Decision Support for Continuous Casting Planning

A Novel Decomposition of the Continuous Casting Planning Problem

by

Oliver Herr

a Thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy
in International Logistics

Approved Dissertation Committee:

Prof. Dr. Werner Bergholz (Jacobs University)
Prof. Dr.-Ing. Katja Windt (Jacobs University)
Prof. Dr. Asvin Goel (Kühne Logistics University)
Prof. Dr. Frank Meisel (Christian-Albrechts-Universität Kiel)

Mathematics & Logistics

Abstract

The steel industry is among the most competitive industries. Especially European companies are in need to improve their processes in order to survive in the market. The tasks of production planning and control have a major impact on the logistics target achievement and therefore on the competitiveness of a company. The steel production planning process is known to be extremely difficult, with various constraints at the different production stages. As a consequence, not all constraints are respected in higher planning levels by current planning procedures. Detailed production planning at the different production stages has the task to derive production programs that are able to respect all local constraints and at the same time lead to appropriate target achievement. Current approaches developed for detailed continuous casting planning are not able to efficiently calculate alternative production schedules and therewith enable decision support for conflicting objectives.

Within this thesis, the detailed continuous casting problem is presented in detail. A new approach to decompose the problem is described. With this decomposition, the problem can be treated as a single machine scheduling problem and effective meta heuristics developed for similar problems can be exploited. Further, with the chosen decomposition it is possible to respect the consumption of hot metal within the scheduling of charges. This important practical constraint could not be respected within the continuous casting problem in the past. Besides the hot metal consumption, casting specific setup requirements are considered in the scheduling model. MILP models are presented for the different extensions of the basic scheduling model. An iterated local search procedure is presented and the effective is shown based on the comparison with a commercial solver.

The findings obtained from the scheduling research are transferred into a decision support system for the detailed continuous casting planning. Based on an industry case study, the application of the developed tool is presented on a real industry situation.

Contents

List of Abbreviations

CB cast batching

CEGP critical element guided perturbation

CHB charge batching

COC commercial order clarification

CAS caster selection

CS cast sequencing

DSS decision support system

EDD earliest due date

GA genetic algorithm

GRASP greedy random adaptive search procedure

GTA group technology assumption

MILP Mixed Integer Linear Program

NBR net benefit of relocation

PSK Panwalker, Smith and Koulamas

SD slab design

SPT shortest processing time

SWPT shortest weighting processing time

TOC technological order clarification

TTD total target deviation

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The steel industry is the most important supplier of raw material for many industries, e.g. automotive industry, aircraft, housing and beverage (Cowling et al. 2003); (Balakrishnan and Geunes 2003) and a central part in industrialized economies (Atighehchian et al. 2009). Besides its importance, the steel industry is among the most competitive industries (Appelqvist and Lehtonen 2005). The ten largest manufacturers together only hold 25% of the market share (IISI 2011). The annual world steel production is constantly increasing. In the year 2000 the annual production has been 849 million metric tons. With average annual growth rates of 6.1% (2000-2005) and 4.3% (2005-2010) it reached 1.414 million metric tons in 2010 (IISI 2011). The supply exceeds the demand, leading to over-capacities and increasing pressure on the world steel market. At the same time the costs for raw material (i.e. iron ore and reducing agents coke, cole, oil, and gas) are rising, adding additional pressure on the profit margins of the companies (Ernst&Young 2012). Due to the capital intensive character of the steel production process (high costs for production machinery), a high throughput rate is unavoidable to keep the unit costs low by maintaining a high machine utilization (Atighehchian et al. 2009).

Especially European manufacturers are affected by high costs and face increasing competition from Asian steel companies (Cowling and Rezig 2000). The market share of China has tripled from 15.1% in 2000 up to 44.3% in 2010 (IISI 2011). According to a study from the European Commission on the competitiveness of the European steel sector, one way to survive on the market is to focus on on high technology and operational excellence (Ecorys 2008). European producers tend towards high product variety and quality in combination with superior logistics performance (i.e. short delivery times and high delivery reliability) in order to differentiate from their competitors (Denton et al. 2003). The importance of due date reliability is further increasing due to the close integration of steel producers into the customers' supply chains. These changes in target focus, however, most likely incorporate rising production costs. The way to success for European steel manufacturers lies in a reasonable balancing of the conflicting logistics objectives, in a way that is suitable for current market demands. In order to be able to make decisions according to a strategic balance of targets, the impact of

different decision alternatives (i.e. generated production plans) must be quantified and understood.

