UNIVERSITY OF TWENTE.

Developing an Order Acceptance Function for Limis Planner

Master's Thesis

Industrial Engineering and Management

Author:

Vusal Abbaszade

Supervisors University of Twente:

Dr. ir. J.M.J. Schutten

Dr. ir. E.A. Lalla

Supervisor Limis BV:

Ir. H. ten Brug

June 16, 2022

Management Summary

We conduct this research for Limis which is the developer of an Advanced Planning and Scheduling software called Limis Planner. Limis' customers, who are users of Limis Planner, typically have job shop-like manufacturing systems in which they make end products fully or partially in response to customer demand. Therefore, they need to respond to dynamically arriving customer requests each of which consists of one or multiple products resulting in one or multiple customer orders. The company (i.e. Limis' customer) needs to determine whether they can deliver the requested quantity by the requested due date for an incoming order based on material and capacity availability. In the case of material or capacity shortage, they typically have three types of flexibility options to consider, namely, due date flexibility (of incoming orders and some selected existing orders), capacity flexibility (by means of overtime, and subcontracting), and quantity flexibility (i.e. fulfilling an incoming order by partial shipments). The currently used scheduling function of Limis Planner does not consider capacity and due date flexibility and assumes flexibility of all order due dates. In addition, it requires a high solution time causing a high customer response time. This research aims to design a fast and reliable order acceptance method to be used by Limis' customers to respond to and act on dynamically arriving customer orders.

The literature review reveals the limitations and assumptions of studies conducted to address similar planning problems. Mainly based on the solution time requirements, we decide to address the problem using a so-called finite loading method in combination with some heuristics applied in the literature to solve permutation-based optimization problems. Finite loading methods are typically implemented like control policies or procedures to load an incoming order on top of existing orders using a discrete-time representation of resources' capacity, for example, for the purpose of quoting a due date to the customer. In finite loading methods, operations are assigned due dates which are used to dispatch them ensuring that capacity is used as planned. The parameters of a finite loading method are typically determined through simulation experiments so as to improve the accuracy of assigned order due dates. We choose Cumulative Forward Finite Loading (CFFL) as the finite loading method as it allows us to advance an operation by activating overtime without the need to replan previously loaded operations. CFFL assumes due date flexibility and we extend it to consider capacity flexibility and the operating conditions of Limis' customers. Moreover, CFFL originally has two planning parameters called the minimum waiting time (MWT) and capacity loading limit (CLL), and we add another planning parameter called the due date buffer (DDB) to further improve tardiness performance. We refer to the CFFL-based procedure as the loading procedure. In order to find a 'good' sequence in which to load the orders under consideration during the order acceptance, we test three heuristics, namely, a constructive heuristic (CH), Local Search (LS), and Tabu Search (TS) which commonly use the loading procedure as a subroutine to evaluate loading sequences.

We conduct experiments using the dataset of a Limis' customer. Through simulation experiments, we gain insights into and configure the parameters of the loading procedure. Based on tardiness performance, particularly the percentage of tardy orders, we choose the configuration of the planning parameters shown in Table M.1. The results show that although the procedure activates overtime on a timely basis (e.g. 83% overtime utilization with the chosen configuration), there is a significant variance of lateness (e.g. 2.75 days with the chosen configuration) which is defined as the difference between the internally assigned due dates of orders and their completion dates. The high variance of lateness makes us set a high value for DDB (i.e. 6 days) to ensure a sufficiently low percentage of tardy orders

(e.g. 0.76% with the chosen configuration) for the chosen Limis' customer. Moreover, although we plan incoming orders based on their due dates without replanning existing orders, the average due date extension of incoming orders is significantly low (e.g. 0.56 days with the chosen configuration). This shows that because the chosen Limis' customer receives orders with loose customer-requested due dates, there is often no need to replan existing orders or use an an advanced method (e.g. CH, TS) to determine a sequence of orders during the order acceptance.

Table M.1: Chosen configuration of the planning parameters

$MWT^*(hours)$	$CLL^*(\%)$	$DDB(days)$
5;0;0;2;3;3;0;2;6;5;5;6;6	0.9;1;1;0.95;0.95;0.95;1;0.95;1;0.9;0.9;1;1	6

*each of the 13 numbers in a cell below is for the corresponding workstation.

We conduct further experiments to compare CH, LS, TS in terms of total due date extension of incoming orders and solution time. The maximum acceptable solution time for Limis' customers is 5 minutes. We run TS with time limits of 5 minutes (i.e. TS-5) and 10 minutes (i.e. TS-10), LS with a time limit of 5 minutes (i.e. LS-5) and without a limit (i.e. LS). Because CH obtains a full solution at the end, we cannot limit its solution time. CH and TS-10 with an average total due date extension of 9.8 days slightly outperform LS, LS-5, and TS-5 which result in 10 days of total due date extension on average. In terms of solution time, CH with an average solution time of 89.4 seconds significantly outperforms LS, LS-5, TS-5, and TS-10 which run in 214.2, 300, 266, and 600 seconds on average respectively.

In contrast with LS and TS, CH is a non-parametric method, which makes it easy to implement for other Limis' customers as well. We recommend that Limis implements the loading procedure in combination with CH. Overall, the proposed order acceptance method (i.e. the loading procedure in combination with CH) is a fast and reliable method that takes into due date and capacity flexibility. Operation due dates assigned by the loading procedure are used as input for detailed scheduling to ensure that capacity is used as planned and detailed schedules are in line with plans.

Acknowledgments

This thesis marks the end of my master's study in Industrial Engineering and Management at the University of Twente. I would like to gratefully acknowledge those who supported me during my studies and graduation project.

I would like to express my sincere gratitude to my first supervisor, Marco Schutten, who helped me to find my internship company, Limis, and guided me during the project with his feedback and ideas. I am also thankful to my second supervisor, Eduardo Lalla, who provided valuable feedback on how I could improve my thesis. I would like to thank Hans ten Brug for giving me the opportunity to conduct my graduation project at Limis and patiently answering my questions during the project.

Moreover, I am thankful to the University of Twente for the generous support through the University of Twente Scholarship which enabled me to undertake my master's study.

Lastly, I would like to thank my family for their immense moral support and encouragement during my studies.

Vusal Abbaszade

Contents

1 Introduction

In this chapter, we introduce the research project conducted at Limis. The chapter starts with the information about Limis and its software Limis Planner. Section 1.2 describes the context of the problem addressed by this thesis. Section 1.3 defines the research objective and outlines the research questions. In Section 1.4, we define the scope of this thesis project.

1.1 Company description

Limis is a 28-year-old technology company located in Enschede, the Netherlands. It develops and implements planning software for companies in the manufacturing industry, called Limis Planner. Limis Planner is an Advanced Planning and Scheduling (APS) system. APS systems are manufacturing planning and control systems that take into account finite material availability and resource capacity constraints and are typically based on optimization algorithms (Hvolby & Steger-Jensen, 2010). The software can be integrated with the existing ERP system of a company for exchange of data, which helps to avoid the tedious and error-prone task of re-entering data about orders, products, and stock and feeds back information about order status and production performance to the ERP system.

Limis Planner provides numerous benefits to its users such as reduction of inventory (i.e. raw materials, work-in-process, and finished goods), shorter and reliable delivery times, savings on setup time and time for planning preparation, cost savings through reduced personnel overtime and activity outsourcing. One of the functionalities of the software is that it offers live insight into order status through regular progress registration by employees on the shop floor. This up-to-date information allows for a timely response in cases of disruptions or overruns in the processing of the current orders as well as for quoting reliable due dates for upcoming orders.

Limis Planner is designed primarily for companies in the discrete manufacturing industry. Currently, Limis' customers are active in metal, machinery, plastic, and high-tech sectors. For example, the customers in the high-tech industry have to deal with complex assemblies having multi-level bills of materials and diverse production routings, and therefore face challenges in maintaining the availability of purchased and manufactured parts, outsourcing activities and also managing available capacities. In addition to serving as a production planning and scheduling tool, the software also supports the material planning activities of these firms through its MRP functionality. Limis' customers mainly have a job shop-like production system, which is characterized by producing a large variety of products in small batches and even one-of-a-kind products.

Limis Planner has a modular structure, which allows to design the software to the needs, sizes, and complexities of different companies. For example, within the scheduling function, machine, material, workforce, and tool planning can be added separately.

1.2 Problem description

As mentioned in Section 1.1, Limis' customers typically have job-shop manufacturing systems. In such systems, resources such as machines and employees are grouped as work centers according to processes (e.g. milling, drilling). Production occurs in small batches and products can follow a large variety of routings along the work centers. Job shops typically supply parts or components that later become part of complex end products.

Some Limis' customers pursue a hybrid production strategy to fulfill the needs of their customers. End products are produced in reaction to demand, while some components are produced based on forecast, meaning that those end products are hybrid make-to-stock make-to-order products. The primary objective of this strategy is to reduce customer lead times.

Some of Limis' customers that are in the high-tech industry supply components or modules to ASML, one of the world's leading semiconductor equipment manufacturers. ASML is the key customer of these firms and requires high service levels for the availability of some products . At the beginning of a year, ASML shares its aggregate (i.e. quarterly) forecast of each product for the coming year with its tier-1 suppliers so that the suppliers can plan their capacities and production rates accordingly. Moreover, the orders for the upcoming 2 months are usually fixed by ASML. ASML is known for constantly altering the production rates at its suppliers and it also tends to place orders with tight due dates or expedite an already placed order. This situation has also been exacerbated by the recent global chip shortage, which has put enormous pressure on semiconductor supply chains to ramp up production.

At Limis' customers, a customer may request multiple products leading to multiple orders. When a customer order(s) arrives, the company (i.e. Limis' customer) has several alternatives to consider if it is not possible to deliver the requested quantity until the requested due date due to capacity or material shortage. First, the company can negotiate an extended due date with the customer. Second, if the requested quantity is more than one unit, the company can check if it is possible to fulfill the order partially until the requested due date and fully until a later date, which also requires negotiation with the customer. Next, the company can delay some of the existing orders, A delayed order can be an order for a make-to-stock component, or a customer order belonging to the same customer or other customers. Delaying a customer order also requires negotiation. Moreover, if the cause of lateness is capacity shortage, capacity can be increased by means of, for example, overtime, subcontracting, and reallocation of employees among workstations. Reallocation is partially possible since employees are usually cross-trained and can operate multiple machines, whereas some machines require highly skilled employees that cannot be easily replaced by other employees. In short, there are three types of flexibility options that a planner can leverage to satisfy an incoming customer order: 1) due date flexibility, i.e. extending the customer-requested due date of the incoming order or extending the agreed-upon due dates of existing orders, 2) capacity flexibility, i.e. overtime, subcontracting, reallocation of employees, 3) quantity flexibility, i.e. fulfilling the incoming order with partial shipments.

Upon the receipt of a customer order(s), the company needs to determine the effects of accepting the order on the current production plan so that they can take necessary actions if there is a need to use any of the above-mentioned flexibility options. In doing so, low customer response time is highly desired. If the due date needs to be altered or partial shipments are required, the customer and the company may need to hold iterative negotiations. Negotiation with the other customers may also be required if their orders have been decided to be delayed by the company. In the case of capacity

increase, the planners need to know the timing of extra capacity requirement so that they can make necessary arrangements.

Currently, the detailed scheduling functionality of Limis Planner is used by Limis' customers to make decisions during the order acceptance phase. All the existing orders and the incoming order(s) are scheduled to see if the customer-requested due date(s) can be met and the due dates of the existing orders are adhered to. Bottleneck resources are also identified as a result of scheduling. The planner may need to run the scheduling multiple times to test the above-mentioned flexibility options. Note that the scheduling as a finite-capacity tool is not able to alter the capacities by itself. The scheduling involves a high level of detail with regard to operations, resources, etc. and as such it is considered highly reliable in terms of meeting the due date once it is agreed on. Reliability in this sense is important as the company may incur high penalty costs in cases of late delivery. However, with some Limis' customers, a single run of the scheduling can last as long as an hour due to the large number of orders and thus using it for the acceptance of dynamically arriving orders is inefficient. Thus, the users of Limis Planner are in need of a decision-making tool that is able to quickly and reliably accept an incoming customer order while taking into account some or all of the above-mentioned flexibility options.

1.3 Research objective and questions

The objective of this research is to design a fast and reliable method within Limis Planner that aids the users in their decision-making during the acceptance of incoming customer orders. Such a tool would allow the users to efficiently plan their production activities in a dynamic environment while improving their customer service in terms of both response time and delivery performance. We define the main research question as follows:

How can a fast and reliable order acceptance method be designed within Limis Planner in order to respond to and act on dynamically arriving customer orders?

To achieve the research objective and answer the main research question, we define several research questions as follows.

To begin with, it is important to have a thorough understanding of the main functionalities of Limis Planner, including the scheduling function. Furthermore, we conduct an analysis of the production environments at Limis' customers. The obtained knowledge can be used to develop and test the performance of the method and understand the interaction between the planning and scheduling. In Chapter 2, we answer the following research question and its subquestions.

1. What is the current situation regarding Limis Planner and the characteristics of Limis' customers?
   - What are the main functions of Limis Planner?
   - How does the scheduling function work?
   - What are the characteristics of the production systems at Limis' customers?

An essential part of the research is to conduct a literature review to obtain knowledge on the methods that have been previously used in similar contexts. We investigate different streams of literature dealing

with order acceptance in any form. The literature review may also help to find out the ways in which the performance of the proposed method can be measured. In Chapter 3, we answer the following research question and its subquestions.

2. What information related to the order acceptance problem is available in the literature?
   - How does the order acceptance process take place in various production environments?
   - What methods have been used to address order acceptance problems? What are the limitations and assumptions of the studies?
   - What are the performance measures used to assess those order acceptance methods?

Based on the findings from the literature, we design a solution method for the order acceptance problem at Limis' customers. In Chapter 4, we answer the following research question and its subquestion.

3. How should the order acceptance problem at Limis's customers be addressed?
   - What methods can be used to solve the order acceptance problem?

Next, we conduct experiments with the proposed method. In Chapter 5, we answer the following research question and its subquestions.

4. How does the proposed method perform according to the selected criteria?
   - How should the experiments be designed?
   - What are the results of the experiments?

Finally, in Chapter 6, we present conclusions and recommendations for future research.

1.4 Scope and assumptions

In this section, we list several assumptions that define the scope of this research project.

1. All the relevant data possessed by Limis' customers is available in the Limis Planner database. This assumption is valid as the customers use Limis Planner as the sole tool for production planning and scheduling.
2. The scheduling function cannot be optimized to run faster as it is based on simple rules. Therefore, we will not consider the detailed scheduling as an alternative or benchmark method to be used in the order acceptance.

2 Context Analysis

This chapter describes the current situation at Limis. Section 2.1 shows what relevant data is available in Limis Planner as entered by companies. Section 2.2 provides insights into the customer data to be used in this research. In Section 2.3, we explain the scheduling function of Limis Planner. Section 2.4 provides the summary of the chapter.

2.1 Input data

The relevant data that is typically entered in Limis Planner is formatted as follows.

- Data about production resources includes the necessary data about workstations, employees, and tools. A workstation consists of either a single machine or a group of identical machines. The most important pieces of information about employees are their regular daily working hours and the workstations where they can be deployed. In each workstation that an employee can be assigned to, he/she may carry out setup or processing (or both) and has a skill level (between 0-9). Skill levels are used during the scheduling to allocate the most suitable employees to the workstations.

  Although workstations are continuously available, their availability is constrained by the availability of employees. In other words, machines are also assigned daily working hours. It may be possible to reduce the processing times of production orders on a machine by increasing the number of employees referred to as the *crew factor*. When a certain operation is subcontracted to an external organization, the subcontractor is registered in Limis Planner as an external workstation.

- Production orders are meant for either make-to-stock items (i.e. *stock orders*) or make-to-order items (i.e. *customer orders*) and consist of operations requiring a particular workstation (and an employee, tool). Operations typically cannot be preempted and they may have a setup time and waiting time after processing. A production order may have a *string-type routing* where each operation has at most one predecessor and at most one successor or an *assembly-type routing* where operations may have multiple predecessors.

  The due dates and quantities of customer orders are typically specified by customers, while the due dates and quantities of stock orders are determined by the built-in or external Materials Requirements Planning (MRP) function. Furthermore, the production specifications (i.e. routings, processing times, material requirements, etc.) of a stock order can be derived from the bill-of-materials and routing of the corresponding make-to-stock item, whereas customer orders for make-to-order items typically have unique specifications.

  Customer requests arrive dynamically at the shop and each customer request includes one or several items resulting in one or several customer orders. The MRP function can generate new stock orders upon the arrival of a customer request.

  Production orders are also assigned a priority between 0 and 9 related to their urgency. These priorities are used by a dispatching rule, as explained in Section 2.3.2.

2.2 Customer case

We have chosen one company whose data is used for experimentation in this research. The company operates in a make-to-order environment where end products are mainly made in response to customer orders with almost no make-to-stock components. This makes the company more suitable for experimentation since we do not have to consider stock orders which are supposed to be generated by an MRP function.

Employee and tool planning are not considered for the company in Limis Planner. All the production orders in the dataset are customer orders. Customer orders have input dates but are not linked to specific customer requests. We assume that the customer orders registered on a certain day all belong to a single customer request, which is often the case in reality. We also obtain the order due date slack factors, by dividing the difference between the due date of an order and its arrival date by its critical path length. The critical path length of an order is the minimum time it would require to complete the order if no waiting time is incurred for the operations. Table 2.1 shows the descriptive statistics on the characteristics of the company.

About 55% of the 346 orders have an assembly-type routing and those orders tend to have more operations (7.45 on average) than the orders with a string-type routing (3.03 on average). The standard deviation of operation processing times is considerably high. 81% of operations have no setup time, while the remaining operations have a negligible setup time (i.e. 0.1 hour). As can be observed from the figures regarding order due date slack factors, the due dates are considerably loose meaning that the company does not typically encounter rush orders.

Of the 23 workstations, 6 are external workstations. For some of the external workstations, the company has internal alternatives. Each workstation has a single machine. Daily working hours varies across the machines, as well as the days of the week.

2.3 Scheduling

As is common in most manufacturing systems, Limis' customers operate in a dynamic stochastic environment. Dynamism relates to the arrival of (rush) customer orders over time as well as other unforeseeable events such as machine breakdowns, and employee absenteeism. Stochasticity refers to, for example, the uncertainty regarding the estimates of processing and setup times. Limis' customers constantly reschedule to respond to the dynamically changing conditions. For example, in the case of a machine breakdown deemed to require considerable repair time, it is wise to update the schedule promptly, whereas (non-rush) customer orders arriving continuously over time are typically incorporated into the schedule periodically (e.g. on a daily basis) to avoid nervousness on the shop floor. Moreover, to decrease the effects of stochasticity or increase the robustness of schedules, Limis Planner offers a functionality to increase processing time estimates by a factor. In essence, Limis Planner treats the scheduling problem as a static deterministic problem.

MRP is run automatically before every run of the scheduler and provides the following input for the scheduling: 1) on-hand inventory of purchased items and the delivery dates and quantities of future deliveries, 2) on-hand inventory of make-to-stock items and the due dates and quantities of future

Table 2.1: Descriptive statistics of Limis' customer

Orders		
Number of (production) orders		346
Percentage of orders with an assembly-type routing		55
Number of customer orders per customer request	*mean*	5.29
	min	1
	max	25
Interarrival time of customer requests (days)	*mean*	3.91
	st. dev.	7.29
Number of operations per order (assembly-type routings)	*mean*	7.45
	min	4
	max	12
Number of operations per order (string-type routings)	*mean*	3.03
	min	2
	max	5
Operation processing time (hour)	*mean*	4.58
	st dev	9.84
Order due date slack factor	*mean*	27.83
	st dev	22.52
Resources		
Number of workstations		23
Number of external workstations		6
Number of machines		23
Weekly machine capacity (hours)	*mean*	38.35
	min	10
	max	50

stock orders. The due date and quantity of a stock order is not only needed for checking the availability of the corresponding make-to-stock item for the production of its parent item, but the stock order also needs to be scheduled by Limis Planner such that its MRP-requested due date is actually met. Indeed, it is hard for Limis Planner to meet MRP-requested due dates, because MRP uses fixed lead times to determine due dates regardless of the quantities and the workload on the shop floor.

The scheduling in Limis Planner takes place in two stages, the rough planning and the detailed scheduling. Figure 2.1 depicts the relationships between the two planning methods and the various output modules in Limis Planner.

2.3.1 Rough planning

In essence, the rough planning method of Limis Planner is backward infinite loading, which takes into account neither the workload at workstations nor material availability.

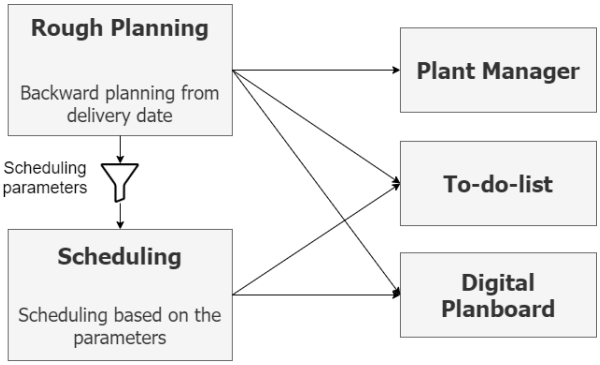


Figure 2.1: Structure of Limis Planner

First, by planning each operation backward from its due date (or late planned finish date), we find its late planned start date. The due date of an operation is equal to the late planned start date of its successor while the due date of the last operation of an order is equal to the due date of the order. The throughput time of an operation is calculated using the following formula which includes possible setup time, waiting time after processing and transport time in addition to the processing time and average waiting time (before processing).

$$\tau = Q + S + P/(C \times CF) + W + T \tag{1}$$

$\tau = throughput\ time\ of\ the\ operation,\ in\ working\ days$
$Q = average\ waiting\ time\ at\ the\ workstation,\ in\ working\ days$
$S = sequence - independent\ setup\ time,\ in\ working\ days$
$W = waiting\ time\ after\ processing,\ in\ working\ days$
$T = transport\ time,\ in\ working\ days$
$P = processing\ time\ of\ the\ operation,\ in\ hours$
$C = average\ daily\ capacity\ of\ the\ machine,\ in\ hours/day$
$CF = crew\ factor,\ or\ number\ of\ employees$

The crew factor in Equation 1 refers to the situation where increasing the number of operators for the machine would decrease the processing time of the operation. The setup time and the waiting time after processing may depend on the workstation, or may be unique to the operation. There is no transport time for the last operation of an order. The average waiting time of operations is estimated by the company through observations on the shop floor and it is meant for absorbing the effects of not taking into account material availability and workload at the workstations.

The release date of an order is set to the minimum of the late planned start dates of its operations. Next, the critical paths of orders are found. The critical path of an order consists of the operations that determine the length of the order and would cause the order to be late if one of them is late. The early planned start and finish dates of a critical operation are set to the late planned start and finish dates respectively. Then, by planning each non-critical operation forward from its release date (or early planned start date), we find its early planned finish date. The release date of an operation is equal to the maximum of the early planned finish dates of its predecessors while the release date of a starting operation of an order is equal to the release date of the order.

As a result of the rough planning, each order is assigned a release date and due date and each opera-

tion is assigned a release date, due date, early planned finish date, and late planned start date. These dates are meant to be used by some priority rules explained in Section 2.3.2 for sequencing operations. It is worth noting orders or operations are typically allowed to start before their release dates. This is referred to as *immediate release* or *uncontrolled release* situation in the literature (Ahmed & Fisher, 1992; Bertrand, 1983). Moreover, the scheduler offers a possibility to delay an order, referred to as a *just-in-time (JIT) order*, up until a certain number of working days before its release date. The period within which the release date of a just-in-time order can be advanced is called *margin*.

2.3.2 Detailed scheduling

The detailed scheduling determines the definite start and finish times of operations and hence the times when orders will be complete. The detailed schedule is visualized as a Gantt chart in the Digital Planboard module of Limis Planner. The operations are also listed in the To-do List module where they can be filtered for each employee, workstation, and machine.

In the detailed scheduling, availability of materials (i.e. purchased or make-to-stock items) required by operations is taken into account. If there is a shortage of a required material at the time an operation can start (because the predecessors are complete), and there is no planned receipt of the required material in the future, a fixed lead time specific to the material is taken into account, after which the operation can possibly start.

The detailed scheduling is essentially based on priority dispatching rules and works similar to the parallel generation scheme (Kolisch, 1996). It results in a non-delay schedule in which no resource is kept idle when there is an operation that can start, i.e. the predecessors are complete and the other required resources are also available. Figure 2.2 shows the flowchart of the detailed scheduling in its basic form. Some of the assumptions are as follows: there is no just-in-time order, there is no waiting time after processing and transport time, materials are continuously available. These points and additional features as mentioned below can be easily incorporated into the algorithm.

The four sets mentioned in Figure 2.2 are defined as follows: the complete set includes the operations that have finished processing, the decision set includes the operations that can be started since all the predecessors are complete, the active set includes the ongoing operations, and the remaining set includes the other operations. In the initialization phase, the active set is filled with operations that are running on the shop floor at the time of the scheduling run. In the decision set, operations are sorted based on the priority rules explained below. In Step 2, some operations cannot be started because the resources (i.e. machine, operator, tools) they require are allocated to some higher priority operations.

Table 2.2 lists the rules that are available in Limis Planner to prioritize the operations in the decision set. PRIO and ORD are among the most commonly used rules by Limis' customers.

