining feasible solutions	+ Greater feasibility probability: We use normal 10 minute timeslot length, representing the full problem
Validity	- Timeslots and the scheduling algorithm are aggregated, while all instances we provide are valid, we cannot assume absolute validity	+ Mimics scheduling algorithm, always valid
Buffer constraints	- Explicitly modelling buffers is difficult	+ Buffer constraints can be modelled explicitly
Modeling breaks	- Cannot explicitly model breaks, must adapt by lowering productivity	+ Operator breaks can be modelled explicitly
Shift time requirements	- Can use any configuration of shifts, however they must start in increments of the planning resolution.	- Can only use constructive heuristic when adjacent shifts spanning full day are available
Additional requirements	ILP solver, e.g.CBC or CPLEX	

Table 14 – Comparison between integrated and iterative strategy

8.2. Applicability to other distribution centres

While this research considers the Albert Heijn distribution centre in Zaandam, the solutions proposed to our problem may be extended to other distribution centres. We consider several factors that influence the applicability of our methods:

- Predictability of orders and processing rates – We consider all demand and supply rates as deterministic. This way, we focus on maximizing utilisation. However, If we were to include for instance stochasticity in demand, we would likely not focus on maximizing utilisation and lowering cost, but also include risks such downtime and queues that disturb productivity.
- Priority rule / quick scheduling evaluation (in this case JIT) – In our case, the scheduling algorithm acts as a means of feedback for the capacity plan. We can iterate relatively quickly to the trivial nature of the scheduling problem as the scheduling rule is simple (JIT). Other, more complex scheduling methods may complicate the problem.
- Clear input parameters (e.g. shifts definition, processing rates) – All input parameters to the problem are clearly defined for this distribution centre. For instance, the length of shifts as well as the offsets and processing rates are available as detailed knowledge as a result of prior simulation studies. For other distribution centre, this may hinder the rate of implementation as these parameters have to be determined first.

8.3. Recommendations for further research

With a limited sample size, we find that both strategies significantly outperform the manual planning procedure. Both strategies can provide solutions within a limited timeframe (10 minutes or less) for all instances tested. These automatically created plans could be used as a starting point or as a frame of reference for planners in order to save time and operator hours.

We recommend Vanderlande to use the iterative strategy for capacity planning in the planning module. While the integrated strategy generally outperforms the iterative strategy in the weighed number of operator hours required, several arguments advocate implementing the iterative strategy.

First of all, we find a mean cost performance difference between the integrated and iterative strategy of less than 2% in weighed operator cost, while we find a difference of 19% compared to manually created plans. The difference between the integrated and iterative strategy is far smaller.

Furthermore, we note that the iterative strategy is more likely to obtain feasible solutions.

Lastly, we find that commercial solvers (i.e. CPLEX) provide feasible solutions within reasonable time. Open-source solvers such as CBC require a significant additional time investment rendering them inferior to the iterative strategy as well. Opting for a commercial solver requires significant additional monetary investment compared to using the iterative strategy.

A sidenote to using the iterative strategy is that a basis of adjacent shifts is required in the constructive phase.

The current planning process is deterministic and does not consider robustness. (Forecasts of) demand is considered as deterministic input in the planning module. As we plan for high mean productivity and low operator cost, utilization approaches 1. As utilization increases, robustness towards uncertainty decreases. In other words, as we plan more efficiently, there is less free capacity to buffer

for unexpected positive demand increments. Therefore, further research could be aimed at creating robust plans. However this requires insights in the stochasticity of both supply and demand at the distribution centre.

As we described in Chapter 2 there are indirect tasks in the distribution centre. These include for instance supplying empty load carriers and defoiling pallets. These tasks are not planned for by manual planners as they are linked directly to processes.

This thesis considers simulated annealing as an improvement heuristic. There are numerous other metaheuristics such as tabu search (Glover & Laguna, 1998) and genetic algorithms (Gen & Lin, 2007) that could perhaps be applied to our problem

While we provide an argument against the integrated strategy being the licensing cost for commercial solvers, there may be significant improvement potential for open-source solvers such as CBC through the tuning of parameters (Baz & Hunsaker, 2007).

Similarly, we consider some experimentation to determine the simulated annealing and constructive heuristic parameters. We could include a more dynamic cooling scheme for simulated annealing, perhaps based on statistical analysis (Aarts & Korst, 2005). Furthermore, the constructive adaptive search procedure could be improved by altering the bias factor over time, i.e. starting deterministically and ending with more stochastic shift choices.

8.4. Conclusion

First, we wanted to know how capacity plans are made currently, and how their performance is assessed. We found that many parameters and variables are used as discrete input into the current manual planning function. Planners use a scheduling priority function to assess the performance of their manually created plans. In practice, production planners aim for maximum productivity while meeting all departure times and not violating any constraints.

We find that planning for a week's demand is complex as the time horizon is highly constrained and partitioned into many timeslots. Literature describes many combinatorial planning and scheduling problems, of which some exhibit properties similar to our problem. One of the most similar combinatorial planning problems is the time-driven rough-cut capacity planning problem. However, it differs in several aspects. The key discrepancy is in the hiring of non-regular capacity. In our problem, non-regular capacity can only be hired in fixed time intervals, and is available over multiple processes during that time. This further complicates the problem.

We explore two approaches to the capacity planning problem: an integrated strategy combining the planning problem and scheduling function and an iterative strategy that communicates between these phases.

The integrated strategy requires that all parameters, variables and constraints are modelled so that we can solve combinations of variable values that are feasible in practice following the scheduling function. As the problem is quite complex, we find that we cannot solve for all constraints and variables explicitly in a combinatorial integrated approach, as the time required for finding reasonable solutions expands quickly. We propose several methods to simplify the problem modelled for the integrated strategy, while retaining feasibility.