1.2 Goals of this work

The process of steel production is organized as a hybrid flow shop with multiple stages. Each stage has highly differing requirements regarding the production sequence (Voß and Witt 2007). The resulting scheduling problem is too complex for all constraints to be respected within one planning procedure (Missbauer et al. 2009). Therefore, production planning in steel production is executed in hierarchical planning levels (Vanhoucke and Debels 2009). In a higher planning level, the customer orders are scheduled without considering all constraints of the different production stages. Based on the resulting schedule, delivery dates are communicated to the customers. Within lower planning levels, the actual production sequence is decided by taking all detailed requirements of the individual production stage into account. As a result, conflicting targets arise in lower level planning. The continuous casting production stage is usually the bottleneck work system in steel production and therefore can be considered as the most important stage for detailed planning (Tang and Wang 2008).

Due to the described complexity of detailed continuous casting planning, the generation of a feasible production program is a time consuming procedure. In practise, this leads to the situation that only one production program is generated and evaluated according to different planning objectives. In case the planner is not satisfied with the results, adjustments are conducted in an iterated way to derive few additional production programs. The goal of this PhD thesis is to develop a decision support system (DSS) that is able to generate multiple alternative schedules for a planning situation automatically. This way, the planner is able to compare more scheduling alternatives and potentially is able to achieve a more balanced target achievement.

The goals of this PhD thesis can be divided into three parts:

(1) a detailed understanding of the continuous casting planning problem will be achieved. This includes a structured analysis of all decisions, constraints and objectives involved in the problem. Based on this understanding and the existing results in the continuous casting planning literature, a decomposition of the problem is proposed, such that the model is efficiently solvable to generate multiple scheduling alternatives and at the same time respects more real industry constraints sufficiently, which makes it useful in practice.

(2), a mathematical model for the problem and a heuristic to solve the problem will be developed, in order to be able to efficiently generate production programs automatically.

Since the selected decomposition is based on a single machine job scheduling model, the second intention of this PhD thesis consists of extending existing scheduling models conducted from the literature, to include relevant continuous casting constraints. Mixed Integer Linear Program (MILP) models for each extension will be presented to be able to compare the schedules generated with the heuristic against optimal solutions derived with a state of the art multi-purpose solver. Appropriate test instances will be created, to be able to conduct experiments.

(3), the heuristic will be embedded into a DSS that is used by continuous casting planners in practice. Since the decomposed model only covers part of the problem decisions, procedures will be developed to solve decision problems not covered in the job scheduling model. Further, approaches to generate scheduling alternatives that are able to represent good schedules regarding conflicting targets will be designed. Finally, the application of the designed DSS and the heuristic for real world instances is presented.

1.3 Overview

This work is organized as follows: Chapter 2 provides a detailed overview of the continuous casting planning problem and explains the approach selected for this thesis. Chapter 3 introduces the state of the art for the continuous casting problem as well as the single machine job scheduling literature that constitute the foundation for the development of a scheduling method. These two chapters contribute to the first goal of this PhD thesis. Chapter 4 contains different model extensions of the single machine family scheduling model to map the continuous casting problem. Chapter 5 presents the iterated local search heuristic developed to solve the problem. The fourth and fifth chapter correspond to the second goal of the thesis. In Chapter 6, the application of the heuristic within a DSS for the continuous casting planning problem is presented. With Chapter 6, the third goal of this PhD thesis is fulfilled. Finally, Chapter 7 concludes the work and provides recommendations for future research.

The Continuous Casting Planning Problem

The description to the continuous casting planning problem in Chapter 2, intends to associate the problem with the overall steel production planning. In order to realize this, the production process of the different stages of steel production are briefly introduced. Further, the higher planning process is described. Afterwards, Chapter 2 provides a detailed classification of the continuous casting planning problem. All containing decision processes are explained and the corresponding constraints are discussed. Further, the

different planning objectives are presented. After a summary of the elements of continuous casting planning problem, the approach selected for this PhD thesis is presented.