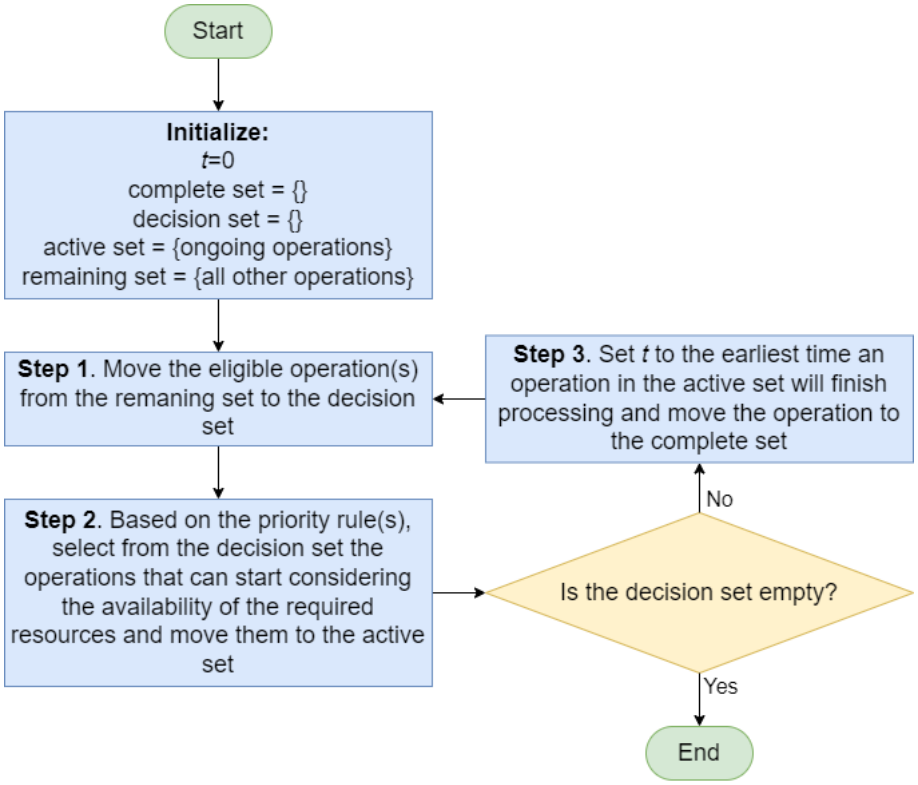


Figure 2.2: Flowchart of the detailed scheduling

Table 2.2: Priority rules

Rule	Purpose
Priority (PRIO)	As mentioned in Section 2.1, each accepted order is assigned a priority number between 0-9. This rule ranks orders based on their preassigned priority numbers. An order with a higher priority number takes precedence over an order with a lower priority number. This rule can be used to ensure that rush or expedited customer orders are delivered in the shortest possible time.
Earliest operation release date (ORD)	This rule ranks the operations based on the earliness of their release date. This rule is used to minimize the variation in the waiting times.
Earliest operation release week (ORW)	This rule ranks the operations based on the earliness of their release week. Although this rule is similar to ORD rule, it is effective in reducing the number of setups, when used in combination with SST rule. In other words, the batching of the operations starting the same week provides more opportunities to combine the operations having the same setup.
Earliest operation release month (ORM)	This rule ranks the operations based on the earliness of their release month. It has the same benefit as ORW rule but may have a negative impact on delivery performance. Long sequences based on the same setup may lead to significant lateness in the other operations.
Earliest operation due date (ODD)	This rule ranks the operations based on the earliness of their due date. This rule improves the delivery performance.

continued on next page

Table 2.2: continued

Rule	Purpose
Earliest operation due week (ODW)	This rule ranks the operations based on the earliness of their due week. When used in combination with AS, it can achieve a balance between delivery performance and the number of setups.
Longest remaining processing time (LRPT)	Operations are sorted in decreasing order of the total remaining processing time of their order in the subsequent workstations. This rule decreases average order flow time.
Shortest processing time (SPT)	Operations are sorted in increasing order of their processing time. This rule is aimed at reducing average order flow time and minimizing queue sizes.
Shortest setup time (SST)	Operations are sorted in increasing order of their setup time. In order to use this rule, sequence-dependent setup times are entered by the company. This rule is often used in combination with other rules such as ORW.
Earliest due date (EDD)	This rule ranks the operations based on the earliness of the due date of their order. It improves the delivery performance.
Earliest due week (EDW)	This rule ranks the operations based on the earliness of the due week of their order. It improves the delivery performance.
Maximum lateness (ML)	Operations are sorted in decreasing order of their lateness, where the lateness of an operation is defined as the time until or past its late planned start date. This rule is aimed at minimizing the maximum lateness of orders.
On-time (OT)	Operations are sorted in increasing order of their lateness. The lateness of an operation is defined the same way as in ML rule. OT is the opposite of ML rule and helps to improve the delivery performance of non-late orders.

There are some practical situations faced by Limis' customers that have lead to the integration of additional features into the detailed scheduling over time.

First, there is a possibility to cluster operations that are to be performed on the same workstation, as specified by the planner. In that case, the arriving operations of a cluster are delayed until all the operations of the cluster arrive at the workstation. *Clustering* can be done, for example, for the purpose of minimizing scrap when multiple operations need to be processed on the same material. In metal shearing processes, metal parts of different shapes can be punched or cut out of the same metal sheet so as to make the best use of the material.

Another feature, referred to as *load balancing*, helps to choose the workstation or an employee for an operation when it can be processed by multiple workstations or employees. For example, the company may have sheet metal working machines that perform the same operation but can handle materials up to different sizes. The operation is allocated to the workstation which becomes available the earliest.

Another special case in job shops is that an employee can operate multiple (identical or non-identical) machines at the same time. This happens when, for example, the operator sets up the machine for an operation and while the machine processes the operation on its own, the operator can set up another operation in a different machine. The operator may need to remove the workpieces from the machines at the end of their processing. The number of machines that an operator can oversee simultaneously depends on the parameter of the machines, called *manned percentage*, which refers to how much of the operator's attention is required. For example, an operator can operate two machines which have

manned percentages of 40% and 60% respectively.

Sometimes, operations or machines need to be processed by multiple operators at the same time. In this case, the operation is delayed until the required number of operators are available, and meanwhile the machine, the already available employees, and the tools are held on. Another practice in batch processing is that an operation is performed on multiple identical machines simultaneously through *lot splitting*, which leads to the shortening of its processing time. Furthermore, in some settings, an employee is linked to an order and has to follow it through all the workstations. Last but not least, preventive maintenance activities can also be taken into account in the detailed scheduling.

2.4 Conclusion

Limis' customers can use the scheduling function for employee, machine, and tool planning. They receive dynamically arriving customer orders that have a string-type or assembly-type routing, own workstations with one or multiple identical resources that have varying levels of daily capacity. The scheduling consists of two stages, namely, rough planning which is backward infinite loading, and detailed scheduling which is based on priority dispatching rules. A built-in or an external MRP function is used to generate stock orders.

3 Literature Review

In this chapter, we present a review of the literature that is related to the order acceptance problem. Section 3.1 provides a classification of manufacturing systems based on the production strategy and briefly discusses the hierarchical approach to production planning. Section 3.2 discusses different aspects of the order acceptance process. We have identified three streams of literature that are related to the order acceptance process and we proceed to discuss these topics. Sections 3.3, 3.4, and 3.5 discuss, respectively, due date assignment, acceptance/rejection, and capacity control. In Section 3.6, we present some common heuristics to solve optimization problems. In Section 3.7, we discuss the applicability of the order acceptance methods from the literature to Limis' customers. Section 3.8 provides the summary of the chapter.

3.1 Typology and planning of manufacturing systems

Companies follow different production strategies to efficiently fulfill customer demand. An important term in this context is the Order Penetration Point (OPP), also known as the Customer Order Decoupling Point (CODP), which is defined as the stage in the manufacturing process where a particular product is linked to a customer order (Olhager, 2003). In relation to different positions of OPP, 4 types of strategies are often distinguished: make-to-stock (MTS), assemble-to-order (ATO), make-to-order (MTO), engineer-to-order (ETO). MTS strategy is typical of consumer goods (i.e. electronics, food) industry, where consumer demand is satisfied off the shelf. At the other extreme is ETO, where the design of a one-of-a-kind product according to the customer specifications is performed first, followed by the procurement of materials, production, and delivery. MTO systems are characterized by a large variety of products manufactured entirely in response to firm customer orders. In such an environment, the design of a product is supplied by the customer for the first purchase and procurement of some materials can be based on forecast if they are common to multiple products. ATO systems can be seen as hybrid MTS/MTO systems, where manufacturing of components as well as procurement of materials are forecast-driven, while a large variety of end products are built from the components after customer orders are placed. This allows companies to avoid high stocks of end products while being able to fulfill customer demand in relatively short time. It is also worth noting that machine (or job) shops typically operate in an MTO or ATO environment (Zijm, 2000).

Although there are multiple product, production, and market-related factors impacting the choice of a production strategy or the position of the OPP, a major factor is the length of the production lead time in comparison with the delivery lead time (Olhager, 2003). Delivery lead time can be seen as the maximum time customers are willing to wait for the product to be delivered. In order to minimize inventory levels of components and end products, companies perform a number of the production steps related to a product in response to customer demand, similar to MTO, within the limits of the delivery lead time. Further reduction in the production lead time of post-OPP operations through efficient production planning and control may bring the company competitive advantage.

Traditionally, production planning and control has been carried out in a hierarchical fashion (Hendry & Kingsman, 1989). Production plans have been made for horizons of various lengths and with different levels of detail. A well-known example of such a system is MRP II (or Manufacturing Resources Planning) framework (Silver, Pyke, & Thomas, 2016). It starts with aggregate production planning for

groups of products with a horizon of up to 2 years into the future, where the time unit is typically a month. Following aggregate plans is the master production schedule which determines how much of each end product to make each time period (e.g. week, day) for a shorter horizon (e.g. 6 months). Master production schedules are further broken down to plans for individual components in a process called Material Requirements Planning. MRP II has been blamed for being material-oriented and lacking a finite capacity planning mechanism which is essential for an MTO environment (Giebels, 2000). Due to the limited predictability of customer demand in an MTO environment, capacity planning, particularly in the short term, is required to check the feasibility of dynamically arriving customer orders. Short-term capacity planning can be regarded as the last point to recognize the capacity problems as a result of incoming customer orders and plan the necessary capacity adjustments (i.e. overtime, hiring labor, subcontracting) so that the agreed-upon delivery dates can later be met during the detailed scheduling and execution of the production orders.

Giebels (2000) proposes a hierarchical planning framework for MTO and ETO environments, depicted in Figure 3.1. Long-term capacity planning refers to the decision making about strategic capacity investments such as purchasing new equipment and increasing staff levels. Tactical level is of particular importance as order acceptance is placed at this level. Resource loading refers to determining the time-phased capacity requirements of the accepted customer orders and planning capacity adjustments in the overloaded periods. It is also indicated that resource loading should also serve as a tool for order acceptance. This framework is based on the assumption that in MTO/ETO environments, it is not possible to work out detailed process plans (i.e. routings and processing times) in the order acceptance phase and therefore resource loading should be carried out on the basis of rough process plans. Detailed process plans are an input only to detailed scheduling of production orders at the operational level. In the order acceptance phase, orders consist of aggregate work packages which can be seen as a combination of operations that need to be processed on a group of resources. There are studies on acceptance/rejection decisions, capacity planning based on rough process plans (e.g. Ebben, Hans, and Weghuis (2005), Hans (2001)). However, since detailed process plans are available in the order acceptance phase for Limis' customers, we do not consider such studies in our literature review. For Limis' customers, order acceptance is placed at the operational level.

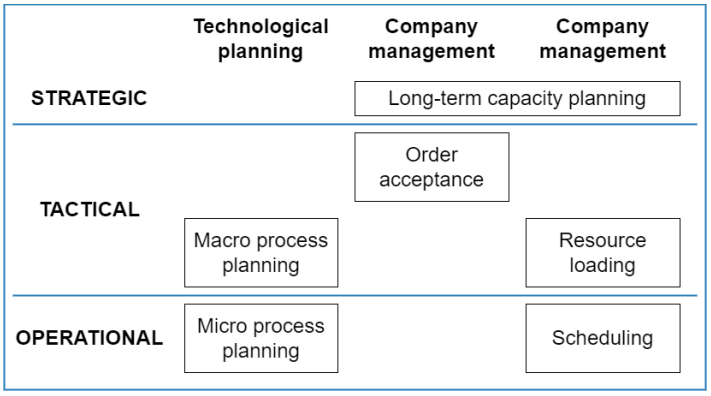


Figure 3.1: Positioning framework for MTO/ETO environments (from Giebels (2000))

3.2 Order acceptance

Order acceptance can be defined as a business process that may involve decisions such as acceptance/rejection, due date, quantity and price determination for incoming customer orders. Decisions on due date, quantity and price need to be made when they are not specified in the customer request or can be subject of negotiation. Order acceptance decisions can be made periodically for a group of incoming customer orders or in real time for each incoming customer order.

Due date and quantity determination is essential in an MTS environment in which prices are likely to be fixed because of generic end products and partial fulfillment of an order is not uncommon. In an MTO environment, due date and price for a customer order might be crucial in winning the order in MTO and ETO environments where customers might procure goods through competitive bidding (Kingsman, Hendry, Mercer, & de Souza, 1996). Partial fulfillment may not be possible in such systems since required quantities are typically small.

So, in an MTO system, companies may have to make a tradeoff between the cost and due date of a customer order. For example, use of additional capacity or delaying other orders for the purpose of minimizing the lead time may increase the costs associated with the order, and hence the price if the company does not decrease the profit margin. There a number of factors affecting the choice of a profit margin, such as the level of competition for the order, the company's need for work, and the company's relationship with the customer. The profitability of a price quote can be sacrificed if, for example, the customer is a strategic customer, or there likely to be repeat orders if the customer is satisfied with the first purchase (Kingsman et al., 1996). A rather implicit cost category is the long-term opportunity cost of having to reject more profitable orders in the future as a result of the high workload caused by accepting a less strategic order. Companies often avoid this cost by reserving a portion of their capacity, thus leading to a less competitive due date quote for the incoming order.

Order acceptance processes in different types of manufacturing systems vary in terms of not only the decisions they make but also the information they use to make decisions. In an MTS system, a customer order is fulfilled fully or partially in the case of sufficient stock of the required end product or the customer is promised a later delivery date based on planned future stock availability. This corresponds with the traditional available-to-promise (ATP) logic that is popular in MTS systems. ATP can be defined as "the uncommitted portion of a company's inventory and planned production, maintained in the master schedule to support customer order promising" (Ball, Chen, & Zhao, 2004). In contrast, a customer order is accepted based on material and capacity availability in an MTO environment. Therefore, a decision-making tool that can quickly analyze the impact of an incoming order on the existing production plan is important.

A recurring topic in the literature is related to the existence of functional silos between the marketing/sales and production planning departments of companies, which inhibits effective customer service. The problem stems from the conflicting objectives of the two functional departments. While the marketing department is interested in maximizing sales and therefore accepts orders with little or no input from the production environment, the production planning department wants to achieve high delivery performance through stable production plans. The result is a bad reputation due to poor delivery performance (Ashayeri & Selen, 2001; Ebben et al., 2005; Kingsman, Worden, Hendry, Mercer, & Wilson, 1993).

In line with the characteristics of Limis' customer, we review the literature on order acceptance in MTO or hybrid MTS/MTO job shops in which customer orders arrive dynamically and there is no information about customer orders until they arrive. In the following sections, orders have a string-type routing and consist of operations each of which requires uninterrupted processing on a certain resource for a certain duration unless stated otherwise.

3.3 Due date assignment

In due date assignment (DDA) models, companies solely assign or quote due dates to incoming orders in the absence of customer-requested due dates. The assumption is that customers always accept due dates quoted by the company. Also, capacity is considered to be fixed.

The problem can be seen as predicting the flow times of incoming orders. The release time r_i of an incoming order i is based on the arrival time of the order, the completion of any design work, and the availability of the required materials (Kingsman, Tatsiopoulos, & Hendry, 1989). We classify the DDA models into rule-based models and finite loading-based models. In the simulation studies investigating these models, orders arrive dynamically one at a time, assigned a due date d_i, and detailed scheduling is typically based on priority dispatching rules.

3.3.1 Rule-based due date assignment

Keskinocak and Tayur (2004) provide a comprehensive review of the rule-based DDA literature. The studies focus on proposing new DDA rules or investigating the performance of DDA rules under various operating conditions (i.e. priority dispatching rules, utilization levels) using measures such as average/standard deviation of lateness/tardiness/flow times, percentage of tardy orders. Here, we provide a list of the most commonly used rules and present some common conclusions from those studies according to Keskinocak and Tayur (2004).

The following rules determine a flow time estimate f_i for an incoming order i based on a measure of the shop status at the arrival time and/or an order characteristic. Order i is assigned a due date found by adding its flow time estimate to to its release time.

$Slack\ (SLK):\ f_i = p_i + k_1$
$Total\ work\ (TWK):\ f_i = k_1 p_i$
$Number\ of\ operations\ (NOP):\ f_i = k_1 n_i$
$(TWK + NOP):\ f_i = k_1 p_i + k_2 n_i$
$Jobs\ in\ the\ system\ (JIS):\ f_i = k_1 p_i + k_2 JIS$
$Workload\ in\ the\ system\ (WIS):\ f_i = k_1 p_i + k_2 WIS$
$Jobs\ in\ queue\ (JIQ):\ f_i = k_1 p_i + k_2 JIQ$
$Workload\ in\ queue\ (WIQ):\ f_i = k_1 p_i + k_2 WIQ$

$f_i = flow\ time\ estimate\ of\ order\ i$
$p_i = total\ processing\ time\ of\ order\ i, p_i = \sum_{j=1}^{n_i} p_{ij}$
$p_{ij} = processing\ time\ of\ operation\ j\ of\ order\ i$

$n_i = number\ of\ operations\ of\ order\ i$
$JIS = number\ of\ orders\ at\ the\ shop$
$WIS = total\ (remaining)\ workload\ of\ all\ the\ orders\ at\ the\ shop$
$JIQ = number\ of\ orders\ waiting\ in\ the\ queues\ of\ the\ workstations\ that\ are\ in\ the\ routing\ of\ order\ i$
$WIQ = total\ (remaining)\ workload\ of\ the\ orders\ waiting\ in\ the\ queues\ of\ the\ workstations\ that\ are\ in\ the\ routing\ of\ order\ i$
$k_1,\ k_2 =\ parameters$

p_i can be increased to account for the sequence-independent setup times of the operations. Parameters k_1 and k_2 are empirically determined. For rules that consider both an order characteristic and shop status information (e.g. JIS, WIQ), regression can be used to link the corresponding order characteristic and shop status measure to order flow times observed in a pilot simulation run and determine the parameters (Thürer, Stevenson, Silva, & Land, 2013).

Below are some common conclusions from the literature:

- DDA rules that consider both an order characteristic and shop status information (e.g. JIS, WIQ) in general perform better than the rules that only consider an order characteristic (e.g. TWK, NOP).
- The performance of a DDA rule also depends on the priority dispatching rule. Due date-based priority dispatching rules (e.g. EDD) perform better than other rules such as SPT, in combination with DDA rules that are based on both an order characteristic and shop status information (e.g. JIS, WIQ).
- The performance of a DDA rule is also affected by the differences in the utilization levels of the shop. Unbalanced shops lead to the deterioration of due date performance, due to the increased system variability.

3.3.2 Due date assignment based on finite loading

Intuitively, information on time-phased capacity availability of resources would be a valuable input for due date assignment. In finite loading, the rolling time horizon is typically divided into T discrete equal-length time periods. In the studies, the operations of an incoming order i are loaded on top of the workload of the existing orders in the load profiles of the corresponding resources taking into account capacity availability and the precedence relations. In other words, the existing orders are not reloaded when a new order arrives. WL_{kt} (CWL_{kt}) represents the (cumulative) workload of resource k in period t, while C_{kt} (CC_{kt}) represents the (cumulative) capacity of resource k in period t. In some finite studies, WL_{kt} (CWL_{kt}) is allowed to be less than or equal to a percentage CLL (i.e. capacity loading limit) of C_{kt} (CC_{kt}), where CLL can be higher or lower than 100%. Each operation j of order i is assigned a due time d_{ij} and the due date of order i is set to the date in which the due time of the last operation falls. Some studies also use a minimum waiting time MWT for each operation as explained below. CLL and MWT are empirically determined parameters meant to improve the accuracy of assigned due dates. In order to ensure that capacity is consumed as planned by the finite loading method, detailed scheduling in the studies is based on the ODD priority dispatching rule, as explained in Section 2.3.2, which means detailed schedules are non-delay schedules.

Thürer et al. (2013) present a forward finite loading (**FFL**) method in which each operation is loaded fully in a single time period. To load operation j of an incoming order i on the required resource l, we first find the period h in which the due time of the predecessor plus the processing time of the operation $d_{i,j-1} + p_{ij}$ falls. If there is enough capacity to load the full workload of the operation in period t', i.e. $WL_{lh} + p_{ij} \leq CLL * C_{lh}$, the operation is loaded in that time period. Otherwise, the next period is considered until a period in which the operation can be fully loaded is found. d_{ij} is set to the end of the time period in which the operation is loaded. d_{i0} is equal to 0, i.e. the release time of the order. Figure 3.2 shows how an operation with $p_{ij} = 5$ is loaded according to FFL, where $CLL = 100\%$. In the figure, although $d_{i,j-1} + p_{ij}$ falls in period 3, the operation cannot be fully loaded in periods 3 and 4, and therefore it is loaded in period 5.

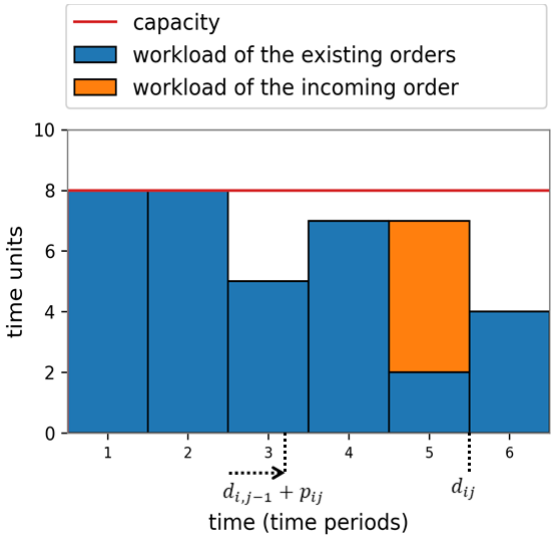


Figure 3.2: Forward finite loading (adapted from Thürer et al. (2013))

Consider the resource depicted in 3.2 and assume that the workload in period 4 arrives at the resource in period 4 and no other order arrives at the shop until period 4. The operation depicted in Figure 3.2 will be scheduled to start in period 3 and finish in period 4, since it arrives at the resource in period 3. As a result, some workload in period 4 will be performed in period 5. This means that when FFL is used some operations will start early delaying other operations, unless the workload in each period does not fill up the capacity. In FFL, setting CLL properly can help to reduce deviations from operation due dates and therefore improve the accuracy of assigned order due dates, as done by Thürer et al. (2013).

Note that in FFL operations are not allowed to be partially loaded in multiple time periods. Therefore, the length of a time period must be no smaller than the maximum operation processing time. Moreover, the due time of each operation is set to the end of the time period in which it is loaded since it is not known when the operation will be performed during the period. This means operations with a very small workload are also planned to take at least 1 time period, which may cause overestimation of order due dates.

When a new order arrives, the workload of the existing orders should reflect the deviations on the shop floor. Thürer et al. (2013) differentiate between positive and negative backlog. Positive backlog is distributed over the time periods starting from the first time period, while negative backlog is removed from the corresponding time periods.

Bertrand (1983) proposes a cumulative forward finite loading (**CFFL**) where cumulative workload and cumulative capacity of resources are considered. To load operation j of an incoming order i on the required resource l, we first find the period h in which the preliminary due time d'_{ij} falls, where $d'_{ij} = d_{i,j-1} + p_{ij} + MWT$. Starting with period h, we search for the earliest period $q \geq h$ in which and all the subsequent periods we can fully load the operation, i.e. the following inequalities hold $CWL_{lm} + p_{ij} \leq CLL * CC_{lm}$, where $m = q, q+1, ..., T$. Once q is found, the operation is fully loaded in periods $m = q, q+1, ..., T$. If q equals h, the operation is expected to finish at the preliminary due time, and therefore d_{ij} is set to d'_{ij}. Otherwise, the operation is expected to finish somewhere in period q, and therefore d_{ij} is arbitrarily set to the beginning of period q plus 0.25 times the period length. d_{i0} is equal to 0, i.e. the release time of the order. Figure 3.3 shows how the starting operation with $p_{i1} = 5$ is loaded according to CFFL, where $CLL = 100\%$ and $MWT = 4$. In Figure 3.3a, although d'_{i1} falls in period 2, because the operation cannot be fully loaded in period 2 it is loaded starting from period 3. In Figure 3.3b, although d'_{i1} falls in period 2 and the operation can be loaded in period 2, because the operation cannot be fully loaded in period 3 it is loaded starting from period 4.