Furthermore, we propose an iterative strategy that automates a process similar to the current planning methodology. The iterative strategy starts with a maximum capacity allocation and continues to remove shifts if those decision are feasible. Then an improvement phase starts. The improvement phase focuses on using overlapping shifts and neglecting expensive timeslots as well as swapping humans over processes during their shifts, all while retaining feasibility following the scheduling priority function.

Both the iterative and integrated strategy show promising results when comparing to solution cost of obtain capacity plans to manually created plans for the Albert Heijn distribution centre. We show that within an order of minutes, we can create plans that use few human operators.

However, we note that especially the integrated strategy is somewhat tailored to our problem. For instance, if the shifts parameters were to change, then the integrated strategy would have to be adapted. Nonetheless, the iterative strategy provides a starting point that may be adapted easily to other distribution centres that use a similar means of iteration between planning and scheduling. However, the availability of accurate input parameters remains vital to the success of the algorithm.

9. References

Aarts, E., & Korst, J. M. (2005). Simulated Annealing. In *Search methodologies* (pp. 187-210). Boston, MA: Springer.

Baz, M., & Hunsaker, B. (2007). *Automated tuning of optimization software parameters.* University of Pittsburgh, Department of Industrial Engineering.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268-308.

Boschetti, M., Maniezzo, V. R., & Roffilli, M. (2009). Matheuristics: Optimization, simulation and control. In *International Workshop on Hybrid Metaheuristics.* Berlin: Springer.

Caceres-Cruz, J., & Arias, P. (2015). Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*.

De Boer, R. (1998). *Resource-constrained multi-project management (Doctoral dissertation, PhD thesis, University of Twente, The Netherlands)*.

Festa, P., & Resende, M. G. (1995). *Greedy randomized adaptive search procedure.* Springer.

Fischetti, M., & Fischetti, M. (2016). Matheuristics. In *Handbook of Heuristics* (pp. 1-33). Springer.

Forrest, J. (2019). *Cbc (coin-or branch and cut) open-source mixed integer programming solver*. Retrieved from https://projects.coin-or.org/Cbc

Gademann, N., & Schutten, M. (2001). Linear-programming-based heuristics for project capacity planning. *IIE Transactions*, 37,153-156.

Gen, M., & Lin, L. (2007). Genetic Algorithms. In *Wiley Encyclopedia of Computer Science and Engineering.*

Glover, F., & Kochenberger, G. (2003). *Handbook of metaheuristics.* Springer.

Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093-229). Boston, MA: Springer.

Hans, E. (2001). *Resource loading by branch-and-price techniques.*

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing. *science*, 671-680.

Kis, T. (2005). A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. *Mathematical Programming*, 103(3), 515-539.

Kis, T. (2006). Rccps with variable intensity activities and feeding precedence constraints. *Perspectives in modern project scheduling*, 105-129.

Kolisch, R., & Drexl, A. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics*, 23-40.

Laguna, M. (2018). Tabu search. In *Handbook of heuristics.*

Law, A., & Kelton, W. (2000). *Simulation modeling and analysis.* New York: McGraw-Hill.

Lawler, E. (1976). *Combinatorial Optimization: Networks nd Matroids.* New York: Holt, Rineh art and Winston.

Leus, R. (2003). The generation of stable project plans, Complexity and Exact Algorithms. *PhD thesis*. Belgium: Katholieke Universiteit Leuven.

López-Ibáñez, M., Stützle, T., & Dorigo, M. (2016). Ant Colony Optimization: A Component-Wise Overview. In *Handbook of heuristics* (pp. 371-407). Springer.

Maravelias, C., & Sung, C. (2009). Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers & Chemical Engineering*, 1919-1930.

Masmoudi, M. (2011). Tactical and operational project planning under uncertainties: application to helicopter maintenance. *Doctorial dissertation, Ecole nationale superieure de l'aeronautique et de l'escape*.

Mitchel, S., O'Sullivan, M., & Dunning, I. (2011). *PuLP: a linear programming toolkit for python.* Auckland, New Zealand: The University of Auckland.

Mitchell, J. (2002). Branch-and-cut algorithms for combinatorial optimization problems. In *Handbook of applied optimization* (pp. 65-77).

Mittelman, H. (2019). *Decision Tree for Optimization Software*. Retrieved from http://plato.asu.edu/guide.html

Möhring, R. (1984). Minimizing costs of resource requirements inproject networks subject to a fixed completion time. *Operations Research, 32(1)*, 89-120.

Muthuraman, S., & Venkatesan, V. (2017). A comprehensive study on hybrid meta-heuristic approaches used for solving combinatorial optimization problems. *2017 World Congress on Computing and Communication Technologies (WCCCT)*.

Neumann, K., Schwindt, C., & Zimmermann, J. (2002). Project scheduling with time windows and scarce resource. In *Lecture notes in economics and mathematical systems (Vol. 508).* Berlin: Springer.

Osman, I., & Kelly, J. (1996). Meta-heuristics: an overview. In *Meta-heuristics* (pp. 1-21). Boston: Springer.

Sevaux, M., Sörensen, K., & Pillay, N. (2018). Adaptive and Multilevel Metaheuristics. In *Handbook of heuristics* (pp. 3-20). Springer.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation.* John Wiley & Sons, 2009.

Thomas, N., & Ruiz, R. (2018). Iterated Greedy. In *Handbook of heuristics* (pp. 547-570). Springer.

Vanderlande. (2018). *Vanderlande Internal Document, TOP SRS.*

Appendix A: Processing parameters for ILP

Both the number of timeslots and jobs (in our case: production area orders) are large multiples of extremes found in literature for RCCP problems. As the RCCP problem is NP-hard, it may be cumbersome to solve for a solution space this large. Therefore we could decide to allow for some aggregation in the length- and planning of timeslots. E.g. if we decide to plan for a timeslot planning resolution of (multiple)hours rather than 10 minutes, we reduce the number of decision variables greatly, and can expect to obtain allocation results with less computational effort.

However, a temporal aggregation decision is likely to come at the cost of accuracy with respect to the scheduling algorithm. If these timeslots are aggregated, then the order timeslots must be aggregated as well, while preserving the condition that orders are finished before their deadline.