Related Work

The related work in Chapter 3 is split into the two components continuous casting planning literature and job scheduling literature. For the continuous casting part the existing literature is clustered into the sub problems involved in the planning problem. For each literature source, the classification scheme developed in Chapter 2 is applied to describe the contribution of the reference. Afterwards, limitations of current research are presented. For the job scheduling part the existing literature is clustered according to the required extensions for modelling the selected problem decomposition as a job scheduling problem. Again, the limitations of existing approaches are presented at the end of the chapter.

Models for the Combined Charge Batching and Sequencing Problem

As described before, the approach of this thesis is to decompose the continuous casting problem in such a way, that single machine job scheduling can be exploited to solve it. Chapter 4 presents four different MILP model extensions. Starting from a basic family scheduling model, upper resource constraints, specific setup constraints and lower resource constraints are included into the model. For each model, a description, a formal notation and the MILP model is presented.

Solution Approach

Chapter 5 has three major contents. First, the iterated local search heuristic is described. The description consists of the general framework, the developed perturbation strategy, a problem specific tardiness evaluation for each model and the different local search operators chosen in this thesis. Second, the design of model-specific test instances is explained and the general experiment setup, chosen to evaluate the developed heuristic, is presented. Chapter 5 concludes with the presentation and discussion of the experiment results. Using the generated test instances, the performance of the developed heuristic is compared to solving the models from Chapter 4 with a state of the art multi purpose solver.

Decision Support System for the Detailed Continuous Casting Planning Problem

Chapter 6 begins with a description of the general structure of the DSS and the data required and used by the system. Afterwards, the selected approach to generate scheduling alternatives is presented. By generating alternative schedules, the DSS is able to respect the multi-objective character of the continuous casting planning problem. In the following section, the selected procedures to solve the preprocessing planning decisions, that are not contained in the job scheduling model, are presented. Afterwards, the adoptions carried out to use the developed heuristic within the DSS for real world instances is described. Besides a description of tardiness and target evaluation procedures for the real world instances, acceleration strategies for the local search operations are presented to further reduce calculation time. Finally, the application of the DSS is shown on the example of three potential analyses that have been conducted with the DSS in an industry case study.

Conclusion

Chapter 7 provides a summary of the thesis, a critical discussion of the scientific contribution of this work and recommendations for future research.

2 The Continuous Casting Planning Problem

This chapter provides a detailed introduction to the continuous casting planning problem. Steel production planning is executed in hierarchical planning processes. This is due to the large complexity arising from the characteristics of the production process in different stages. In order to understand the challenge of the continuous casting planning problem, the steel production process is briefly described in Section 2.1. Afterwards, the high level planning of the entire steel production is introduced in Section 2.2. In the third Section 2.3, the continuous casting planning problem is described in detail.

2.1 Steel Production Process

This section is describing the steel production process in detail, with a focus on aspects relevant for continuous casting planning. The process of steel production is organized as a hybrid flow shop with multiple stages. Each stage has highly differing requirements regarding the production sequence (Voß and Witt 2007). Figure 2.1 displays the major production stages for steel production, the red flash symbols indicate stages with dominant planning constraints.

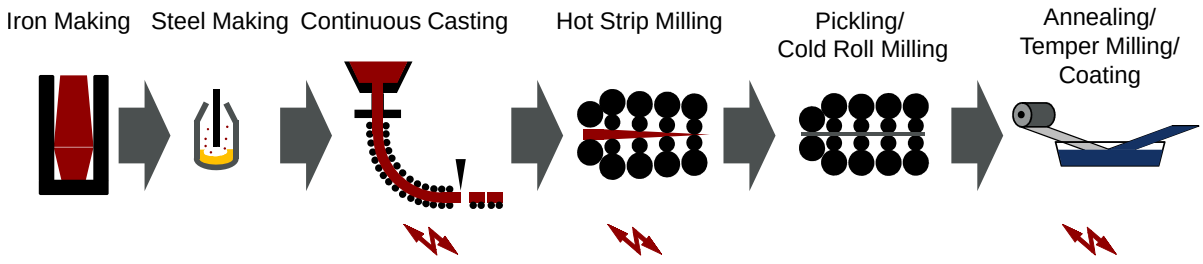

Figure 2.1: The major steps of the steel production process. The red flash symbols depict production stages with dominant constraints that complicate production planning.