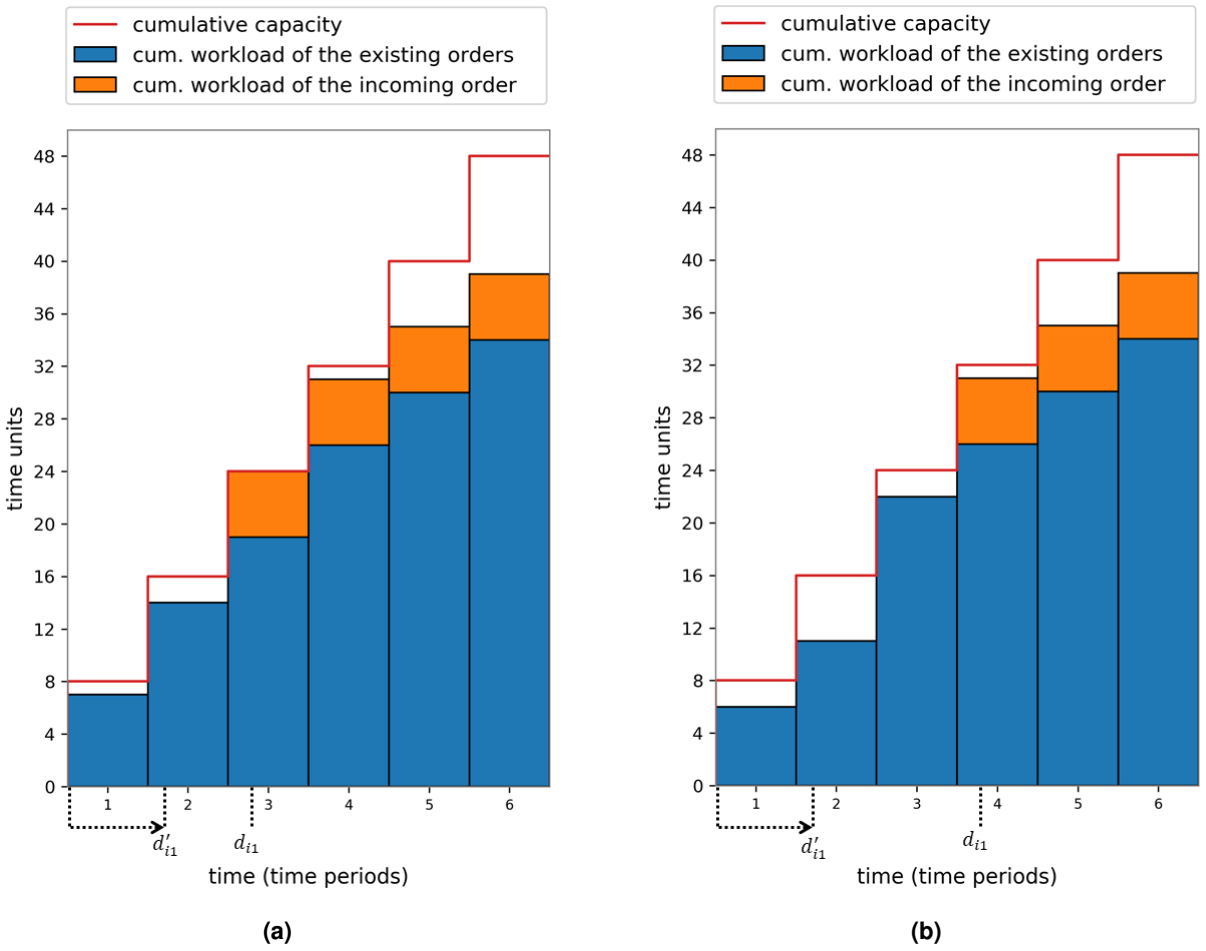


Figure 3.3: Cumulative forward finite loading (adapted from Bertrand (1983))

In CFFL, an operation is loaded starting from the due period (i.e. the period in which the due time falls). Because cumulative workload and cumulative capacity are considered, an operation can consume capacity available in any of the earlier periods. Consider the situation depicted in Figure 3.3a and assume that the workload that is due period 2 arrives at the resource in period 2. The operation

depicted in the figure will be scheduled to start in period 1 and finish in period 2, since it arrives at the resource in period 1. As a result, some workload due period 2 will be performed in period 3. This means that when CFFL is used some operations will start early delaying other operations, similar to FFL. However, in contrast with FFL where the inaccuracy results from pieces of unplanned capacity, in CFFL inaccuracy results from the fact that operations can consume capacity from the periods earlier than their due period. Setting CLL and MWT properly can help to reduce deviations from operation due dates and therefore improve accuracy of assigned order due dates.

Bertrand (1983) sets MWT as a multiple α of the mean operation processing time, where α is an empirically determined parameter similar to CLL. Static and dynamic choices for α are made, where the dynamic α depends on the (remaining) workload of the existing orders in the entire shop. Moreover, the period length in CFFL is not required to be longer than the maximum operation processing time. It is suggested that the choice of a period length be in relation to the mean operation processing time and it is arbitrarily set to 8 times the mean operation processing time.

Bertrand (1983) uses the average and standard deviation of lateness as measures to compare the performances of the proposed CFFL and TWK+NOP rule in a simulation study of a job shop. It is shown that the use of CFFL decreases the standard deviation of lateness, compared to the TWK+NOP rule. Thürer et al. (2013) show that CFFL outperforms FFL in terms of average flow time and standard deviation of lateness.

Robinson and Moses (2006) present a forward loading method in which operations are allowed to be partially loaded (**FFLPL**) and analyze the effect of the period length on the accuracy of assigned due dates and computational time. A period length is referred to as granularity G. To load operation j of an incoming order i on the required resource l, we first find the number of periods required b_{ij} as follows.

$$b_{ij} = \left\lceil \frac{p_{ij}}{G} \right\rceil$$

If $b_{ij} = 1$, starting with the period h that follows the due period of the predecessor, we search for the earliest single period $q \geq h$ in which we can fully load the operation, i.e. $WL_{lq} + p_{ij} \leq C_{lq}$. If $b_{ij} \geq 1$, starting with the period h that follows the due period of the predecessor, we search for the earliest period $q \geq h$ which has some available capacity and the following $b_{ij} - 1$ periods are fully available, i.e. $WL_{lq} < C_{lq}$, $WL_{l,q+1} = 0$, and $WL_{l,q+2} = 0$. Once q is found, we load periods $q+1$ and $q+2$ fully and the remaining workload in period q. Figure 3.4 shows how the first two operations with $p_{i1} = 5$ and $p_{i2} = 17$ are loaded according to FFLPL. The first operation can be fully loaded in period 1, as shown in Figure 3.4a, which means the second operation can be loaded starting from period 2. However, because period 3 is not fully available, the second operation is loaded in periods 3, 4, and 5, as shown in Figure 3.4b.

FFLPL prevents the plan for an operation from being overly fragmented across a large number of time periods. Similar to FFL, there might be pieces of unplanned capacity, referred to as holes in the study, causing deviations from operation due dates. Robinson and Moses (2006) use the median absolute lateness as a measure. The results show that there exists a certain range of moderate granularities which leads to both high accuracy and low computational time. The reason behind the high level of accuracy is that a moderate size period length is coarse enough to accommodate multiple operations leading to fewer holes, and it is fine enough to better estimate due dates.

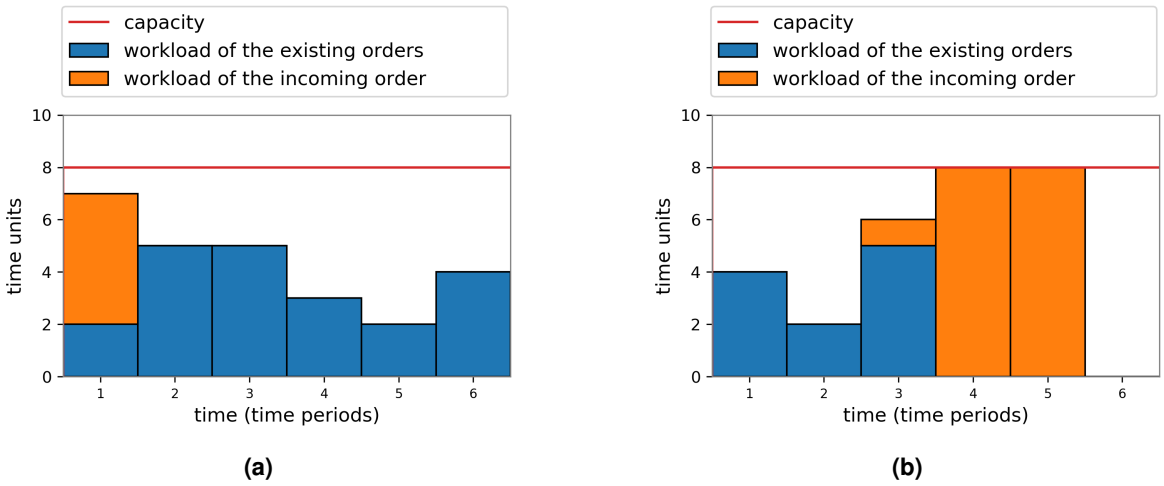


Figure 3.4: Forward finite loading with partial loading (adapted from Robinson and Moses (2006))

3.4 Acceptance/rejection

Another stream of literature is concerned with accepting or rejecting incoming orders in the presence of customer-requested due dates. We classify acceptance/rejection (A/R) models based on due date flexibility and capacity flexibility.

3.4.1 Acceptance/rejection with due date flexibility

In this type of A/R models, tardiness is allowed while capacity is assumed to be fixed.

Philipoom and Fry (1992) propose two similar rules for making A/R decisions for dynamically arriving orders. In the first rule, an incoming order is accepted if the total (remaining) workload of all the existing orders plus the total workload of the incoming order is below a predetermined level. In the second rule, an incoming order is accepted if it does not overload any machine in its routing. More precisely, for each machine in the routing, the sum of the current workload of the machine plus the workload of the incoming order on that machine should be less than a predetermined level, where the current load of a machine is the sum of the workload in its queue and the operation processing times of the existing orders that need to visit the machine later. In either rule, lowering the threshold decreases the utilization level of the shop. The threshold can be empirically chosen to achieve a certain service level (i.e. near-zero percentage of tardy orders) with the maximum possible utilization level.

The proposed rules do not take into account due dates. In the study, every incoming order is assumed to have the same due date allowance (i.e. the difference between the due date and the arrival date) of 12 days. In reality, however, due date allowances may largely differ and using these rules means that an incoming order with a loose due date relative to its workload may be rejected. The rules are based on the well-known concept of order release or input control, which aims at controlling the work-in-process on the shop floor with the belief that constant workload leads to more stable order flow times and higher delivery performance. Philipoom and Fry (1992) conclude that the second rule outperforms

the first rule in terms of measures such as the average tardiness and the percentage of tardy orders.

Nandi and Rogers (2004) take a rescheduling-based approach to make A/R decisions for dynamically arriving orders. Detailed scheduling is based on a priority dispatching rule, least slack per remaining operation, where the slack of an order is its due date allowance minus its total workload. Each accepted order contributes an amount equal to its revenue less any tardiness cost upon completion. The revenue of an order is linearly proportional to its total workload, while the tardiness cost is linearly proportional to the product of its tardiness and its total workload. When a new order arrives, two new detailed schedules are built, first one including the incoming order and the other without the incoming order. If the total net profit of the first schedule exceeds that of the second schedule by a predetermined proportion (i.e. the control parameter) of the order's revenue, the order is accepted.

In the simulation study, the key performance measure is referred to as the overall percentage achievement OPA which is defined as the ratio of the actual net profit to the maximum possible revenue if all orders were accepted and completed on time. It is shown that given the environmental conditions (i.e. order arrival rate, due date tightness, etc.) there is a value of the control parameter for which OPA is maximized. In practice, using a profit-based performance measure like OPA and the proposed profit-based A/R model described above can cause long delays to individual customer orders.

3.4.2 Acceptance/rejection with capacity flexibility

In this type of A/R models, capacity can be increased, for example, by means of overtime while due dates are assumed to be fixed.

Mestry, Damodaran, and Chen (2011) propose a mixed-integer linear program (MILP) to optimally decide which orders to accept from a set of orders with fixed due dates and schedule the accepted orders using regular capacity and overtime. Unlike the DDA and A/R models discussed so far, preemption of operations is allowed here. Moreover, orders are assumed to have a string-type routing. Although the model is developed for static order arrivals, it can also be applied for dynamically arriving orders where rejection is not an option for existing orders. The objective is to maximize the net profit, i.e. the revenues of accepted orders less the cost of using resources in regular time and overtime. It is shown that solving problem instances of small and moderate sizes with a branch-and-bound method available in commercial MILP solvers requires a significant amount of time. For example, it takes 10000 seconds for solving a problem instance with 5 orders each having 8 operations with 5% optimality gap. Mestry et al. (2011) also propose an exact and approximate branch-and-price strategies to solve the problem and show that both methods outperform the commercial solver in terms of both solution time and solution quality.

Akkan (1996) studies the problem of inserting an incoming order with a fixed due date into the existing detailed schedule. Orders are assumed to have a string-type routing. An incoming order is first inserted using backward insertion in which each operation of the incoming order is inserted as late as possible given the order due date and the precedence relations. If the insertion is successful, the order is accepted. Otherwise, the order is inserted using overtime, which means activating the so-called offline segments of the timelines. The operations of existing orders on a schedule are not allowed to be started earlier or completed later, which means only the operations that have been scheduled to start before and finish after an offline segment can be adjusted in order to avoid preemption if needed. The

unit cost of overtime is assumed to be a stepwise decreasing function of time, while the periods of overtime that should not be used unless it is absolutely necessary are given an extremely high cost. The problem is seen as inserting blocks (i.e. the operations of an incoming order) into the schedules of the corresponding resources. A domination rule is established to obtain a reasonable number of blocks that are non-dominated for each operation. Based on this notion of non-dominated blocks, the problem is formulated as a shortest path problem. However, due to the large size of the problem in a practical job shop setting, optimal solutions are computationally expensive. Thus, three heuristic methods are proposed and they are shown to perform reasonably well in terms of the total cost of overtime.

3.4.3 Acceptance/rejection without due date and capacity flexibility

In this type of A/R models, both capacity and due dates are assumed to be fixed.

Roundy et al. (2005) propose a rescheduling-based approach to make A/R decisions for dynamically arriving orders. An incoming order is accepted, if a new schedule including the incoming order and respecting all the due dates can be built. The scheduling problem is decomposed into simple single-machine problems by assuming that each machine has a constant lead time, which allows to assign non-overlapping time windows (i.e. release and due dates) for operations and thus consider each machine in isolation. Another important assumption is that all operation start times and processing times are assumed to be an integral number of hours. Existing operations are allowed to be moved within their time windows to make room for a new operation, which does not cause any tardiness in orders. An incoming order for a product includes multiple shipment dates and quantities, thus requiring lot sizing decisions as well. As a result, each customer order needs to be inserted as a series of production orders with different quantities and due dates. The problem is formulated as a series of network flow problems, one for each machine. Because of the difficulty of solving the integer linear program (ILP) in practical settings, different heuristic methods are used for making different decisions. First, a simple search is performed to find the optimal number of production orders (or batches). Next, a heuristic search for the due dates of those production orders is made through one of the 5 methods (4 methods from the literature and 1 novel method), i.e. tabu search (TS), simulated annealing (SA), genetic algorithm (GA), and randomized local search (RLS), single-machine heuristic. Each of these methods calls the common subroutine to find the optimal batch sizes and the optimal cost. Essentially, the five methods are compared with each other and GA, SA, and single-machine heuristics are shown to be promising.

3.5 Capacity control

Thürer et al. (2013) use forward and backward finite loading methods, including FFL and CFFL described in Section 3.3.2, and another rule-based method from the literature to decide if and on which resources capacity needs to be increased upon the arrival of a new order. All incoming orders with customer-requested due dates are assumed to be accepted.

The procedure for FFL works as follows. An incoming order is loaded as described in Section 3.3.2. If the customer-requested due date can be met, the procedure stops and no capacity adjustment is made. Otherwise, capacity adjustment is done in a resource where the order had the longest

operation throughput time as calculated by FFL, assuming that that the resource is most likely to be the bottleneck. Capacity is first increased in the first period. If an adjustment has already been done in that time period, the capacity in the next period is increased. If no time period without a previous adjustment is found until the due date of the operation, the procedure stops and the order is accepted without meeting the customer-requested due date. If an adjustment can be made in a time period, the order is reloaded and the above steps are repeated. Capacity adjustment in a period is done by decreasing the processing times of the operations loaded in that period by a predetermined percentage. It is not clear, for example for FFL, how adjusting the capacity in the first period on a resource helps to advance an operation that arrives at the resource in a later period because of the timing of its predecessors. In the simulation study, it is shown that CFFL outperforms the other finite loading methods and the rule-based method outperforms all the finite loading methods in terms of the average tardiness and the percentage of tardy orders.

3.6 Heuristics

In some of the studies discussed so far (e.g. Mestry et al. (2011)), the problem is formulated as an optimization problem. Solution methods to optimization problems, particularly to those involving integer decision variables, are classified into *exact* and *approximate* methods. Exact methods such as branch-and-bound are guaranteed to find optimal solutions but might require exponential computational time in the worst case. Approximate methods or heuristics are designed to find good solutions in a considerably less amount of time (Blum & Roli, 2003).

Heuristics can be divided into *constructive* and *improvement heuristics*. Constructive heuristics generate typically low-quality solutions from scratch, whereas improvement heuristics seek to find a near optimal solution starting from an initial solution or a set of initial solutions. Constructive heuristics add components to an initially empty solution until it is complete. They can be *static* or *dynamic*. A static constructive heuristic assign priorities to solution components once beforehand, whereas a dynamic constructive heuristic reassigns priorities to components during the construction (Blum & Roli, 2003). In Sections 3.6.1 and 3.6.2, we present two classes of improvement heuristics, *local search* methods and *metaheuristics* respectively.

3.6.1 Local Search

The basic class of improvement heuristics consists of local search methods. Starting with an initial solution, in each iteration the current solution s is replaced with a better or an improving solution in its *neighborhood* $N(s)$ which is a set of solutions assigned to s by a function referred to as a *neighborhood structure*. A solution s' in $N(s)$ is referred to as a *neighbor* of s and a choice of s' from $N(s)$ is referred to as a *move* (Blum & Roli, 2003). Because only improving moves are performed, in each iteration the current solution s is also the best solution so far s^*. In optimization problems where a solution can be represented with a permutation, two popular neighborhood structures are *swap* and *insertion* (Deroussi, Gourgand, & Norre, 2006). The swap neighborhood consists of the permutations (i.e. solutions) that can be obtained by swapping two elements of a given permutation. The insertion neighborhood consists of the permutations that can be obtained by moving an element from its current position to another position. Figure 3.5 depicts examples of a swap and insertion move. The size of

the swap neighborhood is $\frac{n(n-1)}{2}$, while the size of the insert neighborhood is $(n-1)^2$, where n is the size of a permutation.

Figure 3.5: Example of (a) insertion move, (b) swap move

A neighborhood structure is considered *connected* if it is possible to construct a finite sequence of moves leading from an arbitrary solution to any other solution (Brucker, Hurink, & Werner, 1996). This property implies the possibility of a heuristic based on a connected neighborhood structure to reach a globally optimal solution. Local search methods differ in their strategies to select a neighbor solution $s' \in N(s)$. In *best improvement* strategy, the entire neighborhood of a solution is evaluated and the solution is replaced by its best improving neighbor. In *first improvement*, the exploration of the neighborhood stops as soon as an improving neighbor solution is found. The search over the neighborhood can be done randomly or in an ordered manner. In either case, first improvement strategy is typically less time-consuming than best improvement. The performance of a local search method is impacted by the initial solution, the neighborhood structure, and the neighborhood search strategy (Feo & Resende, 1995). Local search methods terminate when no improving solution is found in the neighborhood of the current solution which is referred to as a *locally optimal* solution. The major drawback of local search methods is that they often end up with locally optimal solutions that are not necessarily *globally optimal*. (Blum & Roli, 2003).

3.6.2 Metaheuristics

Metaheuristics are designed to increase the likelihood of obtaining optimal or near-optimal solutions. Two interrelated concepts in the context of improvement heuristics are *intensification* and *diversification*. The former refers to further exploration of promising areas of the search space, while the latter refers to exploration of unexplored regions of the search space. Local search methods discussed in Section 3.6.1 focus on intensification, whereas the strength of a metaheuristic lies in its ability to achieve a balance between intensification and diversification (Blum & Roli, 2003).

We present four metaheuristics that have found numerous successful applications in scheduling problems, particularly in job shop scheduling and permutation flow shop scheduling (Framinan, Gupta, & Leisten, 2004; Jain & Meeran, 1999). Greedy Randomized Adaptive Search Procedure (GRASP), Tabu Search (TS), and Simulated Annealing (SA) belong to the class of *single solution-based* or local search-based metaheuristics that maintain a single solution in each iteration, Genetic Algorithms (GAs) belong to the class of *population-based* metaheuristics that maintain a population of solutions in each iteration.

GRASP

GRASP is a metaheuristic proposed by Feo and Resende (1995) and it combines a constructive heuristic and a local search method. In each iteration, an initial solution is built using a dynamic constructive heuristic involving randomness and the initial solution is improved using a local search method. A

hybrid metaheuristic can be built by using another metaheuristic instead of a local search method. In the solution construction phase, solution components are added to a partial solution one at a time and the next component to add is chosen randomly from a subset of all possible components. The subset is referred to as the *restricted candidate list* and is composed of the best α components based on their dynamically assigned priorities. α can be static or it can be changed in each iteration randomly or deterministically. The typical stopping criterion for GRASP is a maximum number of iterations.

Tabu Search

TS is a memory-based metaheuristic proposed by Glover (1986). In contrast with local search methods, TS accepts the best neighbor in each iteration even if it is worse than the current solution. As this may cause *cycling*, which is visiting a series of solutions repeatedly, the information on the recently visited solutions is stored in a *tabu list* T to ensure that those solutions are not revisited for a period of the *tabu tenure* (i.e. tabu list size $|T|$). Because storing complete solutions in T and comparing each neighbor with each solution in T is not efficient, often solution attributes are stored. For the swap and insertion neighborhoods mentioned above, a tabu list can be defined in various ways. Table 3.1 shows some types of T and the corresponding *tabu classifications*.

Table 3.1: Examples of a tabu list definition and the corresponding tabu classification

	Attributes stored in T	**Tabu classification of moves**
Swap neighborhood		
1	the pair of elements chosen to be swapped	A move that swaps a pair of elements in T is tabu
2	a tuple of two pairs each consisting of an element chosen to be swapped and its position before the swap	A move that returns both elements of a tuple in T to their corresponding positions is tabu
3	two pairs each consisting of an element chosen to be swapped and its position before the swap	A move that returns an element in T to its corresponding position is tabu
Insertion neighborhood		
1	the element chosen to be moved	A move that moves an element in T is tabu
2	the element chosen to be moved and its position before the move	A move that returns an element in T to its corresponding position is tabu

As a result of those tabu classifications, some solutions that have not been visited are also forbidden and among those solutions can be optimal solutions. To overcome this problem, a common *aspiration criterion* is used which allows a tabu move if it leads to a solution better than the best one so far. Another aspiration criterion allows a tabu move(s) connected with the oldest element in T when all possible moves are classified as tabu. TS in its basic form includes only *short-term memory* in the form of tabu lists. As depicted in Figure 3.6, the basic TS uses best improvement strategy over a *modified neighborhood* $N^*(s)$ which consists of the neighbor solutions that are not tabu or are tabu but meet the aspiration criterion.

Tabu lists are cyclical lists, which means that when a list is full the oldest element(s) are removed in order to add a new element(s). The choice of a tabu tenure depends on the restrictiveness of the tabu classification. For example, in Table 3.1, the third tabu classification is more restrictive than the second one and therefore requires a smaller tabu tenure so that not all possible moves are classified as tabu after a number of iterations. Moreover, a small tabu tenure means a high level of intensification and

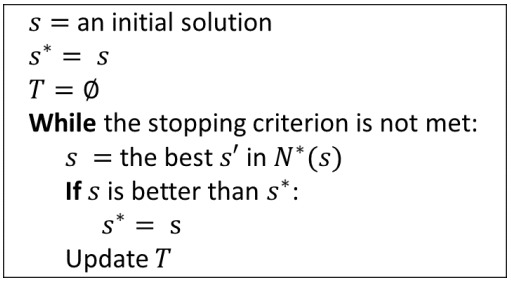


Figure 3.6: Tabu Search algorithm (adapted from Blum and Roli (2003))

might fail to prevent cycling, while a large tabu tenure means a high level of diversification and might decrease solution quality. In more advanced TS applications, a dynamic tabu tenure can be used which varies to guide the search toward intensification or diversification. Possible stopping criteria for TS are the maximum number of iterations, the maximum number of iterations since the best solution last changed, etc. For example, if the maximum number of iterations is used as the stopping criterion, the choice of its value entails a tradeoff between solution quality and solution time.

Simulated Annealing

SA is another local search-based metaheuristic proposed by Kirkpatrick, Gelatt, and Vecchi (1983). It is named so because the algorithm is analogous to the annealing process of solids. Similar to TS, SA accepts non-improving moves but with a certain probability whereas improving moves are always accepted. In each iteration, a move (or a neighbor solution) is randomly selected and if the move is non-improving the probability of acceptance is typically calculated according to the *Boltzmann distribution* $exp\left(-\frac{|f(s')-f(s)|}{T}\right)$, where $f(s)$ is the objective value of solution s and T is the so-called *temperature* parameter (Blum & Roli, 2003). T is decreased during the search and therefore the probability of accepting worse solutions decreases toward the end of the search. This means that SA behaves almost like a random search at the beginning leading to a high level of diversification and resembles a local search at the end leading to a high level of intensification. As can be seen from Figure 3.7, four parameters need to be specified for the SA algorithm:

- an initial temperature
- $Markov\ chain\ length$ or the number of iterations at a certain temperature
- a rule for updating T
- a stopping criterion

A set of choices for these parameters is referred to as a *cooling schedule*. A wide range of cooling schedules are used in the literature. We present some common ways to set up a cooling schedule (van Laarhoven & Aarts, 1987). An initial temperature can be set to achieve a desired *acceptance ratio*, which is the ratio of the accepted moves to the number of proposed moves, in trial runs. The length of a Markov chain can be static or dynamic during the search. A static length can be set as a percentage of the neighborhood size. As for a dynamic one, a Markov chain can be terminated as soon as an improving solution is found. A typical rule for updating T is to multiply it by a constant between 0 and 1 after each Markov chain. Many studies use a constant in the range of 0.5 and 0.99. There is a tradeoff between this constant and the Markov chain length. A stopping criterion can be a final temperature which is usually set close to 0, or terminating SA when the solution reached at the end of a Markov chain does not change for a number of chains. In general, finetuning a cooling schedule entails a

tradeoff between solution quality and solution time.

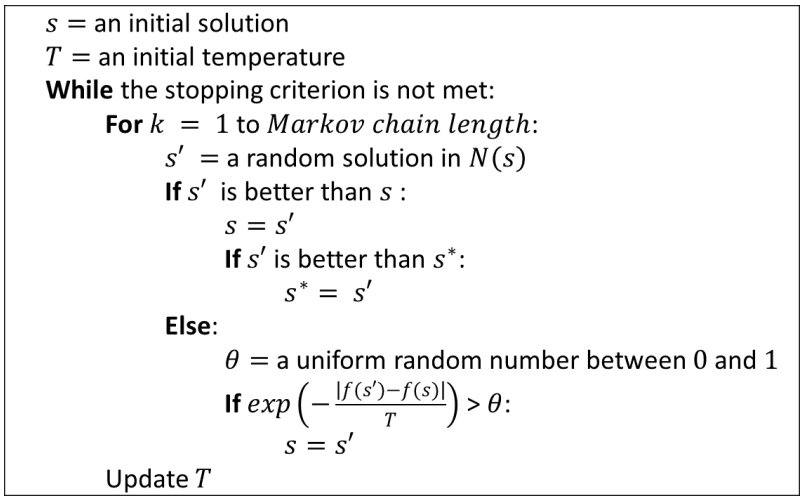


Figure 3.7: Simulated Annealing algorithm (adapted from Blum and Roli (2003))

Genetic Algorithms

GAs are proposed by Holland (1975) to simulate the biological evolution of natural species. It was later applied to optimization problems. In GAs, there is an evolving *population* P (i.e. set) of m *individuals* (i.e. solutions).