These deadline attributes are usually specified at a level of granularity finer than the aggregated planning resolution. To continue finding feasible solutions constrained by e.g. order lead times, we must set upper bounds smaller than true values as we are limited by the level of granularity. E.g. if the maximum lead time for an order is 07:30 (hh:mm) and we decide to use a resolution of 2 hours rather than 10 minutes, we can only state that the maximum lead time is 06:00 as we could otherwise violate hard lead time constraints.

These aggregation decisions impact the solution space and the ability to find feasible solutions. In case the solution space is highly constrained, for instance when the number of required workstations is close to its maximum for several timeslots, we may never obtain feasible solutions when deciding to aggregating as time windows are further constrained. By lowering the planning granularity, we expect a smaller probability of attaining feasible solutions as the solution space for any job is constrained increasingly with coarser granularity.

Introducing a shift resolution for resource planning

Human resources can swap over processes during their shift. For instance, they can work at a semi-automatic depalletizing workstation for the first 5 hours of their shift and at a semi-automatic picking workstation for the last 3 hours. In combinatorial optimization, solver seek the best combination of allocations for a human resource. In practice this may imply that a worker is planned to change process after every planning timeslot. If we plan in timeslots of 10 minutes, this is unrealistic and creates a large decision space. Changing every 10 minutes is unrealistic because there are significant changeover and setup times.

Following the above arguments, there is good reason to reduce the number of times human resources can change over processes during their shift. Therefore, we introduce shift resolution parameter β to bound the number of changes to a maximum of β per shift.

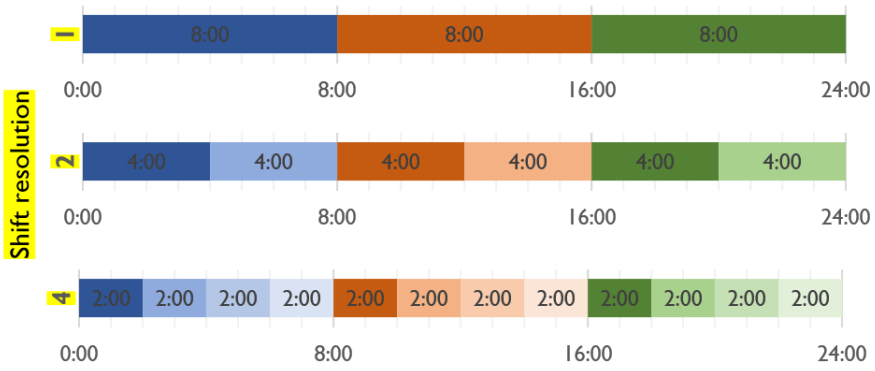


Figure 39 – Illustration of shift resolution

Figure 39 illustrates the introduction of the shift resolution parameter. As the shift resolution parameter β increases, the number of allowed changes per shift increases.

Aggregating store order demand over time

Manual planning is done for timeslots of 10 minutes. For instance, planners can specify that a workstation is available from 00:00 to 00:10 but not from 00:10 to 00:20. The scheduling algorithm uses the same timeslot size. We previously determined that a combinatorial problem with many timeslots requires increasingly many computational steps for solving. For this reason, we suggest aggregating demand and capacity over time.

To do this, we introduce a planning resolution parameter α. Previously, we stated that demand and capacity planning is currently done for windows of 10 minutes, thus planning for intervals of length $[t, t+1)$, where 1 is a timeslot of 10 minutes. We introduce a planning resolution parameter α, where we plan for intervals of the size of α timeslots. This creates new timeslot for intervals of length $[t, t+\alpha)$.

In practice, we know that there are only three processing paths. These are categorized by orders with demand for areas ACP, NC and RRP. As we introduced previously, a store order may specify for instance 14 load carriers from the ACP area and 12 from the NC area. The load carrier processing unit is the finest level of granularity provided in a store order. For that reason, all orders with demand for the ACP area must follow the exact same processing steps, and the rate of producing load carriers is equal for all orders in an area provided they are handled by the same workstation class. Therefore, if two orders constitute the exact same deadline timeslot and cut-off time, they could be aggregated in their area demand.

For this reason, we can aggregate orders order demand for common timeslot deadlines, restricting the solution space. When planning for a greater time resolution (i.e. a multiple of timeslots), we can aggregate all area demand that is due in the window $[t, t+\alpha)$. To satisfy constraints, we must set the cut-off time in that time window as the minimum of all cut-off times of store orders with a deadline in the time window $[t, t+\alpha)$. By setting this minimum cut-off time, we ensure that we never violate any of the cut-off time constraints.

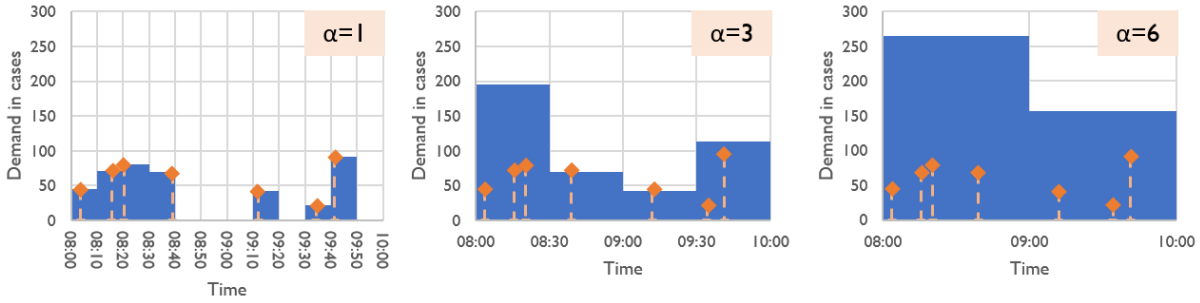


Figure 40 – Illustration of demand aggregation, orders with a departure time in timeslots $[t, t+\alpha)$ are aggregated.

Figure 40 illustrates demand aggregation for orders with a common demand area. In this example there are seven store orders in this example each with a truck departure time in the interval [08:00, 10:00]. In the first scenario, they are aggregated with $\alpha = 1$, or in intervals of 10 minutes. We obtain 7 different parameters with demand > 0. In the second scenario, they are aggregated in intervals of 30 minutes with $\alpha = 3$. The last scenario shows $\alpha = 6$. Thus, the level of aggregation depends on the planning resolution. With an original planning horizon of t timeslots and a resolution of α, we obtain t/α timeslots in aggregation. With this decision, we aim to reduce the problem size while maintaining feasibility by taking the minimum of the cut-off times of the aggregate elements.

Effect of temporal aggregation on offset accuracy

The aggregation of timeslots has impact on the feasibility of processing offsets. In Chapter 2 we determined that the scheduling function uses fixed offsets to schedule demand. These offsets are multiples of 10 minutes and may not necessarily correspond to the aggregated time parameters introduced in the previous paragraph.

By aggregating demand and supply over time, we lose planning accuracy and in some cases find infeasible solutions to allocations that could be feasible in reality. We provide an example. If we plan with $\alpha = 6$ in 1 hour timeslots rather than 10 minutes the available working window for each process

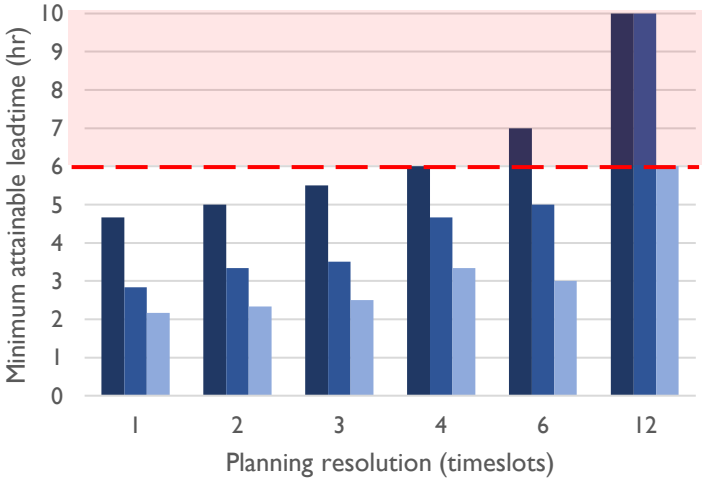


Figure 42 – Minimum attainable lead time for planning resolution

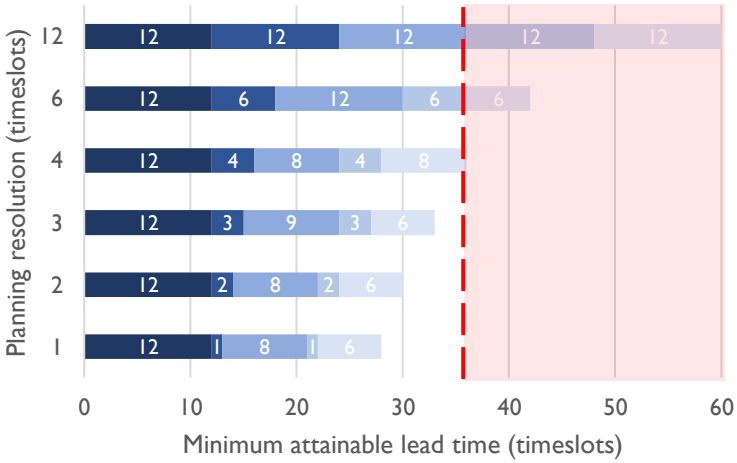


Figure 41 – Planning resolution in the ACP area

decreases. All offsets and processing windows must be rounded up to a multiple of the finest level of granularity α timeslots to maintain feasibility and satisfy constraints. If they are not rounded up, we cannot explicitly model these steps as they might induce infeasible capacity plans. Thus, we lose a planning time window of the amount rounded up for each of these steps. The time lost is equal to $((\alpha - \gamma \ (mod\ \alpha))(mod\ \alpha)$ for each processing step with timeslot length γ and planning resolution α. The solution space decreases as the working window sizes decrease as a function of the planning resolution. Therefore the likelihood of finding feasible allocations for transport plans decreases with a coarser planning resolution parameter α.

Figure 42 shows the minimum attainable lead time for aggregate planning resolutions. The minimum attainable lead time is determined using the offsets and processing times introduced in Chapter 2. We round all offsets and minimum processing time up to the planning resolution to determine the minimum attainable lead time per production area. Figure 43 shows a typical weekly cut-off time distribution. A minimum and maximum cut-off time are promised to the warehouse customer. If we seek to satisfy these minimum cut off lead times, then the minimum feasible lead times in Figure 42 must subceed the minimum cut-off time for any production area in an order. For this reason we draw a bounding line at the minimum attainable lead time equal to 6 hours, as it represents the minimum cut-off time allowed by customer stores in their orders.

Figure 41 shows the composition of minimum attainable lead time for the ACP area for different planning resolutions. The figure shows that with an increasing planning resolution, more time must be reserved for each process to comply with constraints and offsets. The amount of time we reserve for each step is equal to the nearest upper multiple of the planning resolution α. When that finest level of granularity is greater than the time required for the step, the minimum attainable lead time increases. When many processing steps exist in an area, the planning process is constrained further. If the minimum cut off time is 6 hours (Figure 43), we must plan with a resolution smaller than 4 timeslots as the sum of time required for processing steps and offsets (i.e. the minimum attainable lead time) in the ACP area is greater than 6 hours for a planning resolution greater than 4 timeslots highlighted in Figure 41. Note that if we do not include the timeslots required for processing (10 minutes) directly, then we can plan with a resolution of $\alpha = 6$.

The processing and offset times must be rounded up to the nearest multiple of the temporal planning resolution. The reason for this is that we require an upper bound on the total lead time to be created in order to mitigate lead time constraint violation.