An initial population is usually generated randomly. A new population P_n is generated by repeatedly selecting a pair of individuals from the current population as *parents* and applying *crossover* and/or *mutation* operations on them to produce *offspring* (i.e. a new pair of individuals). A common approach for selecting parents is the *roulette wheel* method in which the probability of randomly selecting an individual s is proportional to its *fitness value* $g(s)$. The fitness value of an individual can be defined as the ratio of its objective value to the average objective value of the individuals in the population. A crossover operator combines parts of two parents to produce two new individuals and it is applied to each pair of parents with a certain probability p_c. A mutation operator is used to randomly modify an individual and it is also applied with a certain probability p_m to either the offspring or the parents themselves depending on whether crossover was applied. For permutation-based problems, swap and insertion operators can be used as a mutation operator. An example of a crossover operator for those problems is *partially mapped crossover* (PMX) depicted in Figure 3.8. This operator first randomly selects two cut points on both parents. In order to create an offspring, the sub-string between the two cut points in one parent replaces the corresponding sub-string in the other parent. Then, inverse replacement is applied outside of the cut points, in order to eliminate duplicates. In Figure 3.8, offspring 1 is created by first replacing the sub-string 236 in parent 2 by the sub-string 564. Hence, 5 replaces 2, 6 replaces 3, and 4 replaces 6 (step 1). Since 4 and 5 are now duplicated in the offspring, inverse replacement is applied outside of the cut points. Namely, 2 replaces 5, and 3 replaces 4 (step 2). In the latter case, 6 first replaces 4, but since 6 is already found in the offspring at position 4, 3 finally replaces 6. Multiple replacements at a given position occur when an element is located between the cut points on both parents, like 6 in this example.

Figure 3.9 depicts the traditional GA algorithm in which a new population totally replaces the current population in each iteration (Potvin, 1996). A stopping criterion can be the number of iterations or a maximum solution time.

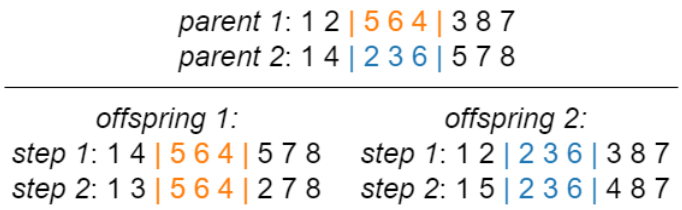


Figure 3.8: Partially mapped crossover (adapted from Potvin (1996))

P = an initial population
Evaluate the fitness value $g(s)$ of every individual in P
While the stopping criterion is not met:
    $P_n = \emptyset$
    While $|P_n| < m$:
        (s_1, s_2) = a pair of individuals selected as parents from P
        With p_c, apply the crossover operator to (s_1, s_2) and replace them with the offspring
        With p_m, apply the mutation operator to s_1 and s_2 and replace them with the offspring
        Evaluate $g(s_1)$ and $g(s_2)$
        $P_n = P_n \cup \{s_1, s_2\}$
    $P = P_n$

Figure 3.9: Genetic Algorithm (adapted from Potvin (1996))

3.7 Discussion

The literature review shows that studies in an MTO or hybrid MTS/MTO system do not typically take into account material availability and assume that materials are continuously available. Furthermore, we have not found a study considering fulfillment of customer orders by partial shipments in such systems. Two of the studies considering capacity flexibility (Mestry et al. (2011); Thürer and Stevenson (2020)) use a discrete-time representation of resources' capacity while one study (Akkan (1996)) focuses on inserting incoming orders into the detailed schedule in continuous time.

Akkan (1996) assumes orders arrive one at a time. The main limitation of the approach is that when a new order arrives the existing orders are not rescheduled. So, high priority (rush) orders arriving during a period of high workload may need to be delayed significantly while the existing orders that are not rescheduled are completed too early, which might result in the loss of high priority customers. Not rescheduling existing orders also prevents us from using the due date flexibility of some existing orders in the case of Limis' customers.

The MILP of Mestry et al. (2011) does not assume orders arriving one at a time and reschedules the existing orders, although it entails other limiting assumptions, i.e. operations can be preempted, orders have a string-type routing. The assumption that operations can be preempted enables them to create detailed schedules using a discrete-time representation of resources' capacity. Another important drawback is that solution times can grow exponentially with the size of a problem instance. The size of a problem instance is determined by various factors, namely the number of orders, the number of operations of each order, the number of time periods (i.e. the length of horizon), the number of resources. The problem instance can be large enough to result in a long solution time for Limis' customers. As mentioned in Section 2.2, in the chosen Limis' customer the number of incoming orders

can be as high as 25 and the average number of operations per order is around 7 for assembly-type routings and around 3 for string-type routings. For comparison, Mestry et al. (2011) use a commercial solver to solve the MILP and show that it takes 10000 seconds for solving a problem instance with 5 orders each having 8 operations with 5% optimality gap. Thus, it seems impossible to use an MILP-based approach as a fast solution method for Limis' customers.

Thürer and Stevenson (2020) do not replan (reload) the existing orders upon the arrival of a new order and therefore this approach suffers from the same drawback as that of Akkan (1996). However, the finite loading methods used by Thürer and Stevenson (2020) can be used to replan the existing orders when a new order arrives. It is important to note that in the literature finite loading methods are not used to solve a planning problem. They are implemented like control policies or procedures whose parameters are determined through simulation experiments. The output mainly consists of assigned due dates for operations. Operation due dates are used in dispatching. Mainly due to the non-delay dispatching of operations, some operations are completed too early making others late. As a result, orders are not completed later than expected. The parameters of a finite loading method are tuned to improve mainly tardiness performance.

Given that detailed scheduling of Limis Planner is based on priority dispatching rules, finite loading methods are a suitable approach for Limis' customers. In addition, they are simple and therefore fast methods. We propose to use a finite loading method and replan the existing orders when a new customer request arrives. Regardless of whether we replan the existing orders or not, we need a method to determine a sequence of orders since an incoming customer request may include multiple orders. Heuristics discussed in Section 3.6 are a promising method to determine a sequence of orders. So, we propose to use a finite loading method in combination with heuristics.

3.8 Conclusion

We focus on the methods that consider the order acceptance problem as an operational planning problem as opposed to some studies that consider it as a tactical problem. The studies focus on assigning due dates to incoming orders, making acceptance/rejection decisions, or controlling capacity. A common assumption is that orders have a string-type routing. Common performance measures are percentage of tardy orders and average tardiness. We also review some heuristics which are used to solve optimization problems in reasonable time.

4 Solution Design

In this chapter, we present how we aim to solve the order acceptance problem. Section 4.1 provides the formal problem definition. In Section 4.2, we discuss the assumptions we make to define and solve the order acceptance problem. Section 4.3 discusses the applicable finite loading methods and explains our choice of a finite loading method which forms the basis of our solution approach. Section 4.4 explains how we extend the finite loading method to account for the characteristics of the order acceptance problem, including capacity flexibility. In Section 4.5, we discuss the methods we use in combination with the procedure presented in the preceding section. Section 4.6 provides the summary of the chapter.

4.1 Problem definition

As mentioned in Section 3.7, we use a finite loading method as the basis of our solution approach. Finite loading methods are not designed to solve a clear-cut planning problem. Indeed, they are typically implemented like control policies or procedures for guiding decisions such as assigning (external) due dates to incoming orders and increasing capacity at short notice. When an incoming order arrives, they take into account information about the system and the incoming order in various ways. For example, in Forward Finite Loading (FFL) and Forward Finite Loading with Partial Loading (FFLPL), operations are loaded against (non-cumulative) capacity, whereas in Cumulative Forward Finite Loading (CFFL) operations are loaded against cumulative capacity. Also, in FFLPL operations can be loaded in multiple consecutive time periods, whereas in FFL an CFFL operations have to be loaded fully in a time period(s). In this section, we formally describe the input that is available during the order acceptance and the output that we want to achieve independent of the finite loading method used.

The production system under study is a job shop that consists of M workstations (index k). Each workstation k consists of m_k (≥ 1) identical resources (index l). Resource (k,l) refers to resource l of workstation k. Customer requests arrive dynamically at the shop and each customer request includes one or several products resulting in one or several orders. Order acceptance decisions can be made periodically or dynamically upon the arrival of every customer request. During the order acceptance, the incoming orders are planned and the existing orders may or may not be replanned. The number of orders under consideration is N. Each order i consists of n_i (≥ 1) operations (index j) with a string-type or assembly-type routing. Each order has an external due date which refers to the customer-requested due date for an incoming order and the agreed-upon due date for an existing order. Operation (i,j) refers to operation j of order i. Each operation (i,j) needs to be performed on a resource of workstation z_{ij} without interruption for the duration of p_{ij}. P_{ij} denotes the set of predecessors of operation (i,j).

The planning horizon is discretized into T time periods (index t) starting with 1. It is preferable to set time periods as a working day, because it would allow Limis' customers: 1) to know on which days orders are expected to be complete so that they can assess the feasibility of an external due date and quote an extended due date if needed, 2) to know the exact days in which additional capacity such as overtime is required. At Limis' customers, the start time and the length of the work shift on a working day may vary across resources as well as along the planning horizon. Resource (k,l) has a regular capacity of RC_{klt} hours and a maximum overtime of MO_{klt} hours in period t. SW_{klt} denotes the start

time of the work shift in period t on resource (k,l). Order i has an external due period d_i which is the external due date relative to the beginning of the planning horizon (i.e. the order acceptance day). β_i denotes the cost of extending d_i of order i one day, while γ_k denotes the cost of overtime per hour on resource (k,l), $l = 1,...,m_k$.

The desired output is: 1) for each order i, due date extension DDE_i denoting how much d_i should be extended in days, 2) for each period t on each resource (k,l), active overtime AO_{klt} denoting how much overtime in hours should be activated, 3) for each operation (i,j), due period DP_{ij} denoting the period in which the operation is expected to finish. Due periods of operations are later converted to due dates which are used as input for the priority dispatching rule ODD. We try to minimize the *total cost*, the sum of due date extension cost over all the orders and overtime cost over all the resources and periods, i.e. $\sum_{i=1}^{N} \beta_i DDE_i + \sum_{k=1}^{m} \sum_{l=1}^{m_k} \sum_{t=1}^{T} \gamma_k AO_{klt}$.

4.2 Assumptions

In this section, we discuss the assumptions we make to define and solve the order acceptance problem.

Limis' customers operate in a dynamic and stochastic environment, which causes them to constantly reschedule using Limis Planner. Limis Planner treats the scheduling problem as a static and deterministic problem while building some robustness into schedules by increasing operation processing times by a factor. Similarly, we address the order acceptance problem under the assumption of static and deterministic conditions.

Limis Planner relies on a built-in or external MRP function to determine the timings and quantities of stock orders and tries to schedule both stock orders and customer orders on time. If an operation is in short of a required material at the time it can be scheduled, it is assumed that the required material will be available after a fixed lead time and an alert is generated for the planner. We do not consider stock orders which are supposed to be generated by an MRP function. Stock orders can be considered just as customer orders. So, we assume a pure MTO job shop, as is the case for the chosen Limis' customer. We do not consider raw material requirements of customer orders assuming that materials are continuously available. Material availability can be considered as described above.

There are three flexibility options that can be used by the planner during the order acceptance, namely, due date flexibility (for incoming orders and some existing orders), capacity flexibility (by means of overtime, subcontracting, and reallocation of employees), and quantity flexibility (i.e. fulfilling incoming orders by partial shipments).

- We consider overtime as the only means of capacity increase. Since employee planning is not done by the chosen Limis' customer we would not be able to consider reallocation of employees. The decision on which operations can be subcontracted is made by the planner during the order acceptance. Subcontracting can be considered as mentioned in Section 4.4.2. For the purpose of simplicity, we choose to focus on the more difficult decision of overtime.
- The decision on which existing orders have flexible due dates is made by the planner. Due date flexibility of existing and incoming orders can be considered in the same way. The planner can influence the output (i.e. which existing and/or incoming orders are extended and how much) by setting due date extension cost β_i (i.e. an input parameter) variably among orders. For

example, we can avoid the due date extension of an existing order by setting the corresponding β_i extremely high. We make no assumption regarding which existing orders have flexible due dates and assume that the due dates of existing orders cannot be extended. We try not to extend the due dates of incoming orders using as much overtime as required and assume that due date extension costs of incoming orders are equal.

- Considering quantity flexibility means that lot sizing decisions need to be made for incoming orders. We do not consider this flexibility option and instead choose to focus on due date and capacity flexibility given the limited time available for this research. The planner may split the incoming orders at his/her own discretion and use the order acceptance method multiple times with a varying number of incoming orders.

We do not consider setup time, as the chosen Limis' customer has negligible sequence-independent setup time for a small portion of operations. We could consider sequence-independent setup times by adding them to processing times. However, in detailed schedules setup time may not be required for an operation if the previously executed operation on the same resource is for the same product. Therefore, by considering a setup time for every operation, we actually build some buffer into plans. As mentioned above, we propose to use a finite loading method which also includes planning parameters meant for building buffer into plans. So, if operations have significant setup times in the case of another company, by carefully configuring the planning parameters we could avoid unnecessary buffer built into plans.

Finally, for simplicity we assume that the work shift on a working day starts at the same time for all the workstations, 9:00, while varying start times can be easily considered.

4.3 Choosing a finite loading method

A finite loading method loads the operations of an order forward or backward in discrete time considering the workload of any previously loaded (existing or incoming) orders, capacity, precedence relations. In Section 3.3.2, we present three studies proposing a finite loading method for due date assignment and show that operations might be scheduled to finish earlier or later than their due date in detailed schedules. In FFL and FFLPL, deviations result from pieces of unplanned capacity, whereas in CFFL inaccuracy results from the fact that operations can consume capacity from the periods earlier than their due period. In the studies proposing FFL and CFFL, the planning parameters, the capacity loading limit CLL and/or the minimum waiting time MWT, are used to reduce deviations from operation due dates and therefore improve accuracy of assigned order due dates.

In those three studies, capacity is fixed, orders have a string-type routing, workstations have a single resource (i.e. all the resources are non-identical), time periods have equal length. Thürer and Stevenson (2020) consider capacity flexibility, while holding the other three assumptions. In the study, capacity is not increased by explicitly considering overtime or subcontracting, rather the workload in the period is decreased by a predetermined percentage. However, as mentioned in Section 4.1, we explicitly consider overtime. In other words, we seek to determine how much overtime is required in each period and on each resource.

We have set time periods as working days. At Limis' customers, the start time and the length of the work shift on a working day may vary across resources as well as along the planning horizon. An

operation may have a processing time larger than the length of the work shift of a day. Therefore, FFL is not applicable as it requires each operation to be fully loaded in a single time period. FFLPL allows partial loading but is not directly applicable as it assumes equal-length work shifts (or time periods) to find the required number of periods b_{ij} to load an operation. It is not difficult to modify FFLPL such that work shifts of varying lengths can be considered.

Given that operations cannot be preempted, the idea behind FFLPL as well as FFL is that loading an operation does not imply a delay to already loaded operations, although delays still occur due to the combined effect of unplanned capacity and the detailed scheduling. In CFFL, loading an operation may imply a delay to others because it can consume capacity from earlier periods than its due period. Nevertheless, CFFL can provide more accurate plans than for example FFL if the planning parameters are properly chosen, as shown by Thürer et al. (2013) and Thürer and Stevenson (2020). In CFFL, it is not needed to replan already loaded operations if we want to advance an operation by using overtime in some periods. However, in FFLPL, we we need to record how much of each operation is loaded in which period and may need to replan already loaded operations in order to to adhere to the idea that operations do not imply a delay to one another, which requires more computational effort. Therefore, we choose CFFL as a loading method.

We show how an operation can be advanced by using overtime in certain time periods according to FFLPL and CFFL with an example. Figure 4.1a illustrates how the first operation of order 5 with $p_{51} = 10hrs$ is loaded according to FFLPL on the required resource after loading orders 1, 2, 3, 4, where (i, j) represents operation j of order i, capacity represents regular capacity plus activated overtime, and maximum capacity represents regular capacity plus maximum possible overtime. Note that the way operation $(5, 1)$ is loaded in Figure 4.1a is not totally in line with FFLPL, but this way of loading does not imply a delay to other operations. Figure 4.2a illustrates how the same operation is loaded according to CFFL, where $MWT = 0$ and $CLL = 100\%$. Operations $(1, 1)$ and $(5, 1)$ arrive at the resource at the beginning of period 1, operations $(2, 2)$ and $(3, 2)$ arrive at the beginning of period 2, and operation $(4, 3)$ arrives at the beginning of period 3.

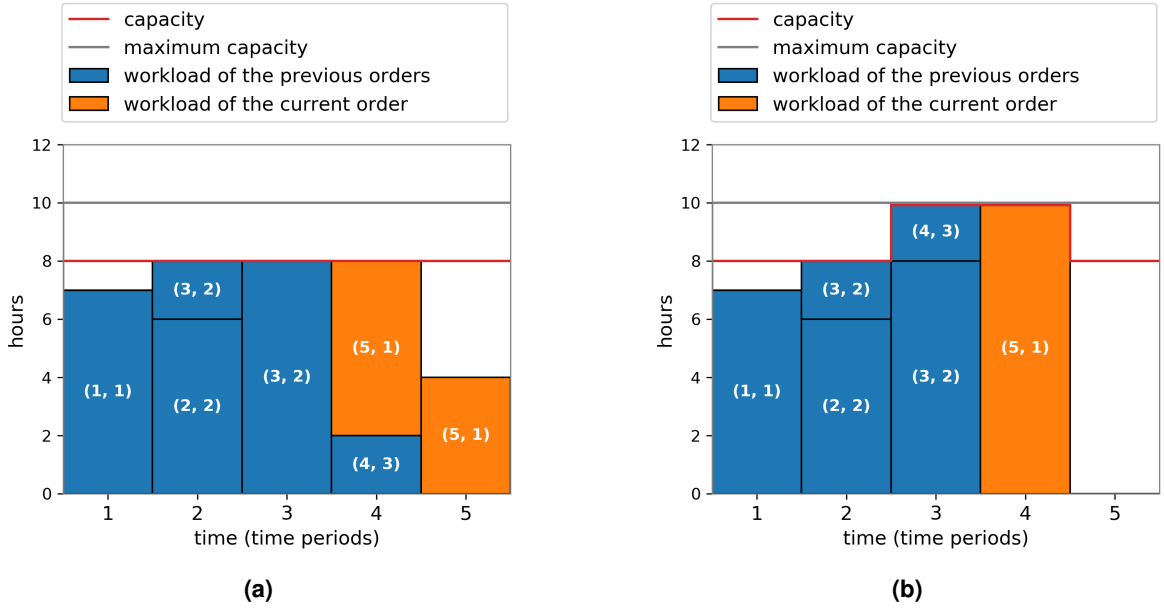


Figure 4.1: Example of advancing an operation in FFLPL

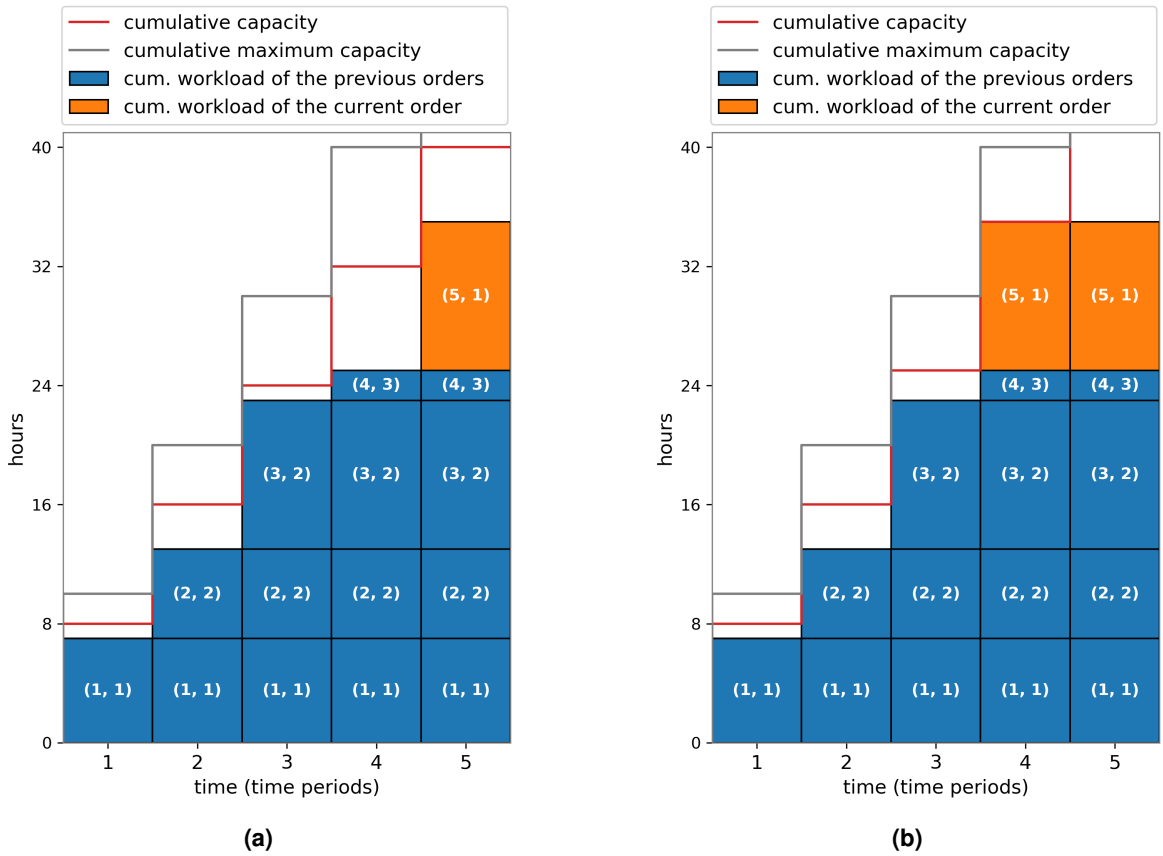


Figure 4.2: Example of advancing an operation in CFFL

Suppose that we want to advance operation $(5,1)$ one day. In FFLPL, for example, we can use 2 hours of overtime in periods 3 and 4 each, and move operation $(4,3)$ to period 3 so that we can advance operation $(5,1)$, as illustrated in Figure 4.1b. This means that in order to replan other operations we need to record how much of each operation is loaded in which period. In CFFL, for example, we can use 1 hour of overtime in period 3 and 2 hours in period 4, and do not have to move another operation to advance operation $(5,1)$, as illustrated in Figure 4.2b. Because cumulative capacity is used, we use less overtime in period 3 than in FFLPL. The implicit assumption in CFFL is that operations $(2,2)$, $(3,2)$, and $(4,3)$ arrive and start in the period preceding their due period, and therefore all the operations will be scheduled to complete on time. However, since $(2,2)$ arrives in period 2, there will be deviations.

Figures 4.3 and 4.4 illustrate the detailed schedules based on the ODD dispatching rule using the due dates set with FFLPL and CFFL respectively and ties are broken by choosing the operation with the shortest processing time. As can be seen from the figures, operation $(5,1)$ starts in period 1 delaying operations $(2,2)$ and $(3,2)$.

4.4 Procedure to load a sequence of orders

In this section, we present the CFFL-based procedure to load a sequence S of orders. We refer to the procedure as the *loading procedure* in the remainder of the thesis.

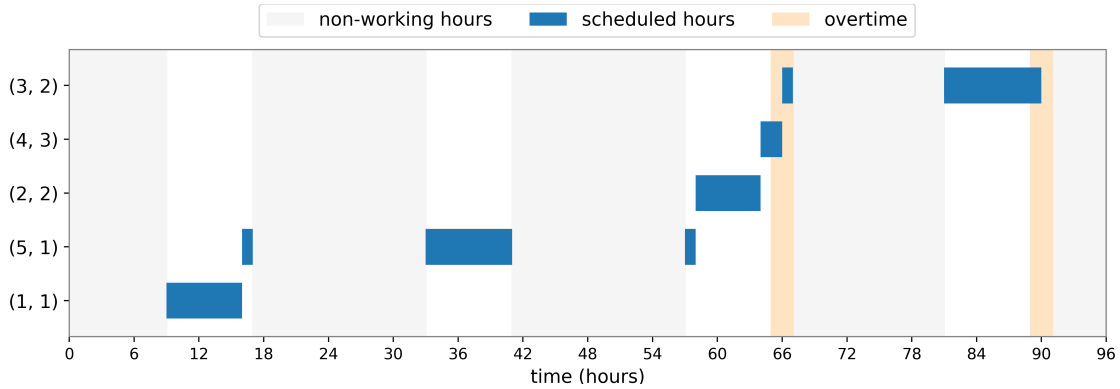


Figure 4.3: Example schedule with due dates set with FFLPL

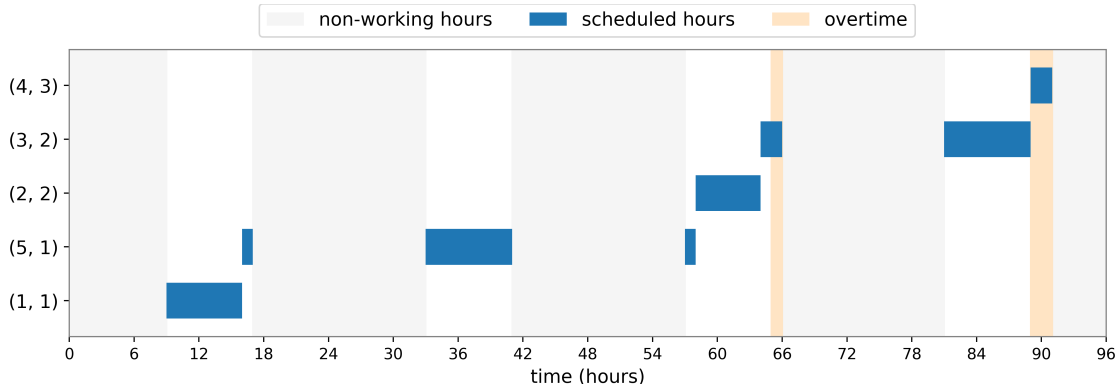


Figure 4.4: Example schedule with due dates set with CFFL

The loading procedure, which is implemented on a rolling horizon basis, receives some input parameters and provides among other variables: 1) due date extension DDE_i for each order i, 2) active overtime AO_{klt} for each period t and each resource (k,l). Given the variables DDE_i and AO_{klt} and parameters β_i and γ_k, we can find the *total cost*, the sum of due date extension cost over all the orders and overtime cost over all the resources and periods, i.e. $\sum_{i=1}^{N} \beta_i DDE_i + \sum_{k=1}^{m} \sum_{l=1}^{m_k} \sum_{t=1}^{T} \gamma_k AO_{klt}$. The loading procedure can be used as part of a heuristic to compare various sequences and choose the sequence with the minimum cost. Table 4.1 provides the list of input parameters and Table 4.2 provides the list of the main variables.

In this and the following sections, we use both due date and due period for operations and orders. Due period is the due date relative to the beginning of the planning horizon (i.e. the order acceptance day). The external due date of order i is converted to external due period d_i to be used as an input parameter for the loading procedure. Due time DT_{ij} of operation (i,j) denotes the time in the planning horizon by which the operation is expected to finish. Due period DP_{ij} of operation (i,j) denotes the period in which DT_{ij} falls. During the loading procedure, each operation (i,j) is loaded on a resource R_{ij} of the required workstation z_{ij} starting from period DP_{ij}. After the order acceptance, due periods of operations are converted to due dates to be used in detailed scheduling. Note that operations only have internal due periods (or dates) which we refer to as due periods (or dates). Moreover, after the order acceptance, each operation (i,j) is loaded on the *parent profile* of resource (z_{ij}, R_{ij}) starting from its due date, which will be explained in Section 5.1.2.

The internal due period of order i equals the due period of its last operation DP_{i,n_i}. After the order ac-

ceptance, internal due periods of orders are converted to internal due dates to be used in performance measurement. We define due date extension DDE_i of order i as the positive difference between the internal due period plus the due date buffer and the external due period, $((DP_{i,n_i} + DDB_i) - d_i)^+$. DDB_i is an additional planning parameter we use to compensate for deviations of order completion dates from internal due dates.

In contrast with Bertrand (1983) who uses the same values of the planning parameters, capacity loading limit and minimum waiting time, for all workstations assuming that workstations have similar queuing characteristics (i.e. processing times, utilization, etc.), we propose to use different values of MWT_k and CLL_k for a resource of each workstation k particularly because the utilization levels significantly differ in the chosen Limis' customer.

Period 1 represents the day of order acceptance. Order acceptance can take place in the middle of a day. Therefore, RC_{kl1}, MO_{kl1}, AO'_{kl1}, CWL'_{kl1} of resource (k,l) represent the values of the corresponding parameters from TOA until the end of the day. Similarly, SW_{kl1} of resource (k,l) represents the maximum of the actual start time of the work shift on the day of order acceptance and TOA. The working hours in period t on resource (k,l) are from SW_{klt} to $SW_{klt} + TC_{klt}$, while the maximum working hours are from SW_{klt} to $SW_{klt} + MC_{klt}$. In other words, TC_{klt} also represents the length of the work shift, while MC_{klt} represents the maximum length of the work shift. AO_{klt} initially equals AO'_{klt} in each period on each resource and may increase during the loading procedure, also increasing TC_{klt}. An increase in TC_{klt} means the length of the work shift also increases during the loading procedure. CWL'_{klt} includes the workload belonging to existing orders and is determined differently depending on whether existing orders are replanned or not. We explain the way in which CWL'_{klt} is determined in Section 5.1.2. CWL_{klt} initially equals CWL'_{klt} in each period on each resource and may increase during the loading procedure.

Release time RT_{ij} of operation (i,j) denotes the time (in the planning horizon) at which the operation is expected to arrive at the required workstation. When an existing order is replanned, its routing consists of the operations that have not been performed yet excluding those that are in progress. Therefore, a starting operation of an existing order may have predecessors that are in progress and in that case the release time of the starting operation equals the maximum scheduled completion time of its predecessors relative to the beginning of the horizon (i.e. the beginning of the order acceptance day). Otherwise, the release time of a starting operation equals TOA. This means that the release times of starting operations are actually parameters whereas the release times of other operations are variables. Hence, release time RT_{ij} of non-starting operation (i,j) equals the maximum due time of its predecessors, $RT_{ij} = \max_{j' \in P_{ij}} DT_{ij'}$. Release period RP_{ij} of operation (i,j) denotes the period in which RT_{ij} falls.

Because resources have varying work shifts in a period, RT_{ij} of operation (i,j) may fall outside the working hours of a resource (z_{ij},l) of the required workstation in period RP_{ij}, i.e. $RT_{ij} < SW_{z_{ij},l,RP_{ij}}$ or $RT_{ij} \geq SW_{z_{ij},l,RP_{ij}} + TC_{z_{ij},l,RP_{ij}}$. This has an impact on the loading of the operation as will be explained in Section 4.4.1 and may also have an impact on the amount of overtime required to advance the operation as will be explained in Section 4.4.2.

During the loading procedure, we consider the orders one at a time according to S. We start with the first order in S, and (re)load it until either its external due period is met, i.e. $DDE_i = 0$, or it is not possible to meet its external due period. The conditions for the latter situation are explained in Section 4.4.3. Once the first order is definitely loaded, we move to the next order. The loading procedure stops

Table 4.1: Input parameters

Parameter	Description
TOA	time of order acceptance on the day of order acceptance
T	number of time periods, i.e. working days, in the planning horizon
M	number of workstations in the shop
N	number of orders under consideration
S	sequence in which the orders are to be loaded
m_k	number of identical resources at workstation k
n_i	number of operations of order i
d_i	external due period (i.e. due date relative to the beginning of the horizon) of order i
DDB_i	due date buffer of order i
p_{ij}	processing time of operation (i,j)
z_{ij}	required workstation of operation (i,j)
P_{ij}	set of predecessors of operation (i,j)
β_i	cost of extending d_i of order i one day
γ_k	cost of overtime per hour on resource (k,l), $l = 1,...,m_k$
RC_{klt}	regular capacity in hours available in period t on resource (k,l)
MO_{klt}	maximum overtime in hours available in period t on resource (k,l)
MC_{klt}	maximum total capacity in hours available in period t on resource (k,l), i.e. $MC_{klt} = RC_{klt} + MO_{klt}$
CRC_{klt}	cumulative regular capacity in hours available from period 1 through period t on resource (k,l), i.e. $CRC_{klt} = \sum_{t'=1}^{t} RC_{k,l,t'}$
CMO_{klt}	cumulative maximum overtime in hours available from period 1 through period t on resource (k,l), i.e. $CMO_{klt} = \sum_{t'=1}^{t} MO_{k,l,t'}$
CMC_{klt}	cumulative maximum total capacity in hours available from period 1 through period t on resource (k,l), i.e. $CMC_{klt} = \sum_{t'=1}^{t} MC_{k,l,t'}$
CWL'_{klt}	existing cumulative workload in hours from period 1 through period t on resource (k,l)
AO'_{klt}	overtime in hours activated previously in period t on resource (k,l)
SW_{klt}	start time of the work shift in period t on resource (k,l)
MWT_k	minimum waiting time in hours, which is used when loading operations requiring resource (k,l), $l = 1,...,m_k$
CLL_k	capacity loading limit showing what percentage of the capacity in each period can be used on resource (k,l), $l = 1,...,m_k$

when all the orders are definitely loaded. The loading procedure consists of three main components, i.e. loading an order, advancing an operation of the order, reloading the order which we explain in Sections 4.4.1, 4.4.2, and 4.4.3 respectively. Some auxiliary variables are defined in those sections.

Table 4.2: Main variables

Variable	Description
AO_{klt}	overtime in hours activated in period t on resource (k,l)
TC_{klt}	total capacity in hours available in period t on resource (k,l), i.e. $TC_{klt} = RC_{klt} + AO_{klt}$
CAO_{klt}	cumulative maximum overtime in hours activated from period 1 through period t on resource (k,l), i.e. $CAO_{klt} = \sum_{t'=1}^{t} AO_{k,l,t'}$
CTC_{klt}	cumulative total capacity in hours available from period 1 through period t on resource (k,l), i.e. $CTC_{klt} = \sum_{t'=1}^{t} TC_{k,l,t'}$
CWL_{klt}	cumulative workload in hours from period 1 through period t on resource (k,l)
RT_{ij}	release time of operation (i,j), i.e. $RT_{ij} = \max_{j' \in P_{ij}} DT_{ij'}$
RP_{ij}	release period of operation (i,j), i.e. $RP_{ij} = \max_{j' \in P_{ij}} DP_{ij'}$
DT_{ij}	due time of operation (i,j)
DP_{ij}	due period of operation (i,j)
R_{ij}	resource on which operation (i,j) is loaded
DDE_i	number of days d_i of order i should be extended, i.e. $DDE_i = ((DP_{i,n_i} + DDB_i) - d_i)^+$

4.4.1 Loading an order

In Section 3.3.2, we present how an operation is loaded according to CFFL assuming that orders have string-type routings, workstations have a single resource, time periods have equal lengths (i.e. work shifts are the same across resources and across time periods), capacity is fixed. We have modified CFFL to account for assembly-type routings, workstations with multiple resources, varying work shifts, and capacity flexibility.

In the case of a string-type routing, the sequence in which the operations of an order are loaded is straightforward, i.e. from the starting operation to the last operation. In the case of an assembly-type routing, we need to determine the sequence in which to load the operations. Such a sequence makes a difference particularly when multiple operations of the order require the same workstation. We load the operations based on their release times. Release time-based sequencing is similar to the logic of detailed scheduling in which a resource is not held on when there is an operation to process. Note that RT_{ij} of operation (i,j) is known once all the predecessors have been loaded.

When there are multiple resources of the workstation required by an operation, we need to choose a resource on which to load the operation. We choose the resource on which the operation can finish the earliest as CFFL is a forward loading method. In order to find the period in which operation (i,j) can finish on resource l of the required workstation z_{ij}, similar to the original CFFL, we first find the period in which $RT_{ij} + MWT_{z_{ij}} + p_{ij}$ falls considering only the working hours of periods starting with RP_{ij}. We refer to that period as the *preliminary finish period* FP'_{ijl}. Note that the working hours in a period depends on the start time of the work shift and the total capacity (i.e. the length of the work shift) as mentioned above. Next, similar to the original CFFL, we find the earliest period $t \geq FP'_{ijl}$ in which and all the subsequent periods the operation can be fully loaded taking into account the availability of cumulative total capacity, i.e. $CWL_{z_{ij},l,t'} + p_{ij} \leq CLL_{z_{ij}} * CTC_{z_{ij},l,t'}, t' = t, t+1, \ldots, T$. We refer to that period as the *finish period* FP_{ijl}.

As mentioned above, we load the operation on the resource with the minimum finish period, i.e. $R_{ij} = \arg\min_l FP_{ijl}$, and set the due period DP_{ij} to the corresponding finish period, i.e. $DP_{ij} = FP_{i,j,R_{ij}}$. This means p_{ij} is added to $CWL_{z_{ij},R_{ij},t'}, t' = DP_{ij}, DP_{ij}+1, \ldots, T$.

To complete the loading of operation (i,j), we need to set the due time DT_{ij} to some time in the due period. According to the original CFFL, we would set DT_{ij} differently depending on whether $FP_{i,j,R_{ij}}$ (or DP_{ij}) equals $FP'_{i,j,R_{ij}}$ or not. If $FP_{i,j,R_{ij}} = FP'_{i,j,R_{ij}}$, DT_{ij} is set to the time of $AT_{ij} + MWT_{z_{ij}} + p_{ij}$ found as explained above. If $FP_{i,j,R_{ij}} > FP'_{i,j,R_{ij}}$, DT_{ij} is set to the start time of the work shift plus 0.25 times the length of the work shift, i.e. $SW_{z_{ij},R_{ij},DP_{ij}} + 0.25 * TC_{z_{ij},R_{ij},DP_{ij}}$. However, we set DT_{ij} to $SW_{z_{ij},R_{ij},DP_{ij}} + 0.9 * TC'_{z_{ij},R_{ij},DP_{ij}}$ regardless of whether $FP_{i,j,R_{ij}}$ (or DP_{ij}) equals $FP'_{i,j,R_{ij}}$ or not, where TC'_{klt} equals the value of TC_{klt} right before the order is loaded. We explain the reason behind our choice in Section 4.4.3. TC'_{klt} is updated with TC_{klt} once an order is definitely loaded.

After the order is loaded if the external due period is met meaning that $DDE_i = 0$, we move to the next order in S if any. Otherwise, we activate additional overtime to advance an operation.

We illustrate the loading of an operation with two examples.

Example 1

Order 1 is the fourth order in S and has an assembly-type routing shown in Figure 4.5. As can be seen from the figure, the starting operations $(1,1)$ and $(1,2)$ require workstation 1. Workstation 1 has a single resource. The order is an existing order and has some operations that are in progress at the time of order acceptance. Because of those ongoing operations, operations $(1,1)$ and $(1,2)$ are released at $RT_{11} = 13$ and $RT_{12} = 13.25$ respectively. Operations $(1,1)$ and $(1,2)$ have a workload of $p_{11} = 3.5$ *hrs* and $p_{12} = 4.25$ *hrs* respectively. We set $MWT_1 = 1$ *hr* and $CLL_1 = 100\%$ for workstation 1. The parameters for resource $(1,1)$ are as follows. Regular capacity $RC_{11t} = 4$ and maximum overtime $MO_{11t} = 1$ in each period. Because no overtime has been activated for the previously loaded orders, i.e. $AO_{11t} = 0$ in each period, total capacity $TC_{11t} = 4$ in each period. CWL_{11t} equals 2, 3, 5, 8, 8, and 15 in periods 1, 2, 3, 4, 5, and 6 respectively and does not increase after period 6.

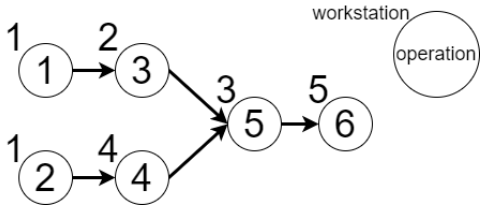


Figure 4.5: Routing of order 1 in Example 1 and 2

Figure 4.6a illustrates how operation $(1,1)$ is loaded first and Figure 4.6b illustrates how operation $(1,2)$ is loaded after $(1,1)$. Note that operation $(1,1)$ is loaded first as it is released earlier. In the figures, the edges of the blue bars along the time axis represent the start and end times of the work shifts in the periods. Since CWL_{11t} does not increase after period 6, for the purpose of conciseness, we show in the figures only the first eight periods along the planning horizon.

As can be seen from Figure 4.6a, $RT_{11} + MWT_1 + p_{11} = 57.5$ falls in period 3 considering only

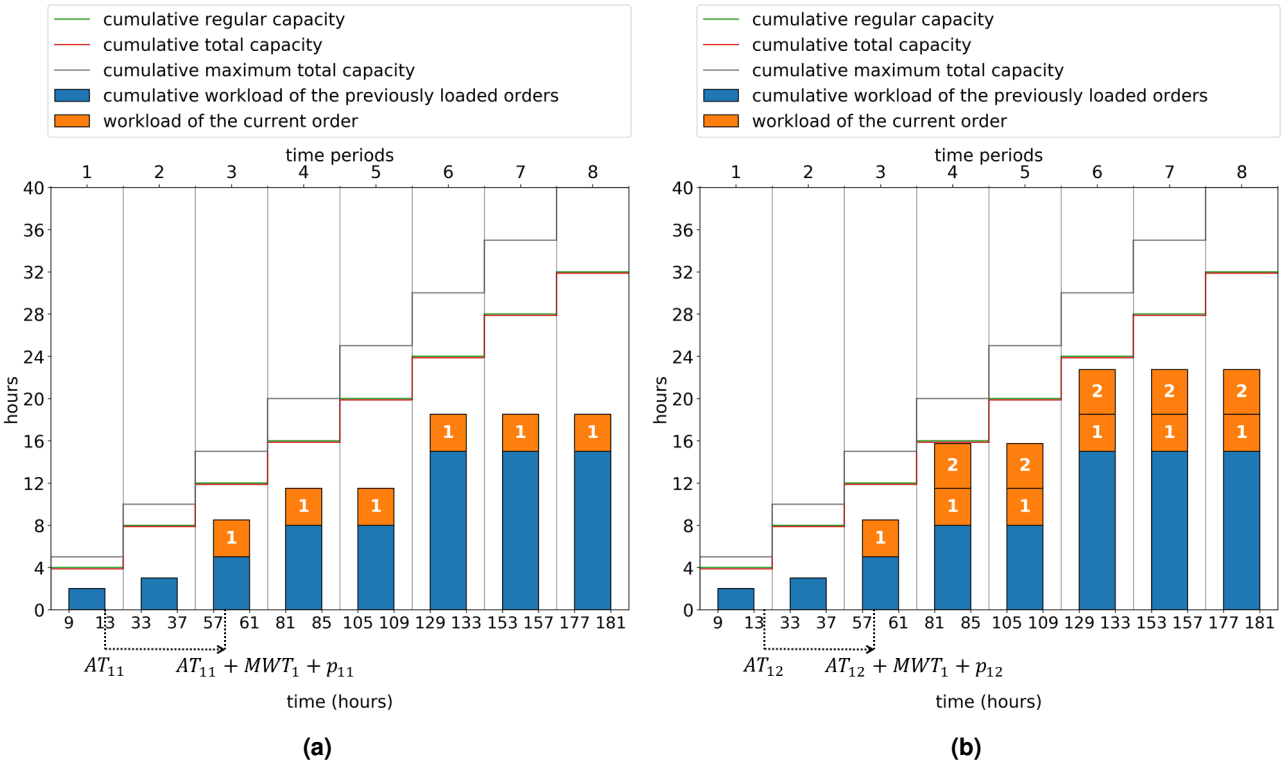


Figure 4.6: Loading (a) operation (1,1) and (b) operation (1,2) in Example 1

the working hours in periods 1, 2, and 3. So, the preliminary finish period $FP'_{111} = 3$. Note that RT_{11} falls right at the end of the work shift in period 1 and therefore no working hour counts toward $MWT_1 + p_{11}$ in period 1. The operation can be fit in period 3 and all the subsequent periods, and therefore the finish period $FP_{111} = 3$. The due period $DP_{11} = FP_{111} = 3$ as there is a single resource. In this particular example, we set DT_{11} according to the original CFFL. Therefore, $DT_{11} = RT_{11} + MWT_1 + p_{11} = 57.5$, since FP_{111} (or DP_{11}) equals FP'_{111}.

As can be seen from Figure 4.6b, $RT_{12} + MWT_1 + p_{12} = 58.25$ falls in period 3 considering only the working hours in periods 1, 2, and 3. So, the preliminary finish period $FP'_{121} = 3$. Note that RT_{12} falls beyond the end of the work shift in period 1 and therefore no working hour counts toward $MWT_1 + p_{12}$ in period 1. The operation cannot be fit in period 3, and therefore the finish period $FP_{121} = 4$. The due period $DP_{12} = FP_{121} = 4$ as there is a single resource. In this particular example, we set DT_{12} according to the original CFFL. Therefore, $DT_{12} = SW_{114} + 0.25 * TC_{114} = 81 + 0.25 * 4 = 82$, since FP_{121} (or DP_{12}) is greater than FP'_{121}.

Example 2

The input of Example 2 is the same as that of Example 1 expect for the values of CWL_{11t}. CWL_{11t} equals 2, 3, 6, 15, 15, 21.75, and 23.25 in periods 1, 2, 3, 4, 5, 6, and 7 respectively and does not increase after period 7. Figure 4.6a illustrates how operation $(1,1)$ is loaded first and Figure 4.6b illustrates how operation $(1,2)$ is loaded after $(1,1)$.

As can be seen from the figures, the output is the same as in Example 1 in terms of the values of $RT_{11} + MWT_1 + p_{11}$ and $RT_{12} + MWT_1 + p_{12}$. Therefore, $FP'_{111} = 3$ and $FP'_{121} = 3$ remain

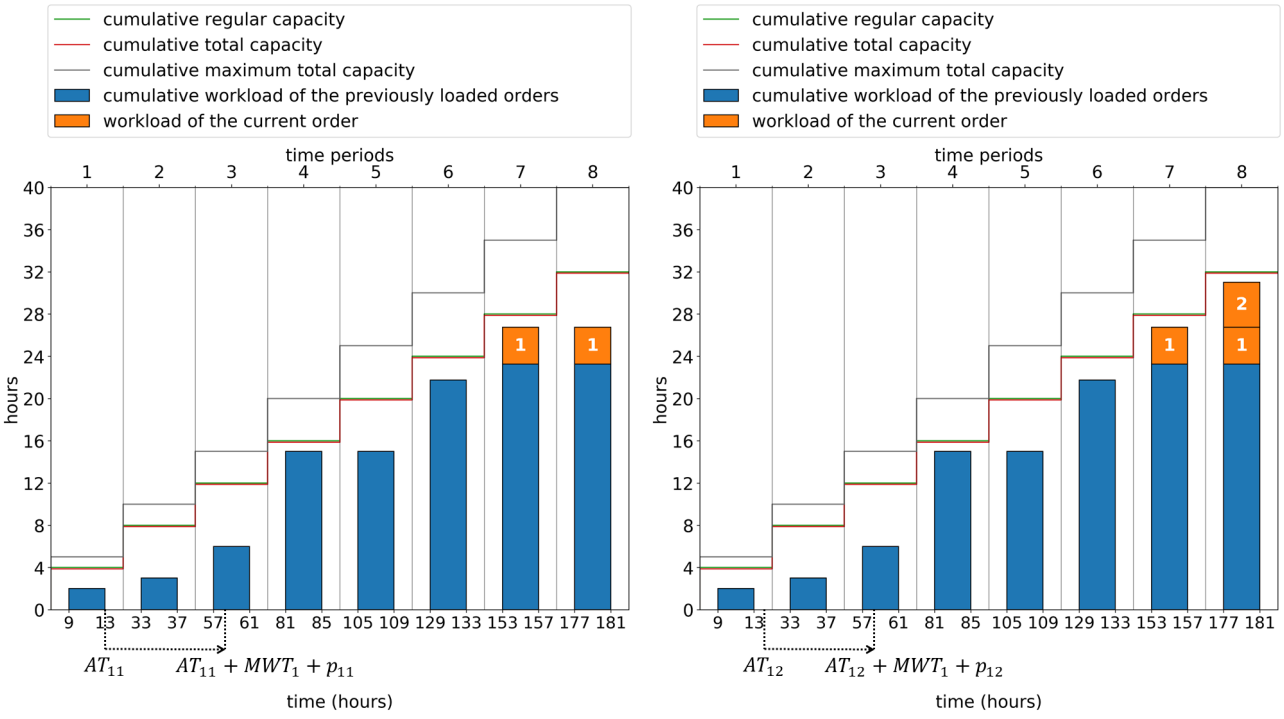


Figure 4.7: Loading (a) operation (1,1) and (b) operation (1,2) in Example 2

the same as well. Because operation $(1,1)$ cannot be fit in periods 4 and 6, $FP_{111} = 7$ as well as $DP_{11} = 7$. Because operation $(1,2)$ cannot be fit in periods 4, 6, and 7, $FP_{121} = 8$ as well as $DP_{12} = 8$. Also in this example, we set DT_{11} and DT_{12} according to the original CFFL. $DT_{11} = SW_{117} + 0.25 * TC_{117} = 153 + 0.25 * 4 = 154$, since FP_{111} (or DP_{11}) is greater than FP'_{111}. $DT_{12} = SW_{118} + 0.25 * TC_{118} = 177 + 0.25 * 4 = 178$, since FP_{121} (or DP_{12}) is greater than FP'_{121}.

4.4.2 Advancing an operation

If the external due period cannot be met, we need to advance an operation by activating additional overtime on a resource of the required workstation. Advancing an operation essentially means reducing the due period of an operation with the hope of meeting the external due period of the order when it is reloaded. It is worth noting that CFFL originally does not consider advancing an operation or increasing capacity by means of overtime.

First, we need to choose an operation to advance. For an assembly-type routing, we need to choose an operation on the critical path, as the flow time of the order cannot be decreased without advancing a critical operation. We find the critical path by starting with the last operation and picking the predecessor with the maximum due period each time until a starting operation is picked. We choose the operation which has the highest due period relative to the earliest period in which it could finish given maximum total capacity and assuming zero workload on resources. We believe that the corresponding workstation is likely to be the bottleneck workstation. This is similar to the logic of Thurer et al. (2020) who choose the operation with the maximum throughput time for the same reason. The earliest period

in which operation (i, j) can finish on resource l of the required workstation z_{ij} given maximum total capacity and assuming zero workload is the period in which $RT_{ij} + MWT_{z_{ij}} + p_{ij}$ falls considering the maximum working hours of periods starting with RP_{ij}. We refer to that period as the *earliest preliminary finish period* EFP'_{ijl}. Note that the maximum working hours in a period depends on the start time of the work shift and the maximum total capacity (i.e. the maximum length of the work shift). Also note that RT_{ij} of operation (i, j) was found during the loading of the order. As mentioned above, we choose the critical operation with the maximum non-zero $y_{ij} = DP_{ij} - \min_l EFP'_{ijl}$.