Breaks

Chapter 2 introduces breaks in the shifts. We note that the ratio of work to breaks is similar among shifts. Each full-time shift incurs 0:50 of breaks for 7:10 of work. As we increase the planning resolution to values greater than 10 minute timeslots, we can no longer explicitly model all these shift breaks. Alternatively

we propose to decrease mean timeslot productivity. We find the new timeslot productivity as: $TSN_k = \frac{43}{48} TSN_k \ \forall k$ by following the same productivity ratio.

Constraining working windows

The transport plan with outbound store orders serves as input for determining an optimal capacity plan. We must make some adaptations to the transport plan to create parameters for feasible ILP formulations. First of all, we introduced parameter R_j as the earliest start timeslot of a work package. The value for this parameter is found by $R_j = D_j - \min(\max(C_j, L), U)$. Parameter D_j is the deadline of a store order j. C_j refers to the cut-off offset. The interval $[L, U]$ denotes the lower and upper bounds of the allowed lead-times respectively. Orders with $C_j < L$ are considered infeasible, so we transform them to the minimum attainable lead time. On the other hand, lead times cannot be greater than U.

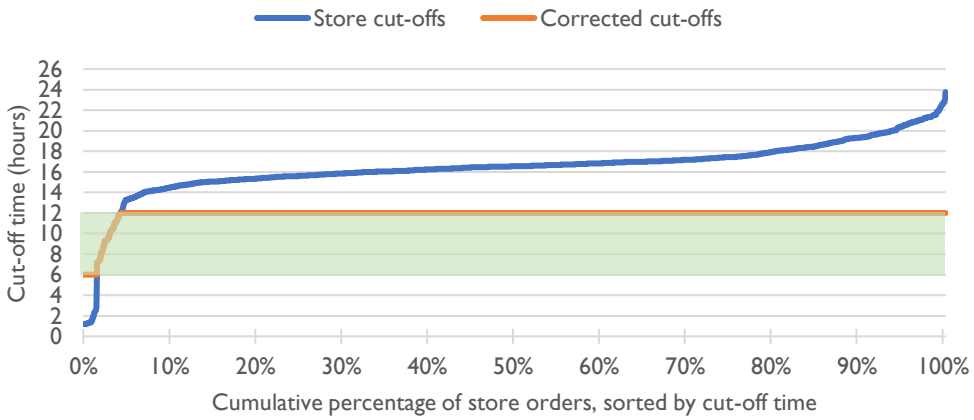


Figure 43 – Cutoff time distribution in a sample order set, correcting for lower- and upper bound values

Figure 43 shows store orders in a sample transport plan sorted by their cut-off times (blue). Many store order cut-off times lie outside the allowed lead time window of $[L, U]$ and therefore require correction. They are transformed to the values highlighted in Figure 43 following the formula for R_j. By constraining the cut-off times this way, we create working windows $[R_j, D_j]$ that always satisfy the maximum leadtime constraint introduced in Section 2.6.1. We introduce the window $[R_j, D_j]$ as constraints for processing store order j. If all work x_j cannot be performed in this window with available capacity, then the constraints cannot be satisfied, thus creating an infeasible solution.

Accuracy and limitations

The ILP formulation of the integrated strategy neglects the constraints on intermediate buffers levels introduced in Chapter 2. Explicitly adding constraints for the intermediate buffers levels between processes for each timeslot requires the introduction of many constraints and variables. This modification is expected to significantly add to required computation time as the program size increases. We cannot model empty buffers in practice as it requires the use of logic unsuitable for linear programs. However, we can add some simplified constraints for buffers that require less variables, as seen in constraint (b1), (b2) where b_{kt} is the variable buffer content in processing units and B_k is the maximum buffer capacity.

$$\sum_j Q_j x_{jt} + b_{kt} \geq \sum_i Q_i x_{it} \qquad \forall k, t, j \;\; if \; K_j = k \,, i \in P_j \quad (b1)$$

$$\sum_{\tau=\max\left(t-\frac{W}{7},0\right)}^{t} b_{k\tau} \leq B_k \quad \forall k,t \ \ (b2)$$

$$b_{kt} \in \mathbb{R}$$

However, we observe that buffer restrictions in their maximum capacity hardly ever constrain the solution space. Lead time constraints introduced by maximum cut-off times bind buffer levels sufficiently so that buffers are unlikely to fill to maximum capacity in practice.

Appendix B: Strategy parameters

In this chapter we cover the parameters for both the integrated and iterative solution strategies. We determine optimal parameters for both these strategies that are used to compare results for different transport plan dependent variables.

For the integrated strategy, we consider including binary or relaxed precedence relations, the inclusion of buffer constraints and

Integrated strategy

In Chapter 5 we introduced the option of relaxing precedence constraints. We test if this decision creates feasible plans, and test its effect on performance over time. With a planning bucket (beta) of 2 and a scheduling bucket (alpha) of 6 we test for 3 instances.

	Mean, median (min, max)	
Instance	Binary	Relaxed
2	**13404**, 11780 (9672, 26393)	**9417**, 9342 (9205, 9912)
4	**13784**, 12993 (12402, 22087)	**10976**, 10969 (10892, 11146)
5	**16002**, 15302 (15102, 19783)	**15005**, 15212 (14942, 15392)

We observe that all allocations created with relaxed constraints are feasible. Moreover, they most often provide the most cost-effective resource plans compared to the binary constraints case.

Table 15 shows the mean solution cost for test instances 2,4,5 with different values for the alpha (scheduling bucket) and beta (planning bucket) parameters. Observe that for small values of alpha and beta, the solver fails to converge. Presumably, this is because the solution space becomes excessive.

		Scheduling bucket (alpha)				
		1	2	3	4	6
Planning bucket (Beta)	1	none	none	23854	17287	14802
	2	none	none	18366	13905	11799
	4	none	none	none	24825	21079
	8	none	none	none	28846	22758

Table 15 – Mean solution cost for test instances {2,4,5} with different values for alpha and beta

When using a planning bucket of 1, the part time overlapping shifts are not considered. Therefore, the solution space is limited and cost remains relatively high. We note that using planning buckets greater than 2 have little benefits as they seemingly only add diversification at the cost of intensification of the solution space.