Next, we need to determine on which resource to advance the chosen operation (i, j) and how much to advance it. We choose to advance the operation by DDE_i, if possible. If not possible, we choose to advance it by the maximum possible amount. We believe that this approach increases our chances of meeting the external due period of the order when we reload it while avoiding excessive use of overtime. In order to determine how much the chosen operation (i, j) can be advanced on resource l of the required workstation z_{ij}, we find the earliest period $t \geq EFP'_{ijl}$ in which and all the subsequent periods the operation can be fully loaded taking into account the availability of cumulative maximum total capacity, i.e. $CWL_{z_{ij},l,t'} + p_{ij} \leq CLL_{z_{ij}} * CMC_{z_{ij},l,t'}, t' = t, t+1, \ldots, T$. We refer to that period as the *earliest finish period* EFP_{ijl}. Note that cumulative workload used to find EFP_{ijl} possibly includes the workload of the other operations of the order as determined during the (re)loading. By keeping the other operations that require the same workstation as the chosen operation loaded on the resources, we prevent them from being delayed as a result of advancing an operation. The operation (i, j) can be advanced on resource l of the required workstation z_{ij} by at most $DP_{ij} - EFP_{ijl}$. If EFP_{ijl} on each resource is greater than or equal to DP_{ij}, it means the operation cannot be advanced on any resource even using maximum capacity and therefore we place the chosen operation, all of its predecessors, the predecessors of its predecessors, etc. until starting operations in set Z_i, i.e. operations in Z_i are not chosen for advancing, and then we choose the next critical operation with the maximum non-zero y_{ij}. Otherwise, the operation can be advanced and we proceed as follows.

As explained above, we choose the resource with the maximum $x_{ijl} = \min(DP_{ij} - EFP_{ijl}; DP_{i,n_i} + DDB_i - d_i)$ and set R_{ij} to $\max_l x_{ijl}$. In case of a tie between the resource on which the operation was loaded previously and another resource, the former is chosen as it is likely to require less overtime to advance the operation by the same amount. We define an auxiliary variable *advanced due period* ADP_{ij} as $DP_{ij} - x_{i,j,R_{ij}}$. We load the operation on the chosen resource, which means p_{ij} is added to $CWL_{z_{ij},R_{ij},t'}, t' = ADP_{ij}, ADP_{ij}+1, \ldots, T$.

If subcontracting were considered and it was possible to subcontract the chosen operation, in addition to advancing the operation on an internal resource, we could do the following. We first find how much the operation can be advanced by subcontracting it since we know a fixed lead time is considered in Limis Planner for subcontracted operations. If the amount is greater than or equal to $x_{i,j,R_{ij}}$, we compare the cost of subcontracting with the cost of additional overtime required to advance the operation and choose the minimum-cost option.

Loading the operation on the chosen resource starting from period ADP_{ij} results in overload in some of the periods ADP_{ij}, $ADP_{ij}+1$, ..., $FP_{i,j,R_{ij}} - 1$ with respect to cumulative total capacity. We need to activate additional overtime on the chosen resource in such a way that there is no period with overload. Also, we need to ensure that $FP'_{i,j,R_{ij}}$ will be less than or equal to ADP_{ij} when the operation is reloaded. As mentioned in Section 4.4.1, $FP'_{i,j,R_{ij}}$ depends on the working hours in periods starting with RP_{ij}. This means the working hours in periods starting with RP_{ij} should be long enough to allow $FP'_{i,j,R_{ij}}$ to be less than or equal to ADP_{ij}. So we activate additional overtime in two steps. In Step

1, we activate overtime to decrease $FP'_{i,j,R_{ij}}$ if required. In Step 2, we activate overtime to eliminate overload in periods if required. It is worth noting that activating overtime in Step 1 can also solve some or all of the overload issues because of the cumulative nature of capacity. Below we explain first how we activate overtime in two steps and later why we do so.

1. If ADP_{ij} is less than the current $FP'_{i,j,R_{ij}}$, we need to activate as much additional overtime in periods RP_{ij}, $RP_{ij}+1,...,ADP_{ij}$ as required to ensure that $FP'_{i,j,R_{ij}}$ will be equal to ADP_{ij}. We denote with θ_{klt} the maximum amount of additional overtime that can be activated in a period on a resource, i.e. $\theta_{klt} = MO_{klt} - AO_{klt}$ and with λ_{ij} the amount of working hours between RT_{ij} and the end of the work shift in period ADP_{ij}. So, the additional overtime that is required to be activated in periods RP_{ij}, $RP_{ij}+1,...,ADP_{ij}$ is at least $\tau = \lambda_{ij} - MWT_{z_{ij}} - p_{ij}$. We start with period ADP_{ij}, activate additional overtime in the amount $\min\left(\theta_{z_{ij},R_{ij},ADP_{ij}};\ \tau\right)$ and update τ, i.e. $\tau = \tau - \min\left(\theta_{z_{ij},R_{ij},ADP_{ij}};\ \tau\right)$. If τ is greater than 0, we do the same for the preceding period $ADP_{ij}-1$. This continues until either τ equals 0 or we reach period RP_{ij}. If we reach period RP_{ij} and τ is still non-zero, we activate additional overtime in the amount τ plus the positive difference between RT_{ij} and the end of the work shift in period RP_{ij}. This means that in period RP_{ij} we might need to activate more than τ if the operation arrives after the end of the work shift in the period in order to obtain continuous working hours. Note that an increase in $AO_{z_{ij},R_{ij},t}$ in a period contributes to $CAO_{z_{ij},R_{ij},t}$ in that and all the subsequent periods.
2. After ensuring that $FP'_{i,j,R_{ij}}$ will be less than or equal to ADP_{ij} by possibly updating $AO_{z_{ij},R_{ij},t}$ and $CAO_{z_{ij},R_{ij},t}$ in some periods, in step 2, we need to eliminate overload issues in periods ADP_{ij}, $ADP_{ij}+1$, ..., $FP_{i,j,R_{ij}}-1$ if any. We denote with μ the required amount of additional cumulative overtime in a period with overload, i.e. $\mu = CWL_{klt} - CLL_k * CTC_{klt}$. As additional overtime in a period contributes to the cumulative overtime in that and all the subsequent periods, we start with the first period t' with overload and do the following. We activate additional overtime in the amount $\min\left(\mu;\theta_{z_{ij},R_{ij},t'}\right)$ and update μ, i.e. $\mu = \mu - \min\left(\mu;\theta_{z_{ij},R_{ij},t'}\right)$. If μ is greater than 0, we do the same for the preceding period. This continues until μ equals 0. Note that an increase in $AO_{z_{ij},R_{ij},t}$ in a period contributes to $CAO_{z_{ij},R_{ij},t}$ in that and all the subsequent periods. We do the same for the next period with overload, if any, until there is no period with overload.

As mentioned in Section 4.3, in detailed schedules some operations are completed early making others late. So operations for which overtime are activated are not always the operations being performed in those overtime hours. Nevertheless, timing of overtime should be chosen such that it overlaps with periods of actual need. In the case of ADP_{ij} being less than $FP'_{i,j,R_{ij}}$, we distribute the required amount of overtime τ backward starting with ADP_{ij} because if the operation arrives later than the planned arrival time it will still use the overtime allocated for it. Similarly, in the case of a period with overload, we distribute the required amount of overtime μ backward starting with that period, which will still increase the chances of the operations planned to finish in the period with overload to use the overtime allocated for them.

We continue with the examples presented in Section 4.4.1 to illustrate how we advance an operation.

Example 1 (cont.)

Suppose that the external due period of order 1 cannot be met and $DDE_1 = 2$. First, we find the earliest preliminary finish periods for the operations of order 1. For operation $(1,1)$, $EFP'_{111} = 2$ as $RT_{11} + MWT_1 + p_{11} = 36.5$ considering the maximum working hours in pe-

riods 1 and 2. For operation $(1,2)$, $EFP'_{121} = 2$ as $RT_{12} + MWT_1 + p_{12} = 37.5$ considering the maximum working hours in periods 1 and 2. Suppose that operation $(1,2)$ becomes the critical operation with the highest $y_{12} = DP_{12} - EFP'_{121} = 4 - 2 = 2$. The earliest finish period $EFP_{121} = 2$ given the availability of cumulative maximum total capacity in period 2 and therefore $x_{121} = \min(4-2;2) = 2$. Since there is a single resource, $R_{12} = 1$ and $ADP_{12} = DP_{12} - x_{121} = 4 - 2 = 2$. Loading operation $(1,2)$ starting from period 2 causes overload in period 3 with regard to cumulative total capacity, as depicted in Figure 4.8.

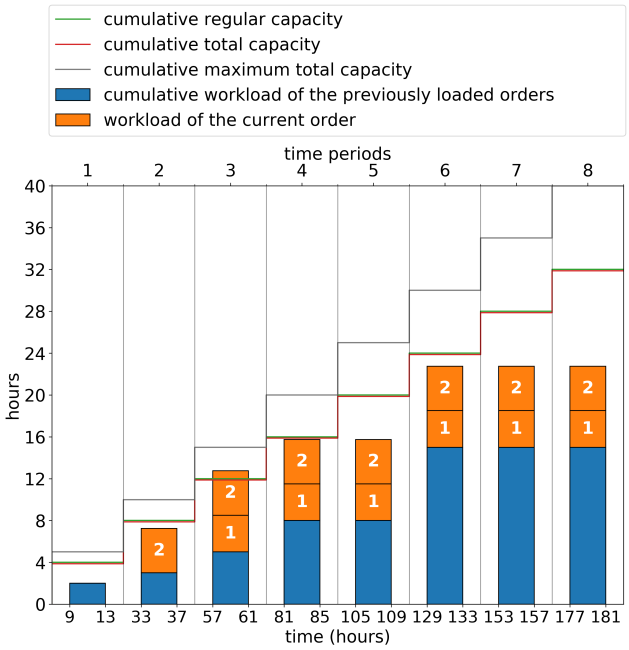


Figure 4.8: Advancing operation (1,2) in Example 1

Since $FP'_{121} = 3$ is greater than $ADP_{12} = 2$, we need to activate overtime in periods 1 and 2 to decrease the value of FP'_{121} to 2. Following step 1 described above, we activate overtime in periods 1 and 2 in the amounts of $0.5hrs$ and $1hrs$ respectively. As a result, $RT_{12} + MWT_1 + p_{12} = 38$ considering the working hours in periods 1 and 2 and therefore $FP'_{121} = 2$, as depicted in Figure 4.9. Note that we activate $0.5hrs$ of overtime in period 1 because operation $(1,2)$ is released at 13.25 and needs 0.25 working hours after 13.25 in period 1. As a result of activating overtime in step 1, the overload in period 3 is also eliminated, as can be seen from Figure 4.9. So, we do not need to activate additional overtime in step 2.

Example 2 (cont.)

Similar to Example 1, suppose $DDE_i = 2$ and operation $(1,2)$ becomes the critical operation with the highest $y_{12} = DP_{12} - EFP'_{121} = 8 - 2 = 6$. Therefore, $EFP_{121} = 2$ given the availability of cumulative maximum total capacity in period 2 and $x_{121} = \min(8-2;2) = 2$. However, $ADP_{12} = DP_{12} - x_{121} = 8 - 2 = 6$. Loading operation $(1,2)$ starting from period 6 causes overload in periods 6 and 7 with regard to cumulative total capacity, as depicted in Figure 4.10.

Since $FP'_{121} = 3$ is less than $ADP_{12} = 6$, we do not need to activate overtime in step 1. So, first we activate overtime to eliminate the overload in period 6. Following step 2 described above, we activate overtime in periods 5 and 6 in the amounts of $1hr$ and $1hr$ respectively, as

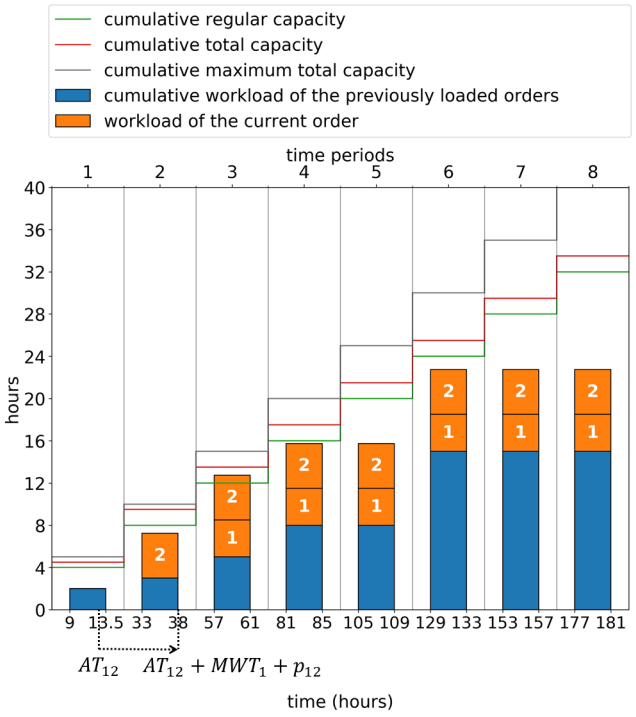


Figure 4.9: Activating overtime for operation (1,2) in step 1 in Example 1

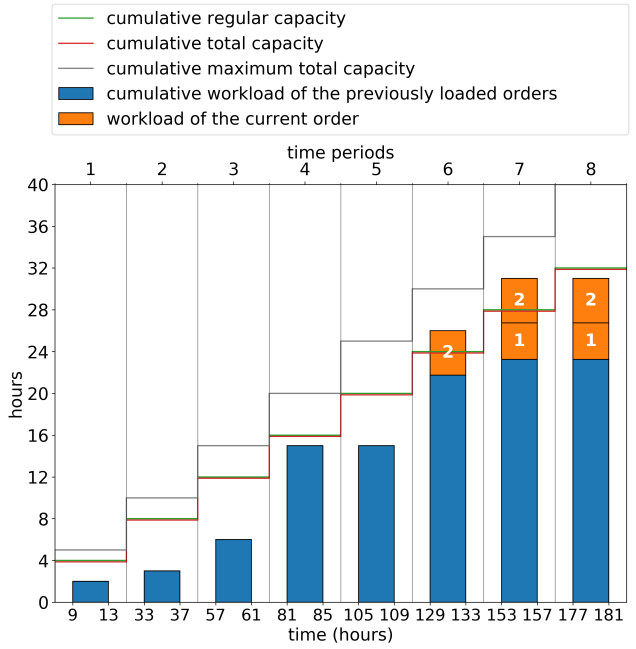


Figure 4.10: Advancing operation (1,2) in Example 2

depicted in Figure 4.11a. Because there is still overload in period 7, following step 2 again, we activate $1hr$ of overtime in period 7, as depicted in Figure 4.11b.

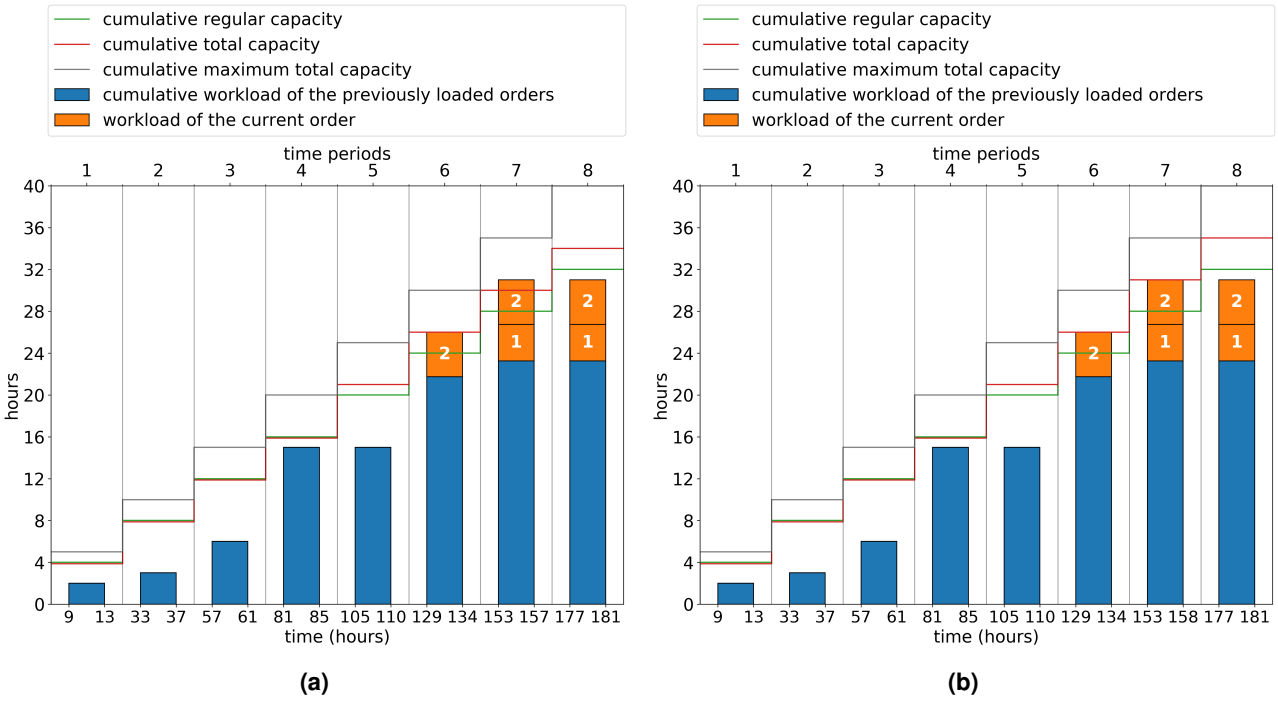


Figure 4.11: Activating overtime to eliminate overload in period (a) 6 and (b) 7 in Example 2

4.4.3 Reloading the order

Reloading an order is essentially the same as loading. The main difference is that we unload each operation right before reloading it. This is to ensure that the due period of an operation does not increase after it is loaded during the first loading of the order, particularly in an assembly-type routing where multiple operations might require the same workstation. In addition, we reload an operation that has been advanced before specifically on the resource where overtime was activated for it rather than loading it on the resource with the minimum finish period. In other words, we do not change R_{ij} of operation (i, j) during the reloading if it has been advanced before. This is to ensure that operations use the overtime activated for them. It is also worth noting during the reloading the due period of the operation that has just been advanced can be lower than the advanced due period because of the way we define x_{ijl}.

After the reloading, if the external due period of the order can be met, we move to the next order in S, if any. Otherwise, another operation is advanced in the same way as described above and the order is reloaded again. The loading procedure moves to the next order either when the external due period is met or when there is no critical operation to advance which has a non-zero y_{ij} and does not belong to set Z_i. The latter case indicates that the internal due period of the order cannot be decreased further and therefore it is not possible to meet its external due period. Once all the orders are definitely loaded, we find the total cost of the sequence, $\sum_{i=1}^{N} \beta_i DDE_i + \sum_{k=1}^{m} \sum_{l=1}^{m_k} \sum_{t=1}^{T} \gamma_k AO_{klt}$.

We continue with the examples presented in Sections 4.4.1 and 4.4.2.

Example 1 (cont.)

Figure 4.12a depicts how operation $(1, 1)$ is reloaded and Figure 4.12b depicts how operation

$(1,2)$ is reloaded. $RT_{11} + MWT_1 + p_{11} = 37$ falls in period 2 considering the working hours in periods 1, and 2 and therefore $FP'_{111} = 2$. As can be seen from Figure 4.12a, operation $(1,2)$ is kept loaded on the resource so that operation $(1,1)$ does not consume the overtime activated for operation $(1,2)$. Therefore, since operation $(1,1)$ cannot be fit in period 2, the finish period $FP_{111} = 3$ as well as the due period $DP_{11} = 3$. Note that DP_{11} has remained the same. However, according to the original CFFL, DT_{11} now should be set to $DT_{11} = SW_{113} + 0.25 * TC_{113} = 57 + 0.25 * 4 = 58$ since FP_{111} or (DP_{11}) is greater than FP'_{111}. So, DT_{11} has increased from 57.5 to 58 although DP_{11} has remained the same. This unjustified increase in DT_{11} might cause delay in the due period of the successor operation. Therefore, setting DT_{ij} of operation (i,j) depending on whether $FP_{i,j,R_{ij}}$ (or DP_{ij}) equals $FP'_{i,j,R_{ij}}$ is not favorable when there is capacity flexibility.

As can be seen from Figure 4.12b, $RT_{12} + MWT_1 + p_{12} = 38$ falls in period 2 and therefore $FP'_{121} = 2$. As expected, $FP_{121} = 2$ and $DP_{12} = 2$.

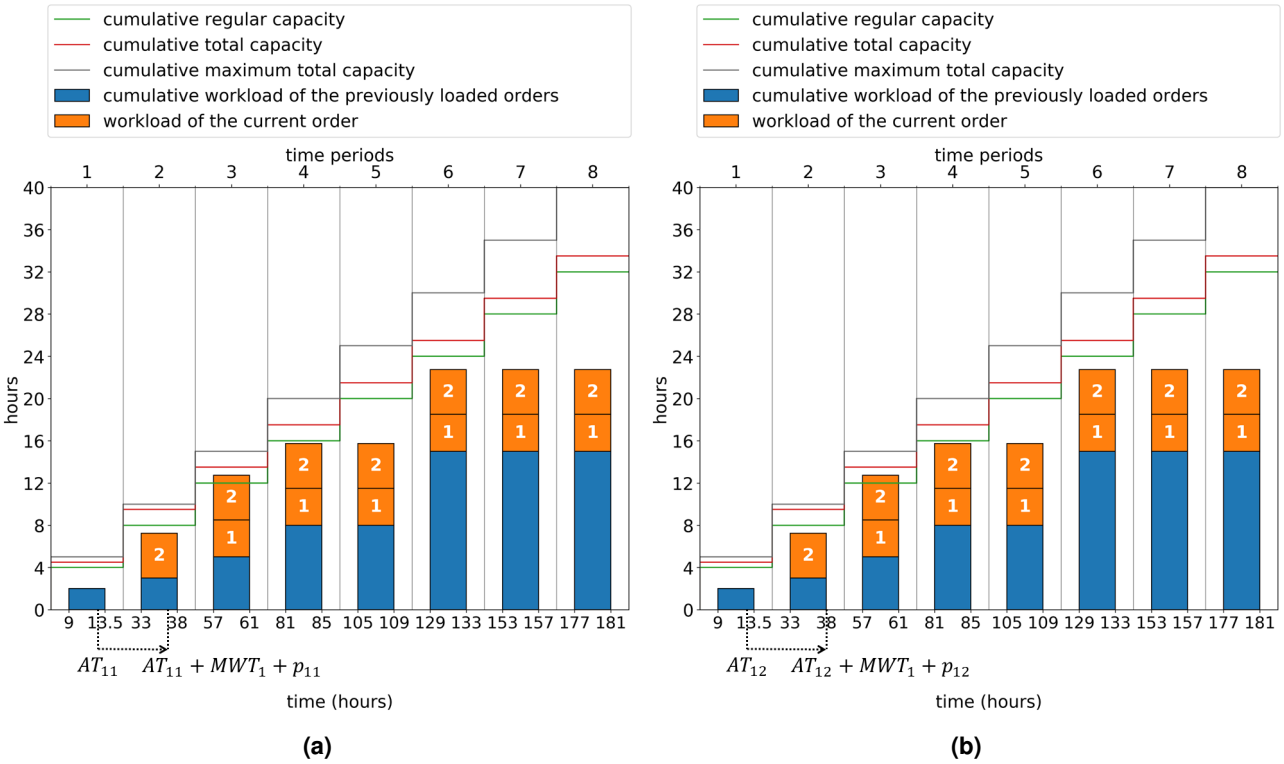


Figure 4.12: Reloading (a) operation (1,1) and (b) operation (1,2) in Example 1

Example 2 (cont.)

Figure 4.13a depicts how operation $(1,1)$ is reloaded and Figure 4.13b depicts how operation $(1,2)$ is reloaded. As can be seen from Figure 4.13a, $RT_{11} + MWT_1 + p_{11} = 57.5$ falls in period 3 and therefore $FP'_{111} = 3$. Because the operation cannot be fit in periods 4 and 6, $FP_{111} = 7$ and $DP_{11} = 7$ as before. As can be seen from Figure 4.13a, operation $(1,2)$ is kept loaded on the resource so that operation $(1,1)$ does not consume the overtime activated for operation $(1,2)$. Note that DP_{11} has remained the same. However, according to the original CFFL, DT_{11} now should be set to $DT_{11} = SW_{117} + 0.25 * TC_{117} = 153 + 0.25 * 5 = 154.25$ since TC_{117} has increased. So, DT_{11} has increased from 154 to 154.25 although DP_{11}

has remained the same. This increase in DT_{11} might cause delay in the due period of the successor operation. Therefore, setting DT_{ij} of operation (i, j) depending on $TC_{z_{ij},R_{ij},DP_{ij}}$ is not favorable when there is capacity flexibility.

As can be seen from Figure 4.13b, $RT_{12} + MWT_1 + p_{12} = 58.25$ falls in period 3 and therefore $FP'_{121} = 3$. Because the operation cannot be fit in period 4, $FP_{121} = 5$ and $DP_{12} = 5$. Note that $DP_{12} = 5$ has become lower than $ADP_{12} = 6$ as mentioned above.