Iterative strategy

The approximate strategy considers a constructive and improvement heuristic. For the adaptive search constructive procedure, we consider the bias factor and number of shifts to remove as parameters. For the simulated annealing improvement heuristic, we consider the starting temperature, stopping temperature, decrease factor and markov chain length.

Initial phase parameters

Figure 44 shows solution cost over three different test instances given their input bias factors. We test 10 different bias factors. Also, the mean time to solution is displayed.

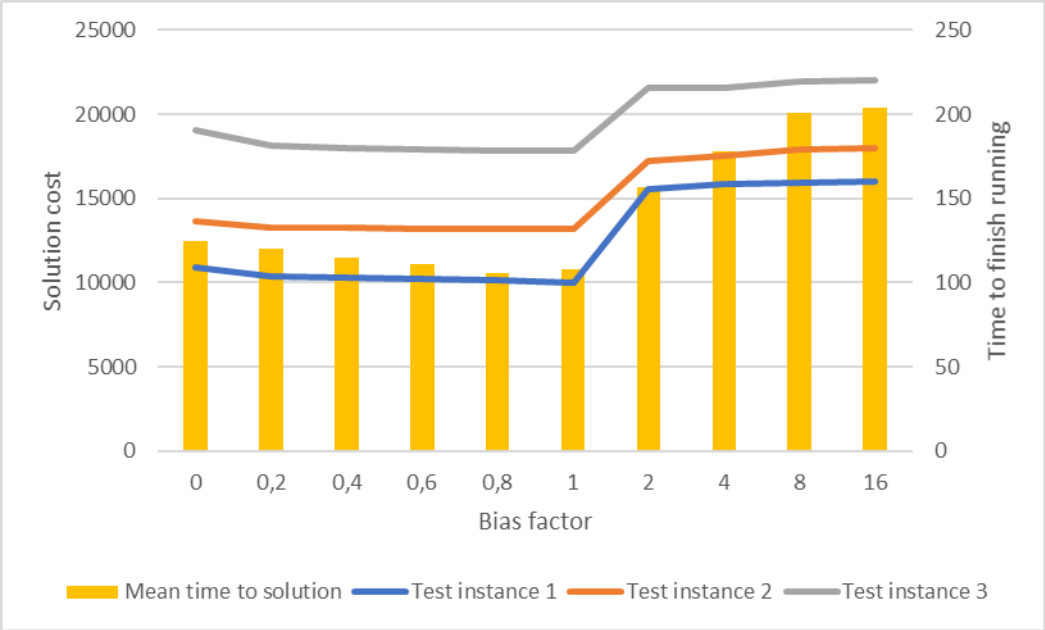


Figure 44 – Solution progression over time for different Bias Factor values

Choice for stopping criterion

Figure 45 shows the tradeoff between the time used and the solution cost for different values of the stopping criterion. The mean indexed solution cost is determined by averaging the solution costs of the initial phase, after which the values are normalized with a minimum of 100. Logically, the mean time used increases with a greater stopping criterion value as more removal attempts will be made.

Figure 45 shows that the solution cost initially declines quickly with a greater stopping criterion length. After some time however, the gains are only marginal while the time used continues to increase. For this reason, we set the length of the stopping criterion i to 100. This means that the initial phase stops after 100 iterations without a feasible removal attempt.

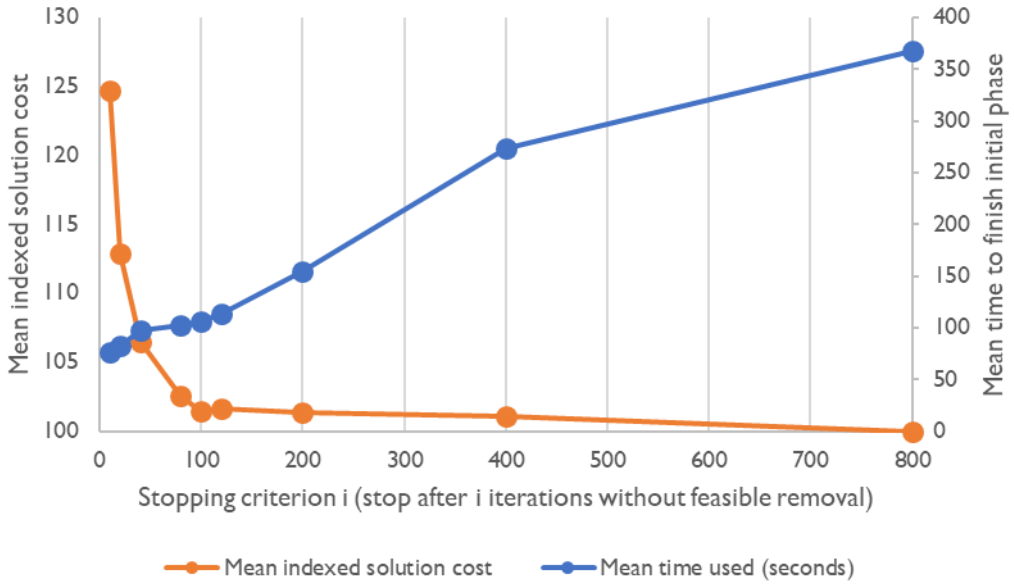


Figure 45 – Mean indexed solution cost and mean time used for different stopping criterion lengths

Improvement phase parameters

We introduced several parameters for the simulated annealing procedure. We test different values to determine the optimal values.

Determining starting and stopping temperatures.

Worsening solutions are accepted with probability $e^{-\frac{f(s')-f(s)}{T}}$. $\chi(c)$ is defined as the acceptance ratio of these worse solutions. We seek a value of $\chi(c) = 1$ (Aarts & Korst, 2005) slowly converging towards $\chi(c) = 0$, so that we start with a global optimisation approach, ending in local optima.

By intuition, we find that a single change in neighbourhood can never deteriorate the solution cost more than 48 * 2 (by swapping a shift from the least to most expensive time window in the semi-automatic depalletizing process). Figure 46 shows that the probability of accepting the worst objective function deterioration is close to 1 for $c > 10000$, implying that $\chi(c) \approx 1$ for $c = 10000$. Figure 46 tests this hypothesis. We observe that from around c = 8000, the acceptance ratio is close to 1 indeed. Therefore we set $c_{start} = 10000$

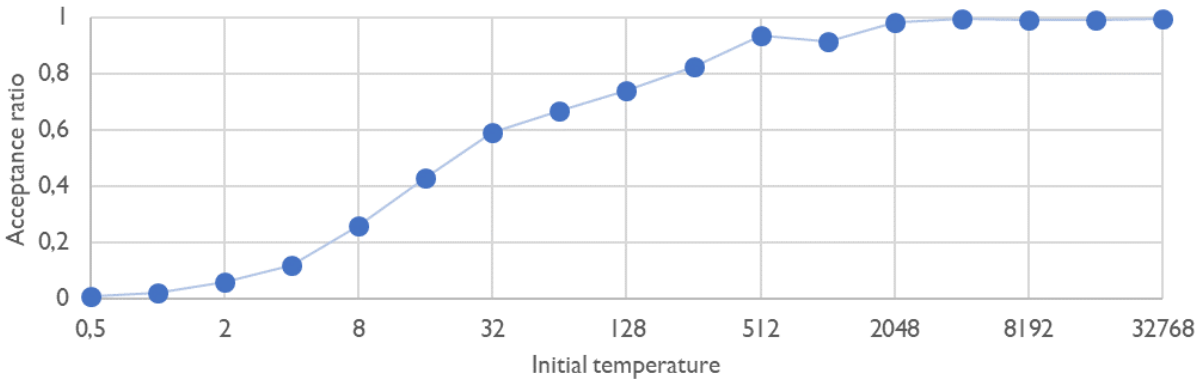


Figure 46 – Acceptance probability for minimum and maximum cost deterioration

We do not want to continue annealing if the acceptance probability stays close to 0. Therefore, we decide that $c_{stop} = 1$ as the minimum deterioration f(s') – f(s) = 2, we expect few improvements below $c_{stop} = 1$ as Figure 46 shows.

Markov chain length

As we finish the constructive heuristic, many 'easy wins' are available, such as swapping to less expensive timeslots. Therefore we expect that we can find improvements relatively often in early annealing. Perhaps an incremental chain length works better than a constant length, where the improvement over each chain is constant. We observe that the scheduling algorithm can be performed around 10 times per second on a regular laptop. For the experiments, we target a running time of 10 minutes to compare the two strategies. Therefore, we expect to be able to perform around 600*10 = 6000 iterations in total.

Incremental length markov chain

We test four incremental length markov chain, each amounting to around 6000 total iterations and 300 chains, so equating to approximately the same runtime length. Table 16 shows multiple parameter configuration for the starting markov chain length and the number of chains to increment after. We note that an initial short length is generally beneficial, advocating the hypothesis that many improvements can be found early on following the introduction of new neighbour operators. We decide to use an initial length of 5, incrementing the length by 1 after every 10 iterations.

Starting chain length	Increment chain length by +1 after n iterations	Mean solution cost (of 3 test instances, each ran 5 times)
3	9	12682
5	10	12665
8	12	12818
10	15	12940

Table 16 – Mean solution cost for different incremental markov chain parameters

We test two scenario's: one where the markov chain length is constant at k = 20, therefore we expect to be able to evaluate around 300 chains. Also, we test an incremental markov chain length starting at length 5, then incrementing the length by 1 after every 10 chains. This way, we evaluate the same 300 chains ending at a length of k = 35.

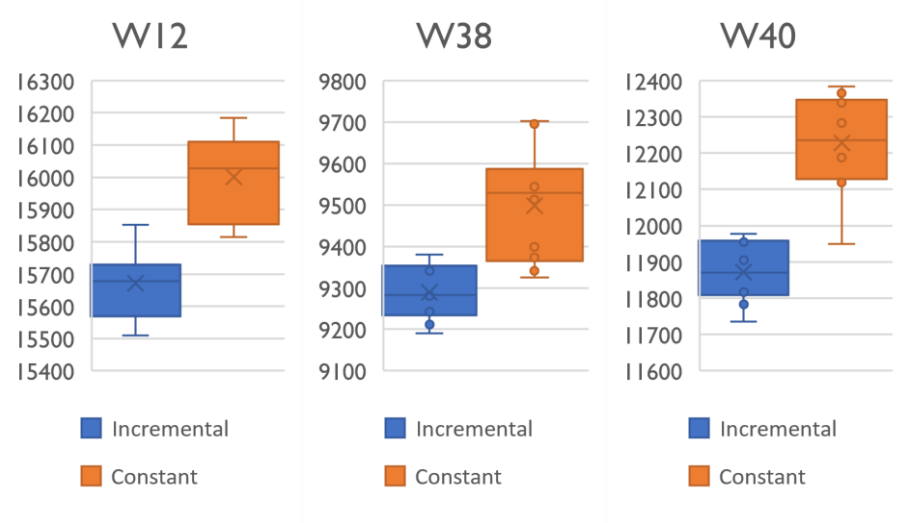


Figure 47 – Comparing the incremental and constant markov chain length

We use three test instances, (t2,t4,t5) specified in Chapter Appendix C. Figure Figure 47 shows that the incremental chain length outperforms the constant chain length in most cases. We decide to use the incremental chain length.

Decrease factor

We aim to go from initial temperature 10000 to the stopping temperature 1 in about 10 minutes, this equates around 300 chains. This equates a decrease factor of 0.97

Appendix C: Test Instances

Table 17 shows the test instance properties. We use these instances to compare strategies. These instances are based on (expected) demand in the distribution centre for a given week. We extend the test set by a factor 2 by adding instances with demand multipliers 0,8 and 1.2, obtaining 15 total test instances.