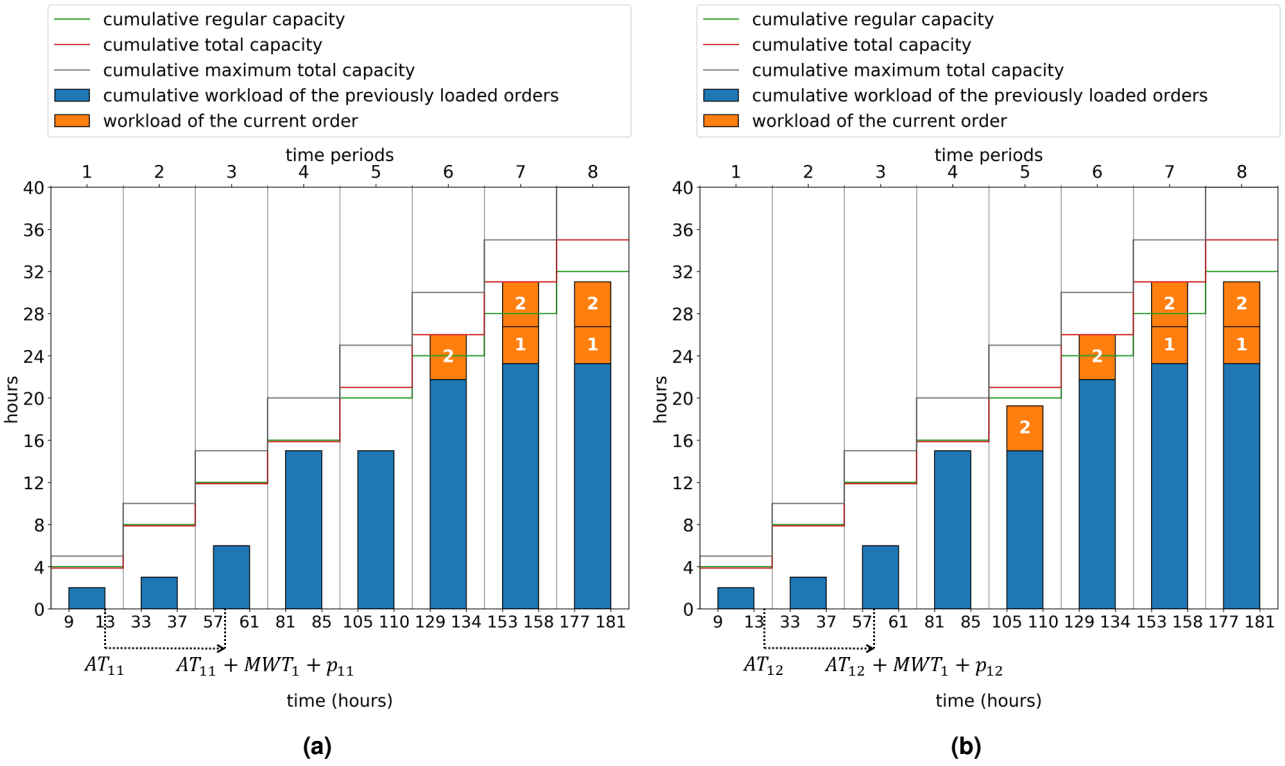


Figure 4.13: Reloading (a) operation (1,1) and (b) operation (1,2) in Example 2

The examples have shown that setting DT_{ij} differently depending on whether $FP_{i,j,R_{ij}}$ (or DP_{ij}) equals $FP'_{i,j,R_{ij}}$ or setting it depending on $TC_{z_{ij},R_{ij},DP_{ij}}$ is not favorable. In order to obtain more stable due times, we set DT_{ij} to $SW_{z_{ij},R_{ij},DP_{ij}} + 0.9 * TC'_{z_{ij},R_{ij},DP_{ij}}$ regardless of whether $FP_{i,j,R_{ij}}$ (or DP_{ij}) equals $FP'_{i,j,R_{ij}}$ or not, where TC'_{klt} equals the value of TC_{klt} right before the order is loaded. TC'_{klt} is updated with TC_{klt} once an order is definitely loaded. The choice of 0.9 instead of, for example, 1 in our definition of $DT_i j$ is meant to avoid overestimation of order flow times. Initial experiments have shown that 0.9 is a reliable choice.

4.5 Methods for determining a loading sequence

As mentioned in Section 3.6, heuristics are known to provide 'good' solutions in a small amount of time to optimization problems, particularly permutation-based problems. Determining a sequence in which to load the orders under consideration during the order acceptance can be seen as a permutation-based optimization problem and therefore heuristics are promising methods. In this section, we explain which heuristics we choose for determining a loading sequence. A solution in this section refers to a

loading sequence. An important factor we take into account to choose the methods is that the loading procedure and evaluate its total cost requires quite some computational time and we look for a method that can provide a good solution in minutes. For example, for a solution with 25 orders it takes 0.1 seconds on average to evaluate it. It is worth noting that according to Limis 5 minutes is the maximum acceptable time to solve the order acceptance problem for Limis' customers.

Given the limited number of solutions that can be evaluated within 5 minutes, we need to choose such improvement heuristics that focus on intensification as opposed to diversification. Local search (LS) is a relatively simple improvement heuristic that focuses solely on intensification. Although it can get stuck in a local optimum, it can still provide a good solution in a relatively short time. Hence, we consider it.

Population-based metaheuristics in general have a higher chance of missing 'good' solutions while being so close to them especially due to the high level of randomness involved in recombination operators (Blum & Roli, 2003). Therefore, we eliminate Genetic Algorithm since it is a population-based metaheuristic. GRASP requires running LS multiple times, each time with a different initial solution. Since a single run of local search may take minutes, it is not desirable to use GRASP. Simulated Annealing focuses on diversification at the beginning of the procedure and therefore is not likely to provide a high-quality solution when terminated early. The basic Tabu Search (TS) can be seen as an extension of LS which does not get stuck in a local optimum while focusing on intensification. Therefore, we choose TS as the only metaheuristic.

We build an initial solution for both LS and TS as follows. We sequence the existing orders based on their due dates and then sequence the incoming orders based on their due dates. We join the two sequences to form an initial solution and we refer to the solution as the earliest due date (EDD) solution. The EDD solution is also used as the starting point for the constructive heuristic described below. We build the initial solution in this way to ensure that the external due dates of existing orders do not have to extended, as will be explained in Section 5.1.3.

In addition to LS and TS, we also consider a constructive heuristic (CH). CH is an adaptation of a well-known constructive heuristic proposed by Nawaz, Enscore, and Ham (1983) for the permutation flow shop scheduling problem in which the orders are processed in the same sequence on all the machines of the flow shop. The procedure starts by building the EDD solution. The first two orders in the EDD solution are selected and the minimum-cost (partial) sequence of the two orders is found. Next, the third order in the EDD solution is selected and tested in the three possible positions (i.e. at the beginning, middle, and end of the partial sequence) preserving the relative positions of the first two orders. The minimum-cost insertion of the third order is applied and the partial sequence is updated. The relative positions of the orders in a partial sequence are preserved in the following iterations. The procedure continues until a full loading sequence is obtained. When ties occur between the two possible partial sequences in the first iteration, the order with the earliest due date is placed in the first position. When ties occur among the positions in which an order can be inserted, it is inserted in the latest position. Orders that are loaded early typically do not experience any shortage in capacity and therefore can be inserted in any position with zero cost but are inserted at the end of the partial sequence according to the above-mentioned tie-breaking rule. Therefore, this heuristic will behave like the EDD heuristic until all the existing orders are loaded and then start to insert the incoming orders in between the existing orders if it costs less. This heuristic is likely to result in a better solution as it evaluates $\frac{N(N+1)}{2} - 1$ solutions, where N is the number of the orders under consideration. For the same reason, it will require higher solution time than EDD. CH can technically result in a worse

solution than the EDD solution. If that is ever the case, we can replace the resulting solution with the EDD solution. It is worth noting that CH is a non-parametric heuristic, which is its main advantage.

Configuring LS and TS

We use the maximum solution time as the stopping criterion for both TS and LS. In TS, this is the only stopping criterion, whereas LS may stop earlier if a local minimum is found. We test different values of maximum solution time as will be explained in Section 5.3. For both LS and TS, we choose swap as the neighborhood structure and first improvement as the neighborhood search strategy as this combination performs much better than the other combinations, i.e. best improvement with swap/insert, first improvement with insert. Note that TS originally is based on best improvement strategy. However, best improvement strategy has poor performance because it is typically possible to search the entire neighborhood within 300 seconds only once or twice, whereas with first improvement strategy more iterations take place. In first improvement strategy, we need to decide how to search the neighborhood. After experimenting with several ways of searching, we decide to search the swap neighborhood simply as follows because of better performance. First, we choose the element in $1st$ position (i.e. the last element) of the current solution, we swap it with the element in $2nd$ position, then with that in $3rd$ position, etc. Next, we choose the element in $2nd$ position and swap it with that in $3rd$ position, etc.

We set the tabu tenure to $0.2 * N$, where N is the number of the orders under consideration. Since not many iterations take place within 300 seconds, the results are insensitive to this parameter. We store in a tabu list the pair of elements being swapped, as it is relatively fast to check the tabu classification of this attribute. In TS, we also use the common aspiration criterion which allows a tabu move if it leads to a better solution than the best one.

4.6 Conclusion

We propose to use a finite loading-based method to solve the order acceptance problem taking into account due date flexibility and overtime. We extend CFFL to account for capacity flexibility (i.e. overtime), orders with assembly-type routings, workstations with multiple resources which have varying work shifts. The proposed procedure to load a sequence of orders provides operation due dates as output among other things and operation due dates are supposed to be used by the ODD priority dispatching rule to make detailed schedules in line with the planned capacity. In other words, the procedure replaces the rough planning (i.e. backward infinite loading) that is currently used in Limis Planner to generate operation release and due dates. We also consider three heuristics to determine a sequence of orders during the order acceptance based on solution time limitation.

5 Experimental Design and Results

In this chapter, we present the design and results of experiments. Section 5.1 describes the design of two types of experiments we conduct. Section 5.2 describes the results of the simulation experiments we conduct to measure the accuracy of and configure the loading procedure. Section 5.3 describes the results of the experiments we conduct to compare the methods to determine a loading sequence. Section 5.4 provides the summary of the chapter.

5.1 Experimentation

We conduct experiments on a computer with Intel Core i7-9750H running at 2.6 GHz with a 16 GB of RAM. We implement the simulation model as well as the solution methods using the Python programming language. We conduct two types of experiments. In Section 5.1.1, we explain how we generate data regarding orders and regular capacity of resources. In Section 5.1.2, we explain how we conduct simulation experiments to measure the accuracy of and configure the loading procedure. In Section 5.1.3, we explain how we conduct experiments to compare the heuristics for determining a loading sequence.

5.1.1 Data generation

In Section 2.2, we present the descriptive statistics of the company whose dataset we use for experimentation. The dataset includes information regarding production orders (arrival times, routings, processing and setup times, due dates) and resources (capacities, number of machines). Because the dataset is not large enough to run all the simulation experiments, we generate data based on these original dataset as follows.

The chosen Limis' customer receives customer requests which typically include multiple products and therefore result in multiple production orders with unique characteristics (i.e. routings, processing times, etc.). In the original dataset, it is not possible to track the customer orders to their customer requests and therefore we assume that the orders arriving on a certain day all belong to a single customer request, which is often the case in reality. Then, we find the empirical distributions for the interarrival times of customer requests and for the number of orders in a customer request.

To determine the type of routing for an order, we use the percentages of string-type and assembly-type routings in the original dataset as the probability of them occurring. We determine the number of operations for string-type and assembly-type orders through empirical distributions derived from the orders having a string-type and assembly-type routing respectively. We also derive an empirical distribution for the number of predecessors of an operation from the operations (excluding starting operations) of assembly-type orders. Figure 5.1 depicts the procedure to generate an assembly-type routing given the required number of operations n_i.

We determine the processing time of each operation through an empirical distribution derived from all the operations in the original dataset. We randomly assign a resource (machine) to each operation,

where the probability of a resource to be chosen is proportional to its frequency in the original dataset. We exclude 10 workstations as they are used very few times in the dataset and focus on 13 main workstations.

The original dataset also includes the order due dates from which we find the due date allowances, i.e. due date minus arrival date. By dividing the due date allowance of each order by its critical path length, we obtain the due date slack factors. The critical path length of an order is the minimum time it would require to complete the order if no waiting time is incurred for the operations. Then we find the empirical distribution of these due date slack factors. We assign a due date to each generated order by multiplying its critical path length by a randomly selected due date slack factor.

Finally, the regular capacities of resources are always the same as in the original dataset.

```
GenerateAssemblyRouting(n_i)

ε = 1     \\current number of operations
j = n_i     \\current operation
l = n_i     \\last generated operation
L = {n_i}  \\set of lowest-level operations
While ε < n_i:
        δ = the number of predecessors of j generated randomly from
              the empirical distribution
        If ε + δ > n_i:
                δ = ε + δ − n_i  \\decrease δ so that the total number of
                                 \\operations does not exceed n_i
    P_ij = {l − 1, … , l − δ}
    ε = ε + δ
    l = l − δ
    L = L\{j}
    L = L ∪ {l − 1, … , l − δ}
    j = a randomly selected operation in L
```

Figure 5.1: Procedure to generate an assembly-type routing

5.1.2 Simulation experiments

First, we conduct simulation experiments to measure the accuracy of the loading procedure as well as to configure its planning parameters, namely the due date buffer, capacity loading limit, and minimum waiting time. We are interested to know the accuracy of assigned internal due dates and therefore we measure the *average* and *standard deviation of lateness* of orders, where lateness of an order is the difference between its completion date and internal due date. We also measure the *utilization of activated overtime* in the whole shop in order to see the timeliness of activated overtime or whether overtime is activated in periods of actual need. In order to improve the customer service level, we measure the *average* and *standard deviation of tardiness* of orders, and the *percentage of tardy orders*, where the tardiness of an order is the positive difference between its completion date and external agreed-upon due date.

Figure 5.2 shows the flowchart of the simulation experiments. In the figure, the order acceptance method refers to the loading procedure.

At the time of a customer request arrival, the input parameters regarding incoming orders are set as

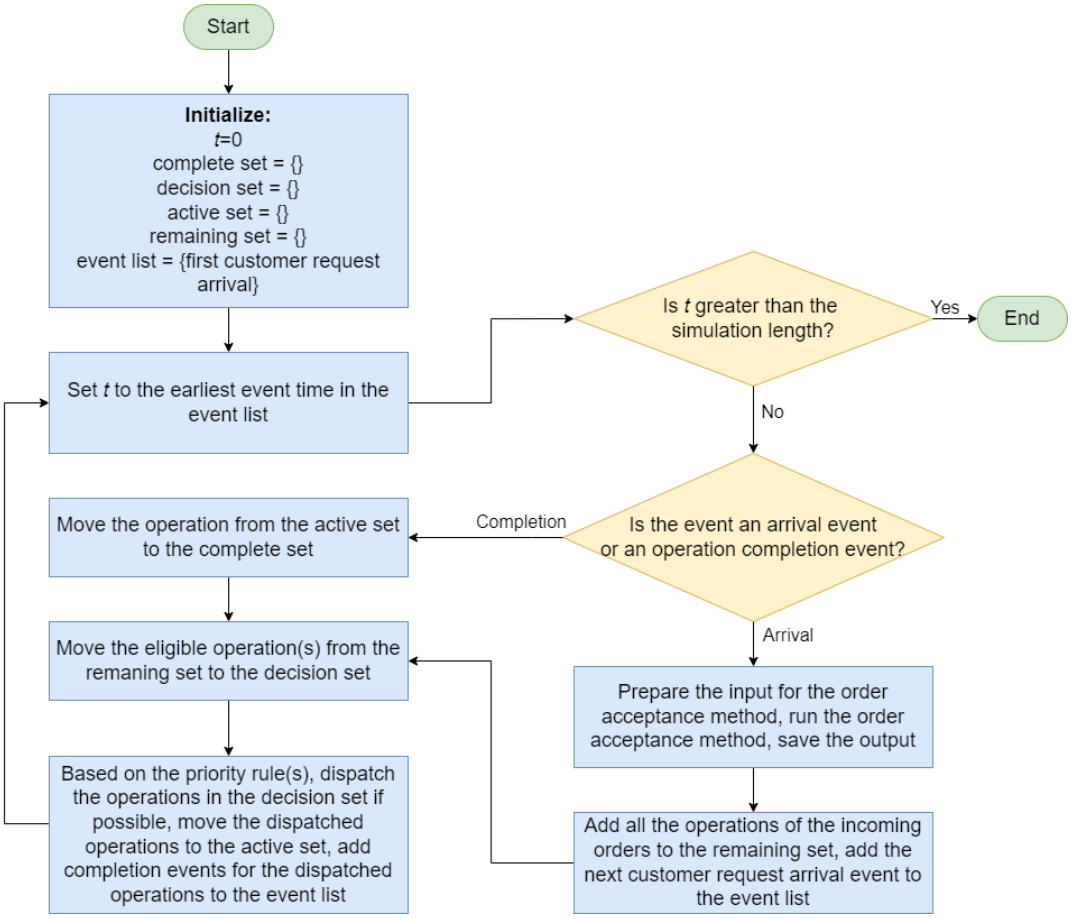


Figure 5.2: Flowchart of a simulation experiment

described in Section 5.1.1. In order to reveal the true performance of the procedure, we do not replan existing orders when a new customer request arrives and we commit to overtime once it is activated. In order to be able to conduct experiments for a sufficient length in reasonable time, we determine S by sequencing incoming orders simply based on their due dates. We assign a common due date buffer DDB' to all incoming orders. Ideally, when the internal due dates are highly accurate, i.e. the average and standard deviation of lateness lie close to zero, the due date buffer can be set equal to zero. The customer-requested due date of an incoming order is extended as long as required, when it cannot be met during the acceptance. Operations are dispatched using the ODD rule and ties are broken using the SPT rule explained in Section 2.3.2.

To build the total cost function, we assign a significantly higher cost of due date extension per day β_i for incoming orders than the overtime cost per hour γ_k for resources. Note that we use the total cost function to compare various loading sequences. Since we consider a single loading sequence of incoming orders based on due dates, the total cost function has no effect in the simulation experiments.

We maintain a parent profile for each resource which stores the cumulative workload, regular capacity, active overtime, and maximum overtime for each day of the whole simulation. After each instance of order acceptance, each operation of each incoming order is loaded on the parent profile of the chosen resource starting from its due date. Operations remain loaded on the parent resource profiles until they are complete. Note that operations may be completed before or after their due date and they may be performed on a resource different from the one on which they are loaded. Also after each instance

of order acceptance, we update active overtime on the parent resource profiles.

Suppose that a new customer request arrives on day t^* of the simulation. First, we derive a child profile for each resource which is a copy of the parent profile from day t^* to day $t^* + T$, where T is the length of the rolling planning horizon. Suppose that the latest day on which an existing operation is planned to finish is day f of the simulation. T should be at least $f - t^*$ and it should be high enough so that all the incoming orders can be loaded within the planning horizon. We set T to $\max(f - t^*, 100)$. Periods are numbered from 1 to T in the child profiles.

Cumulative workload CWL'_{klt} of the child resource profiles includes four types of existing operations: 1) operations that are planned to finish before day t and have not started, 2) operations that are planned to finish before day t and are in progress, 3) operations that are planned to finish on and after day t and have not started, 4) operations that are planned to finish on and after day t and are in progress. The (remaining) workload of operations of type 1 and 2 constitute *explicit backlog*. Note that due to the nature of CFFL, we do not know when operations are supposed to start and we only know when they are supposed to finish. Therefore, it might be that part of the workload belonging to operations of type 3 and 4 should have been completed already, which we refer to as *implicit backlog*. As a result of explicit and implicit backlog, in the child profiles there can be overload in some periods, i.e. $CWL'_{klt} - CLL_k * CTC_{klt} > 0$. Therefore, we update cumulative workload of child profiles in 4 steps.

1. We unload operations of type 1 starting from period 1 and add their workload to the backlog of the resources on which they are loaded.
2. We unload operations of type 2 starting from period 1. We load the remaining workload of each of those operations as if we load an operation as described in Section 4.4.1 but assuming no cumulative workload on the resource on which they are being processed. Similarly, we unload operations of type 3 starting from their due period and load their remaining workload in the same way.
3. For each resource (k,l), we find the workload that should have been completed as follows, $\max_t (CWL_{klt} - CLL_k * CTC_{klt})^+$ and add the amount to the backlog of the resource.
4. For each resource, we load the backlog as if we load an operation as described in Section 4.4.1.

In the resulting child profiles, there is no overload in any period and cumulative workload CWL'_{klt} accurately includes both explicit and implicit backlog. Reloading of backlog is important to ensure that incoming orders that are loaded on top of the existing orders do not impose additional lateness to existing orders.

5.1.3 Experiments to compare the heuristics

Once an appropriate set of values are determined for the planning parameters of the loading sequence through simulation experiments, we run experiments to compare the heuristic methods mentioned in Section 4.5 for determining a loading sequence. Using these methods to (re)plan incoming orders (and existing orders) at each instance of order acceptance in a whole simulation experiment requires a lot of computational time and is also not necessary to determine their relative performances.

Therefore, we run a simulation experiment as described above until a certain instance of order acceptance and at that point we use these methods to (re)plan incoming and existing orders. We then terminate the simulation at that point. So, in the chosen instance of order acceptance, the loading

procedure is used as a subroutine for the heuristics. The input parameters of the loading procedure are set as explained in Section 5.1.2 except for cumulative workload CWL'_{klt}.

Replanning an existing order means we reload all the operations that have not been performed yet excluding those that are in progress (i.e. operations of type 2 and 4). Therefore, CWL'_{klt} is first set to zero in each period and on each resource. Next, we reload the remaining workload of each operation of type 2 or 4 as if we load an operation as described in Section 4.4.1 on the resource on which they are being processed.

Until the chosen instance of order acceptance, incoming orders are assigned a common due date buffer DDB' and loaded on top of existing orders. The role of the due date buffer is to compensate for deviations from internal due dates as a result of the detailed scheduling and arrivals of future orders. When the existing orders are replanned in the chosen instance of order acceptance, we also replan the existing operations that might have otherwise constituted explicit and implicit backlog (i.e. operations of type 1 and 3). In other words, we replan also the existing operations that are supposed to be fully or partially complete until that point. Suppose that we reload the existing orders before loading incoming orders but without activating additional overtime. Even in that case, the internal due dates of the existing orders are likely to increase because we also replan operations of type 1 and 3. As a result, the new internal due dates might be too close to (i.e. $DP_{i,n_i} + DDB' > d_i$) or higher than the agreed-upon external due dates resulting in positive due date extension DDE_i. However, given that DDB' is chosen to minimize the likelihood of orders being tardy in the long run, we can safely assume that DDE_i that existing orders incur when they are reloaded even before incoming orders, for example, based on due dates, and without using additional overtime is acceptable. Therefore, in the chosen instance of order acceptance, we reload the existing orders before the incoming orders based on due dates and without using additional overtime and assign to each of them a unique DDB_i as follows, $DDB_i = \min(DDB', (d_i - DP_{i,n_i}))$.

To build the total cost function, we assign a significantly higher cost of due date extension per day for existing orders than for incoming orders and we assign a significantly low value (i.e. much lower than the due date extension cost of incoming orders) to the overtime cost per hour. Using the updated due date buffers, the EDD solution described in Section 4.5 would always become ”feasible” in the sense that the due dates of existing orders are not extended. The goal is then to find such a sequence of the incoming and existing orders together that minimizes the total due date extension of the incoming orders. Because of the definition of the total cost function, the minimum-cost sequence is likely to be the one which avoids due date extension of existing orders and minimizes due date extension of incoming orders using as much overtime as required. In addition, we should be able to find a good sequence in a reasonable amount of time so that the proposed solution approach can be considered a fast method. Hence, we measure the *solution time* and *the total due date extension of incoming orders*.

5.1.4 Determining simulation parameters

In order to run simulation experiments as described in Section 5.1, we need to determine the warm-up period, the length of simulation, and the number of replications. In the simulation experiments, we arbitrarily decide to set the maximum overtime in a period on a resource to 25% of the regular capacity. In Section 5.2, we analyze the effect of this parameter on the results.

As a starting point we determine a good configuration of the planning parameters, MWT_k, CLL_k, and DDB'. Based on some initial experiments we observe that the workstations in the company dataset can be divided into groups based on their utilization and we assign the common set of values for MWT_k and CLL_k for the workstations in a group, as shown in Table 5.1. In the table, the utilization level of each workstation represents the average over 10 replications with no warm-up period and a simulation length of 1000 days. We set the due date buffer to 6 days as it results in a sufficiently low percentage of tardy orders. Note that the higher is the due date buffer, the lower is the percentage of tardy orders.

Table 5.1: Initial configuration of two planning parameters based on utilization

Workstation	2	3	7	4	8	5	6	1	10	11	9	12	13
$MWT(hours)$	0			2		3		4			5		
$CLL(\%)$	100			95		95		90			100		
Utilization (%)	5	7	2	14	22	32	38	47	50	44	53	59	53

To determine the warm-up period, we use the well-known Welch's procedure (Law, 2015). First, we run 10 replications each with a length of 1000 days. In each experiment, we save the flow times of orders as they finish processing, where the flow time of an order is the difference between its completion date and arrival date. We believe order flow times are a good indicator to observe the transient behavior of the system. At least 1071 orders are completed over the replications. Then, we find the average over 10 replications of the flow times of the first order, second order, etc. Finally, we find the moving averages of the averages with a window of 100 to smooth out the oscillations in the averages, as described by Law (2015). Figure 5.3 depicts the resulting moving averages plotted against the orders. As can be seen from the figure, nearly after 130th order, the system enters a steady state. We choose a slightly more conservative number of orders, 150, which nearly corresponds with day 150 in the simulation. So, we set the warm-up period to 150 days.

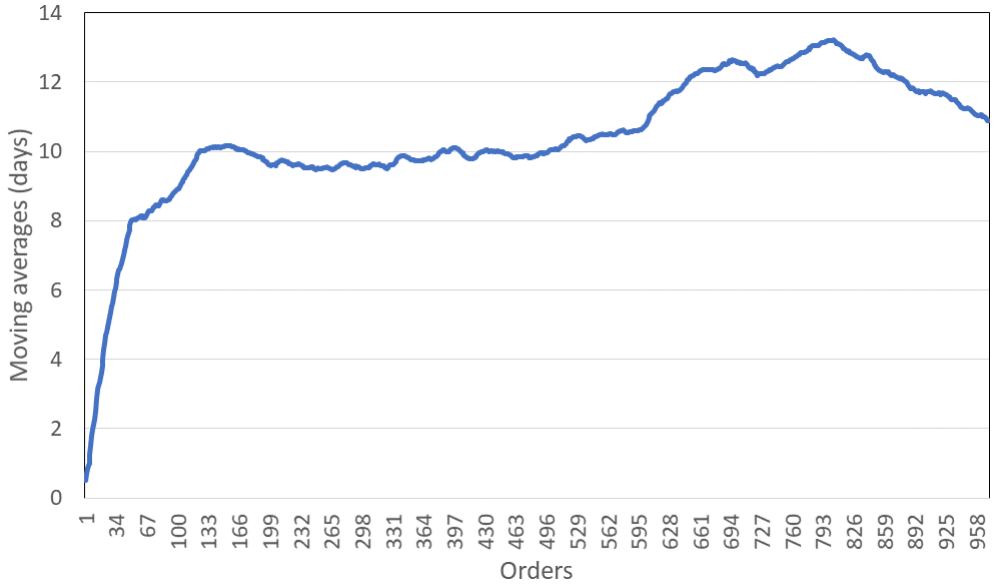


Figure 5.3: Determining the warm-up period

We set the simulation length to 1000 days as it is sufficiently larger than the warm-up period. To determine the number of replications, we use the Sequential procedure presented by Law (2015).