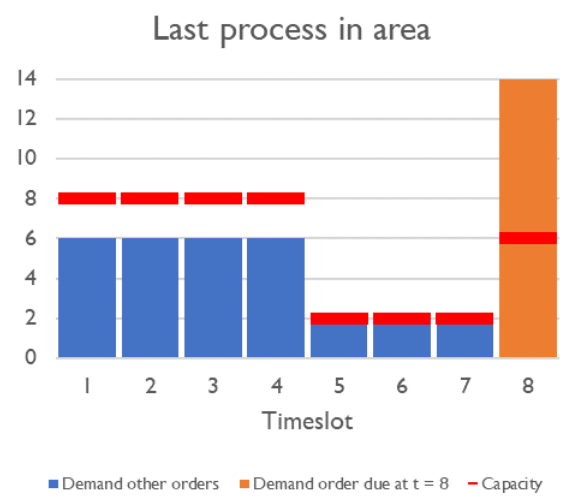
Instance	**1**	**2**	**3**	**4**	**5**
Demand(week) in thousands	1107	1234	1356	1751	1979
D(Mon)	61	159	242	242	262
D(Tue)	158	157	194	237	237
D(Wed)	178	180	169	267	285
D(Thu)	189	205	282	282	354
D(Fri)	198	215	180	301	397
D(Sat)	209	196	169	302	380
D(Sun)	114	122	120	120	64
RRP included?	Yes	Yes	No	No	No
Maxlead (stdev) in hr	8.5 (0)	11.84 (0.86)	12 (0)	12 (0)	12 (0)

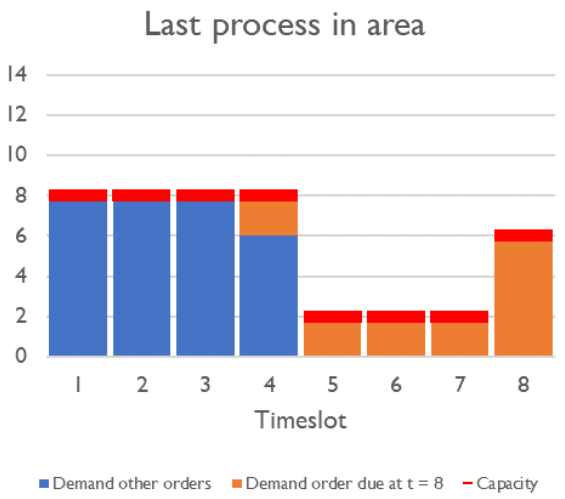
Table 17 – Test instance properties

Appendix D: Infeasible allocation example through lead time violation

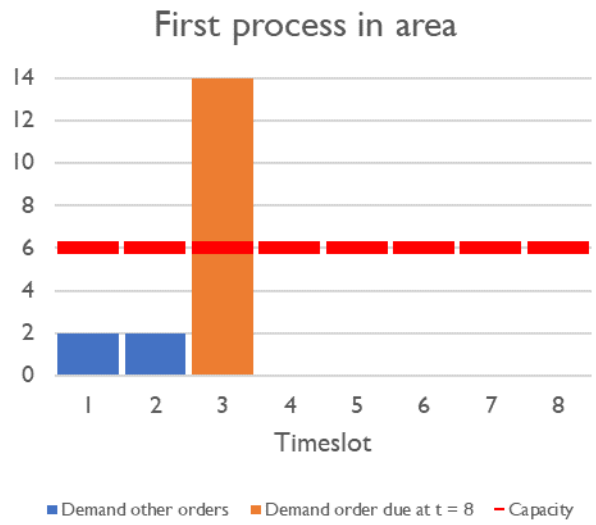
We consider demand for an area that constitutes two processes. As we schedule recursively following the priority rule, we start with the last process. The capacity at the process differs over time due to different multiples of workstations being allocated. First, we schedule all orders in the timeslots in which they are due. In this example, we highlight demand due in timeslot 8. The demand that is due in this timeslot has a maximum lead time of 4 timeslots, this cannot be exceeded (cut-off offset constraint).

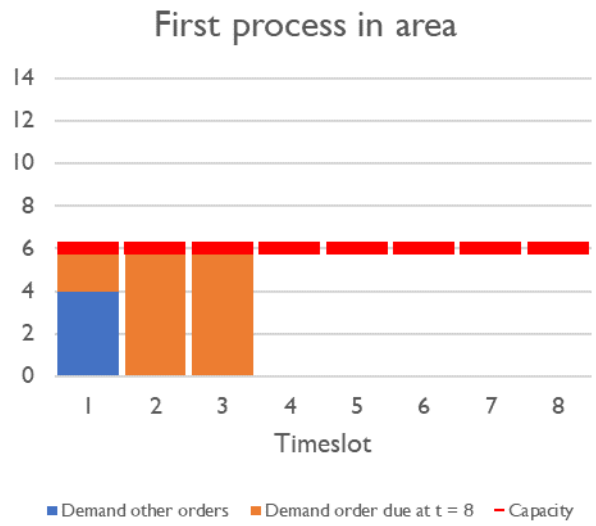
We observe that initially, there is excess demand at the final timeslot, so we must shift some demand forward.





As we shift demand, lead time is added. We note that production in the last process must start at timeslot 4 already, rather than timeslot 8, as the production must be finished by the timeslot in which it is due (timeslot 8). Now this create some problems: processing at the first process cannot start at t=7, but only at t=3.





Similarly, there is under capacity at the first process. Again, lead time is added as processing starts at timeslot 1.

While eventually there is no undercapacity for this demand, another constraint causes infeasibility. The lead time for timeslot t = 8 is exceeded. Production for this order must start at timeslot t=1 if it is to be finished by t=8, while only 4 timeslots (or a start at t=5) would be allowed.