First, we run 5 replications, as suggested and find the average order flow time for each replication which are shown in Table 5.2. The averages together form a sample of size $n = 5$. We find the sample mean $\bar{X} = 10.22$ and sample variance $S^2 = 0.31$. Then we find the half-width of the confidence interval with a confidence level of $100(1-\alpha) = 90\%$ relative to the sample mean,

$$\frac{t_{n-1,1-\alpha/2}\sqrt{S^2/n}}{\bar{X}} = 0.052 \tag{2}$$

where $t_{n-1,1-\alpha/2} = 2.13$ is the upper $1-\alpha/2$ critical point for the *t* distribution with n-1 degrees of freedom. Next, we find the *adjusted desired relative error* $\gamma' = \gamma/(1-\gamma) = 0.11$, where $\gamma = 0.1$ is the desired relative error. Since 0.052 is smaller than 0.11, we conclude that 5 replications is sufficient. Otherwise, we would need to run additional replications until the half-width of the confidence interval relative to the sample mean is smaller than the adjusted desired relative error. Note that our choices of the confidence level and the relative error are in line with the suggestions by Law (2015). The reader is referred to Law (2015) for the details.

Table 5.2: Average order flow times found in 5 replications

Replication	1	2	3	4	5
Average flow time (days)	9.37	10.02	10.48	10.39	10.85

5.2 Results of simulation experiments

After determining the warm-up period, simulation length, and number of replications, we conduct experiments with various configurations. Based on our observations in the initial experiments, we consider a single value for MWT_k and CLL_k of 3 groups of workstations. These workstations have a low utilization as can be seen from Table 5.1 and therefore changes in their parameters do not affect results significantly. For the two groups of workstations with a high utilization, we consider 2 values for both MWT_k and CLL_k. We set the due date buffer to 6 days. A lower value of the due date buffer results in poor tardiness performance, whereas a higher value has negligible influence on the results. In total, we consider 16 experiments resulting from the possible values of the planning parameters shown in Table 5.3. The configuration of the planning parameters for each experiment is shown in Appendix A.

Table 5.3: Tested values of the planning parameters

Workstation	2 3 7	4 8	5 6	1 10 11	9 12 13
$MWT(hours)$	0	2	3	4 ; 5	5 ; 6
$CLL(\%)$	100	95	95	90 ; 95	95 ; 100

Table 5.4 shows the average results of the experiments over the replications. As can be seen from the table, the shop utilization is the same in all the experiments and the experiments differ only slightly in the other performance measures. The high levels of overtime utilization indicate that the procedure accurately activates overtime in periods of actual need. The same cannot be said about the assigned

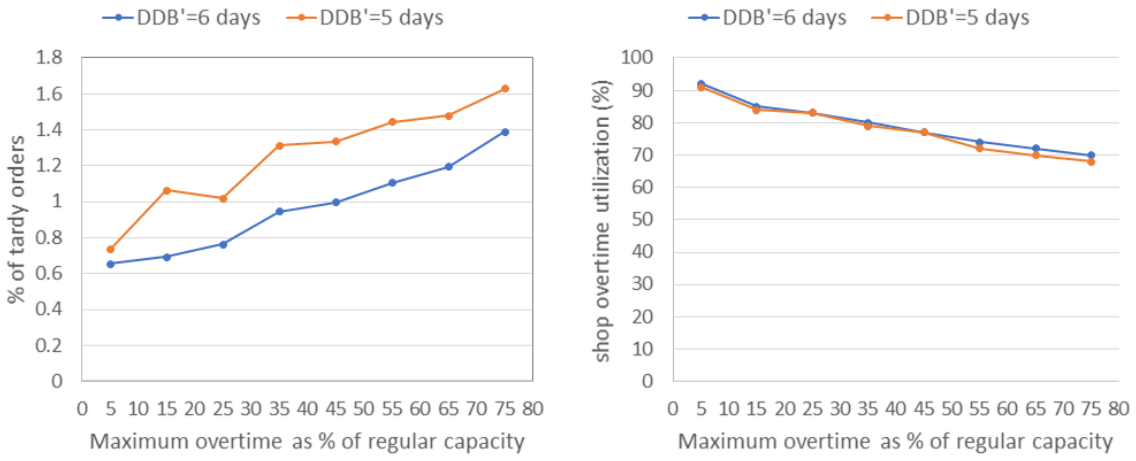
internal due dates. Although the average lateness is near zero in all the cases, the standard deviation of lateness is extremely high. We can attribute such a high variance partly to the high variance of operation processing times. The high variance of lateness also justifies our choice of a high value for the due date buffer (i.e. 6 days) in order to obtain high tardiness performance. The average and the standard deviation of tardiness as well as the percentage of tardy orders are close to zero in all the experiments. We choose Experiment 12 as it results in the lowest percentage of tardy orders. In the corresponding configuration $MWT_k = 5hrs$ and $CLL_k = 90\%$ for $k = 1, 10, 11$ and $MWT_k = 6hrs$ and $CLL_k = 100\%$ for $k = 9, 12, 13$. In addition, we observe that in all the experiments the customer-requested external due dates of incoming orders are extended less than a day on average. Knowing that incoming orders are planned without replanning existing orders and simply based on external due dates, these figures mean that external due dates are typically so loose that there is no need for replanning existing orders or using an advanced method to determine a loading sequence.

Table 5.4: Results of simulation experiments

Exp	% of tardy orders	average tardiness (days)	st dev of tardiness (days)	average lateness (days)	st dev of lateness (days)	shop overtime utilization (%)	shop utilization (%)	avg due date extension of incoming orders (days)
1	1.03	0.030	0.35	-0.14	2.79	80	34	0.53
2	0.93	0.028	0.32	-0.25	2.76	81	34	0.55
3	1.01	0.028	0.33	-0.23	2.77	80	34	0.54
4	0.93	0.028	0.33	-0.20	2.76	82	34	0.54
5	0.86	0.024	0.31	-0.02	2.70	82	34	0.56
6	0.89	0.024	0.31	0.05	2.60	83	34	0.53
7	0.89	0.022	0.30	0.03	2.69	82	34	0.54
8	0.93	0.026	0.34	0.02	2.65	83	34	0.55
9	1.06	0.032	0.36	-0.07	2.69	80	34	0.53
10	0.89	0.024	0.30	-0.02	2.75	84	34	0.53
11	1.01	0.028	0.33	-0.13	2.73	80	34	0.55
12	0.76	0.022	0.30	-0.16	2.75	83	34	0.56
13	1.04	0.028	0.33	-0.13	2.70	80	34	0.54
14	0.91	0.026	0.32	-0.07	2.81	82	34	0.55
15	1.01	0.030	0.36	-0.10	2.70	81	34	0.54
16	0.80	0.026	0.32	-0.08	2.72	84	34	0.54

We conduct further experiments using the chosen configuration of MWT_k and CLL_k in order to observe the effect of the maximum overtime on the performance. The maximum overtime is still expressed as a percentage of the regular capacity which ranges from 5% to 75%. Also we consider two values for the due date buffer, 5 and 6 days. Figure 5.4a depicts the percentage of tardy orders against the maximum overtime and Figure 5.4b depicts the overtime utilization against the maximum overtime. As can be seen from the figures, increasing the maximum overtime negatively affects both performance

measures with both values of the due date buffer, which we cannot explain. Although the changes in the performance measures are not so significant, this observation indicates that the other planning parameters MWT_k and CLL_k have to be slightly modified depending on the level of maximum overtime. We also observe that the due date buffer has almost no influence on overtime utilization and is only useful to improve tardiness performance.



(a) **(b)**

Figure 5.4: Effect of maximum overtime on two performance measures

5.3 Results of experiments to compare the heuristics

As explained in Section 5.1.3, we compare the methods mentioned in Section 4.5 in 5 instances of order acceptance during simulation runs. We choose instances where the EDD solution results in a non-zero total due date extension for incoming orders. In other words, there is room for improvement in the chosen instances, which is not always the case as customer-requested external due dates are typically loose for the chosen company. Moreover, in the chosen instances, the number of (existing and incoming) orders is high, which allows us to test the worst-case performance in terms of solution time. For the configuration of the loading procedure, we use Experiment 12 which was found to be the best configuration in Section 5.2.

The maximum acceptable solution time for Limis' customers is 5 minutes. We limit the maximum solution time of TS to 5 minutes and 10 minutes. LS naturally stops when a local minimum is found. In addition, we test LS with a maximum solution time of 5 minutes. We cannot limit the solution time of CH because a full solution is obtained when the method runs until the end.

Table 5.5 shows the results for each instance and for each method. For example, LS-5 represents LS with a time limit of 5 minutes, while LS represents LS without a time limit. We do not publish the solution times for the EDD solution, as it typically runs in less than a second. On average, CH and TS-10 slightly outperform LS, LS-5, and TS-5 in terms of due date extension. In terms of solution time, CH significantly outperforms all the other methods. The due date extension results of LS, LS-5 and TS-5 are identical. As can be observed from the solution times of LS-5 and LS, a local minimum is reached earlier than 5 minutes in Instances 2, 4, and 5 and no later than 9 minutes in Instances 1

and 3. By nature, TS behaves like LS until a local minimum is reached. Although a local minimum is found earlier than 10 minutes in all the instances, TS-10 slightly improves upon LS only in Instance 2. It might be because the local minima found in Instances 1, 3, 4, and 5 are optimal solutions, which we cannot check. Overall, CH proves to be a promising method for determining a loading sequence because of its ability to produce good solutions in significantly less time.

Table 5.5: Results of methods for determining a loading sequence

Instance	**num of existing orders**	**num of incoming orders**	**total due date extension of incoming orders (days)**						**solution time (seconds)**				
			EDD	CH	LS-5	TS-5	LS	TS-10	CH	LS-5	TS-5	LS	TS-10
1	36	25	56	19	19	19	19	19	195	300	300	481	600
2	18	22	14	11	12	12	12	11	55	197	300	197	600
3	23	25	15	5	5	5	5	5	93	300	300	378	600
4	35	7	11	8	8	8	8	8	60	136	300	136	600
5	23	14	19	6	6	6	6	6	44	138	300	138	600
Average			23	9.8	10	10	10	9.8	89.4	214.2	300	266	600

5.4 Conclusion

We observe that overtime is activated in a timely manner by the loading procedure, although there is a significant variance of lateness with regard to assigned internal order due dates, which makes us use a high due date buffer for incoming orders in order to achieve good long-run tardiness performance. We also observe that because the chosen Limis' customer receives orders with loose customer-requested due dates, often there is no need for replanning existing orders or for finding a good sequence of orders. Comparison of the methods to determine a loading sequence reveals that the constructive heuristic is a promising method that performs equally with local search and tabu search but requires much less solution time.

6 Conclusions and Recommendations

In this chapter we present our conclusions and recommendations for future research.

Conclusions

This research was motivated by a challenge faced by Limis' customers. Limis' customers need to respond to and act on dynamically arriving customer requests in an MTO or hybrid MTS/MTO environment. In other words, the company (i.e. Limis' customer) needs to determine whether they can deliver the requested quantity by the requested due date for an incoming order based on material and capacity availability. If the company is not able to do so, they have three types of flexibility options to consider, namely, due date flexibility of incoming orders and some selected existing orders, capacity flexibility by means of overtime, subcontracting, and quantity flexibility (i.e. fulfilling an incoming order by partial shipments). The order acceptance method preferably considers these flexibility options. The currently used scheduling function of Limis Planner does not consider capacity and due date flexibility and assumes flexibility of all order due dates. In addition, it requires a high solution time causing a high customer response time. The main research question of this thesis was as follows.

How can a fast and reliable order acceptance method be designed within Limis Planner in order to respond to and act on dynamically arriving customer orders?

First, we analyzed the context with regard to how the scheduling function of Limis Planner works and more importantly the operating characteristics of Limis' customers. When designing a solution to address the order acceptance problem, we had to take into account that Limis' customers typically have orders with complex assembly-type routings, workstations with identical multiple resources and resources with varying daily work shifts. Also, we learned that detailed schedules are constructed using priority dispatching rules such as ORD and ODD which use operation release and due dates assigned by backward infinite loading.

Review of the literature revealed that some studies consider the order acceptance problem as a tactical planning problem, which is not applicable in the case of Limis' customers. In the studies approaching the problem as an operational planning problem, some limiting assumptions are made, i.e. orders only have string-type routings, operations can be preempted. However, we benefited from the literature by using a so-called finite loading method as the starting point for solution design and some heuristics commonly used to solve permutation-based problems.

Due to the requirement for a fast solution approach, we decided to use a finite loading method in combination with a heuristic to determine a loading sequence. We chose CFFL as a finite loading method because of its nature that allows to advance an operation by activating overtime without the need to replan previously loaded operations. We extended CFFL to account for capacity flexibility (i.e. overtime, subcontracting) and the operating conditions of Limis' customers, namely, orders with assembly-type routings, workstations with multiple resources which have varying work shifts. In addition, we used a third planning parameter called due date buffer which is to obtain high tardiness performance.

The results of the simulation experiments show that although the procedure successfully activates overtime in periods of actual need, there is a significant variance of lateness with regard to assigned order due dates. Therefore, we recommend to use a due date buffer of 6 days for the chosen Limis'

customer in order to achieve high tardiness performance. Also, we recommend to group the workstations based on their utilization and assign a common set of values per group for the other two planning parameters, namely, minimum waiting time and capacity loading limit. This reduces the effort to configure the method. Experiments with varying levels of maximum overtime show that it might be required to adjust the planning parameters of minimum waiting time and capacity loading limit slightly depending on the level of maximum overtime. Another observation is that the chosen Limis' customer typically does not need to replan existing orders or use an an advanced method to determine a sequence of orders during the order acceptance because of loose customer-requested due dates.

We decide to consider three methods for determining a loading sequence based on the solution time requirement. The constructive heuristic (CH) performs equally with local search and Tabu search in terms of total due date extension of incoming orders, but achieves its results in much less time. Another advantage of CH is that it is a non-parametric method, which makes it easy to implement for other Limis' customers. The limitation of CH is that it can lead to a solution worse than the EDD solution which is used as the starting point. We recommend to use CH to sequence orders and replace its final solution with the EDD solution if the EDD solution is ever better.

Overall, we can say that the proposed loading procedure provides reliable output in terms of timing of overtime and due date performance. In addition to reliability, we can obtain more efficient solutions in terms of due date extensions and amount of used overtime by replanning existing orders along with incoming orders. By using CH to determine a loading sequence, we can obtain efficient solutions within a shorter solution time. So, we recommend that Limis implement the loading procedure in combination with CH.

Note that we did not consider quantity flexibility and focused on due date and capacity flexibility due to the limited time available for this research. The proposed order acceptance method (i.e. the loading procedure in combination with CH) supports the planner in deciding which orders to extend and how much, in which periods on which resources to use overtime and how much, which operations to subcontract and when. The output of the order acceptance method not only is used to respond to a customer request or take actions for overtime and subcontracting but also is used as an input for detailed scheduling as operation due dates.

It is worth noting that this research also contributes to the theory by extending a finite loading method to consider capacity flexibility and other practical aspects including orders with assembly-type routings, workstations with multiple resources which have varying work shifts. Also note that in the studies using finite loading methods existing orders are not replanned. We have shown that replanning existing orders along with incoming orders is promising.

Recommendations for future research and limitations

The main limitation of this research is that we experiment using the dataset of a single Limis' customer due to the time limitation of this research. Because the loading procedure is a parametric procedure, it is important to run experiments and configure the procedure for other companies.

Also, we assumed deterministic conditions in this research. Limis Planner provides an option to increase processing times in order to build robustness into detailed schedules. It is worth studying the effects of stochastic conditions such as operation processing time variability on the performance measures. Since the planning parameters of the loading procedure are also meant to build buffer into plans, it might be interesting, for example, to set them more conservatively instead of increasing operation

processing times.

References

Ahmed, I., & Fisher, W. W. (1992). Due Date Assignment, Job Order Release, and Sequencing Interaction in Job Shop Scheduling. *Decision Sciences*, *23*(3), 633–647. https://doi.org/10.1111/j.1540-5915.1992.tb00409.x

Akkan, C. (1996, September). Overtime Scheduling: An Application in Finite-capacity Real-time Scheduling. *Journal of the Operational Research Society*, *47*(9), 1137–1149. https://doi.org/10.1057/jors.1996.142

Ashayeri, J., & Selen, W. J. (2001, October). Order selection optimization in hybrid make-to-order and make-to-stock markets. *Journal of the Operational Research Society*, *52*(10), 1098–1106. https://doi.org/10.1057/palgrave.jors.2601204

Ball, M. O., Chen, C.-Y., & Zhao, Z.-Y. (2004). Available to Promise. In D. Simchi-Levi, S. D. Wu, & Z.-J. Shen (Eds.), *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era* (pp. 447–483). Boston, MA: Springer US.

Bertrand, J. (1983). The use of workload information to control job lateness in controlled and uncontrolled release production systems. *Journal of Operations Management*, *3*(2), 79–92. https://doi.org/10.1016/0272-6963(83)90009-8

Blum, C., & Roli, A. (2003, September). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, *35*(3), 268–308. https://doi.org/10.1145/937503.937505

Brucker, P., Hurink, J., & Werner, F. (1996). Improving local search heuristics for some scheduling problems—I. *Discrete Applied Mathematics*, *65*(1), 97–122. https://doi.org/10.1016/0166-218X(95)00030-U

Deroussi, L., Gourgand, M., & Norre, S. (2006). *New effective neighborhoods for the permutation flow shop problem* (Tech. Rep.). https://hal.archives-ouvertes.fr/hal-00678053

Ebben, M., Hans, E., & Weghuis, F. M. O. (2005). Workload based order acceptance in job shop environments. *OR Spectrum*, *27*, 107–122. https://doi.org/10.1007/s00291-004-0171-9

Feo, T. A., & Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, *6*(2), 109–133. https://doi.org/10.1007/BF01096763

Framinan, J. M., Gupta, J. N. D., & Leisten, R. (2004, December). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, *55*(12), 1243–1255. https://doi.org/10.1057/palgrave.jors.2601784

Giebels, M. (2000). *EtoPlan: a concept for concurrent manufacturing planning and control - Building holarchies for manufacture-to-order environments* (Ph. D. thesis). University of Twente, Enschede, the Netherlands.

Glover, F. (1986, January). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533–549. https://doi.org/10.1016/0305-0548(86)90048-1

Hans, E. (2001). *Resource Loading by Branch-and-Price Techniques* (Ph. D. thesis). University of Twente, Enschede, the Netherlands.

Hendry, L. C., & Kingsman, B. G. (1989, May). Production planning systems and their applicability to make-to-order companies. *European Journal of Operational Research*, *40*(1), 1–15. https://doi.org/10.1016/0377-2217(89)90266-X

Hvolby, H.-H., & Steger-Jensen, K. (2010, December). Technical and industrial issues of Advanced

Planning and Scheduling (APS) systems. *Computers in Industry*, *61*(9), 845–851. https://doi.org/10.1016/j.compind.2010.07.009

Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, *113*(2), 390–434. https://doi.org/10.1016/S0377-2217(98)00113-1

Keskinocak, P., & Tayur, S. (2004). Due Date Management Policies. In D. Simchi-Levi, S. D. Wu, & Z.-J. Shen (Eds.), *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era* (pp. 485–554). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4020-7953-5

Kingsman, B., Hendry, L., Mercer, A., & de Souza, A. (1996, December). Responding to customer enquiries in make-to-order companies Problems and solutions. *International Journal of Production Economics*, *46-47*, 219–231. https://doi.org/10.1016/0925-5273(95)00199-9

Kingsman, B., Tatsiopoulos, I. P., & Hendry, L. C. (1989, May). A structural methodology for managing manufacturing lead times in make-to-order companies. *European Journal of Operational Research*, *40*(2), 196–209. https://doi.org/10.1016/0377-2217(89)90330-5

Kingsman, B., Worden, L., Hendry, L., Mercer, A., & Wilson, E. (1993, July). Integrating marketing and production planning in make-to-order companies. *International Journal of Production Economics*, *30-31*, 53–66. https://doi.org/10.1016/0925-5273(93)90081-U

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kolisch, R. (1996, April). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, *90*(2), 320–333. https://doi.org/10.1016/0377-2217(95)00357-6

Law, A. M. (2015). *Simulation modeling and analysis*. New York: Mcgraw-Hill. (OCLC: 1022581268)

Mestry, S., Damodaran, P., & Chen, C.-S. (2011, June). A branch and price solution approach for order acceptance and capacity planning in make-to-order operations. *European Journal of Operational Research*, *211*(3), 480–495. 2021-07-19https://www.sciencedirect.com/science/article/pii/S037722171100004X doi: 10.1016/j.ejor.2011.01.002

Nandi, A., & Rogers, P. (2004, March). Using Simulation to Make Order Acceptance/Rejection Decisions. *SIMULATION*, *80*(3), 131–142. 2021-07-19https://doi.org/10.1177/0037549704045046

Nawaz, M., Enscore, E. E., & Ham, I. (1983, January). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91–95. https://10.1016/0305-0483(83)90088-9

Olhager, J. (2003, September). Strategic positioning of the order penetration point. *International Journal of Production Economics*, *85*(3), 319–329. https://doi.org/10.1016/S0925-5273(03)00119-1

Philipoom, P. R., & Fry, T. D. (1992, November). Capacity-based order review/release strategies to improve manufacturing performance. *International Journal of Production Research*, *30*(11), 2559–2572. https://doi.org/10.1080/00207549208948176

Potvin, J.-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, *63*(3), 337–370. https://doi.org/10.1007/BF02125403

Robinson, K. R., & Moses, S. A. (2006, December). Effect of granularity of resource availability on the accuracy of due date assignment. *International Journal of Production Research*, *44*(24), 5391–5414. https://doi.org/10.1080/00207540600665810

Roundy, R., Chen, D., Chen, P., Çakanyildirim, M., Freimer, M. B., & Melkonian, V. (2005, December). Capacity-driven acceptance of customer orders for a multi-stage batch manufacturing system:

models and algorithms. *IIE Transactions*, *37*(12), 1093–1105. https://doi.org/10.1080/07408170500288042

Silver, E. A., Pyke, D. F., & Thomas, D. J. (2016). Material Requirements Planning and Its Extensions. In *Inventory and Production Management in Supply Chains* (4th ed., p. 810). Boca Raton: CRC Press.

Thürer, M., & Stevenson, M. (2020, June). The use of finite loading to guide short-term capacity adjustments in make-to-order job shops: an assessment by simulation. *International Journal of Production Research*, *58*(12), 3554–3569. https://doi.org/10.1080/00207543.2019.1630771

Thürer, M., Stevenson, M., Silva, C., & Land, M. (2013, August). Towards an Integrated Workload Control (WLC) Concept: The Performance of Due Date Setting Rules in Job Shops with Contingent Orders. *International Journal of Production Research*, *51*(15), 4502–4516. https://doi.org/10.1080/00207543.2013.774485

van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). Towards implementing the algorithm. In P. J. M. van Laarhoven & E. H. L. Aarts (Eds.), *Simulated Annealing: Theory and Applications* (pp. 55–75). Dordrecht: Springer Netherlands. https://doi.org/10.1007/978-94-015-7744-1

Zijm, W. (2000, August). Towards intelligent manufacturing planning and control systems. *OR Spectrum*, *22*(3), 313–345. https://doi.org/10.1007/s002919900032

Appendix A

Table A.1: Values of the planning parameters used in simulation experiments

Exp	$MWT^{*}(hours)$	$CLL^{**}(\%)$	$DDB'(days)$
1	4;0;0;2;3;3;0;2;5;4;4;5;5	0.9;1;1;0.95;0.95;0.95;1;0.95;0.95;0.9;0.9;0.95;0.95	6
2	5;0;0;2;3;3;0;2;6;5;5;6;6	0.9;1;1;0.95;0.95;0.95;1;0.95;0.95;0.9;0.9;0.95;0.95	6
3	4;0;0;2;3;3;0;2;6;4;4;6;6	0.9;1;1;0.95;0.95;0.95;1;0.95;0.95;0.9;0.9;0.95;0.95	6
4	5;0;0;2;3;3;0;2;5;5;5;5;5	0.9;1;1;0.95;0.95;0.95;1;0.95;0.95;0.9;0.9;0.95;0.95	6
5	5;0;0;2;3;3;0;2;6;5;5;6;6	0.95;1;1;0.95;0.95;0.95;1;0.95;1;0.95;0.95;1;1	6
6	4;0;0;2;3;3;0;2;5;4;4;5;5	0.95;1;1;0.95;0.95;0.95;1;0.95;1;0.95;0.95;1;1	6
7	4;0;0;2;3;3;0;2;6;4;4;6;6	0.95;1;1;0.95;0.95;0.95;1;0.95;1;0.95;0.95;1;1	6
8	5;0;0;2;3;3;0;2;5;5;5;5;5	0.95;1;1;0.95;0.95;0.95;1;0.95;1;0.95;0.95;1;1	6
9	4;0;0;2;3;3;0;2;5;4;4;5;5	0.95;1;1;0.95;0.95;0.95;1;0.95;0.95;0.95;0.95;0.95;0.95	6
10	4;0;0;2;3;3;0;2;5;4;4;5;5	0.9;1;1;0.95;0.95;0.95;1;0.95;1;0.9;0.9;1;1	6
11	5;0;0;2;3;3;0;2;6;5;5;6;6	0.95;1;1;0.95;0.95;0.95;1;0.95;0.95;0.95;0.95;0.95;0.95	6
12	5;0;0;2;3;3;0;2;6;5;5;6;6	0.9;1;1;0.95;0.95;0.95;1;0.95;1;0.9;0.9;1;1	6
13	4;0;0;2;3;3;0;2;6;4;4;6;6	0.95;1;1;0.95;0.95;0.95;1;0.95;0.95;0.95;0.95;0.95;0.95	6
14	4;0;0;2;3;3;0;2;6;4;4;6;6	0.9;1;1;0.95;0.95;0.95;1;0.95;1;0.9;0.9;1;1	6
15	5;0;0;2;3;3;0;2;5;5;5;5;5	0.95;1;1;0.95;0.95;0.95;1;0.95;0.95;0.95;0.95;0.95;0.95	6
16	5;0;0;2;3;3;0;2;5;5;5;5;5	0.9;1;1;0.95;0.95;0.95;1;0.95;1;0.9;0.9;1;1	6

*each of the 13 numbers in a cell below is for the corresponding workstation.

**each of the 13 numbers in a cell below is for the corresponding workstation.