Iron Making

In the first stage of steel production, raw material is processed into liquid hot metal. There are different technologies to execute this process, e.g. blast furnace, direct reduction or smelting reduction. In the following only blast furnace will be described, since it is the most common method. One major part of the raw material are ferrous materials (iron ore) that provide the iron. Iron ores are naturally existing rocks and minerals that are compounds of iron and oxygen mixed with impurities. Two types of ores are used in the blast furnace, fine and lump ore. In order to produce fine ore, it is ground and cleaned to increase the iron content and afterwards agglomerated by sintering and palletizing to be in the right shape. Within the blast furnace process, the oxygen is separated from the iron by binding it to a reducing agent, usually coke, coal oil or gas. The impurities of the ore are also split up using flux additions like limestone. The materials are fed into the top of the blast furnace in alternating layers. At the lower part of the furnace, a hot air blast is injected which raises the temperature to about 2200 °C. This high temperature leads to a complex reaction process of the incoming materials. The result is a pool of molten iron at the bottom part of the furnace. The flux additions in combination with the impurities form a liquid slag on top of the molten iron. Slag and molten iron are withdrawn from the furnace and afterwards separated. The molten iron, often called hot metal, is tapped into ladles or torpedo cars for further processing. The blast furnace is operated continuously for around 15-20 years (furnace cycle). Since the hot metal can only be stored for a very limited time in the torpedo cars, the need of hot metal consumption is an important steel production planning constraint (Ouelhadj 2003), (Degner et al. 2008).

Steel Making

In the second stage, the hot metal is processed into a certain steel grade (i.e. type of steel) that is requested by the customer. The goals of steel making are to reduce the carbon content to the specified value, to establish the requested alloy composition and to remove all unwanted elements left in the hot metal. The steel making process can be divided into oxygen steel making using a converter and secondary metallurgy where different ladle treatments are executed to further increase the steel quality. Similar to iron making, fluxes and alloying agents are added together with the hot metal and scrap into the converter. The fluxes again build up a slag layer to absorb impurities and to protect the molten steel from the atmosphere. The necessary chemical reactions are triggered by blowing pure oxygen into the metal bath. After all required specifications for a certain steel grade are reached, the liquid steel is tapped into ladles for further processing. In the field of secondary metallurgy, processes like homogenizing, heating, degassing and vacuum treatment are used to increase the cleanness and quality in order to meet today's high requirements on steel products (Ouelhadj 2003), (Degner et al. 2008).

Continuous Casting

Continuous casting is one of the major process steps in steel production in which liquid steel gets solidified into slabs of certain dimensions and weight for further processing. The input for the casting process is the liquid steel, transported in ladles of a specific size, referred to as charge or heat. The charge is poured into the first continuous casting device, the tundish (holds approx. 55 t). The tundish serves as a buffer and transfers the liquid steel through nozzles into one or multiple water-cooled copper moulds. The buffer function of the tundish allows for a consecutive production of multiple ladles. The flow rate can be regulated to realize a stable solidification process. At the beginning of a run, the mould is closed off by a so-called dummy bar. When the required level of filling is reached within the tundish, the mould starts to oscillate to avoid an adhesion of material. Within the moulds, the steel starts to solidify from the outside to the inside. When the surface of the strand is solid, the stand is transported through the mould using driving rolls. The geometry of the mould determines the dimensions of the steel strand. It is usually possible to adjust the mould to a certain degree, allowing varying slab dimensions within one caster run. Also different strand widths, in case of multiple strands, are possible. Because the strand shell is very thin, rough or extensive width changes could lead to a tearing of the shell. When a so-called breakout happens, liquid steel flows into the casting machine and thus results in loosing the charge. In addition, expensive cleaning and repair is necessary. The ability to change width during a charge depends on the steel grade and needs to be considered during caster planning. At the end of the casting process, the solid strand is cut into the required slab length using gas torches.

Due to high process temperatures, the refractory lining of the tundish and nozzle can only be in operation for a limited time (about 8 charges, depending on the steel grade and cast width). Afterwards, they need to be replaced and prepared in order to be reused. Since tundish and nozzle together form a piece of equipment, in the following only the term tundish will be used to describe both parts. To be produced within the same tundish, consecutive ladles need to consist of similar steel grades in terms of overlapping tolerances for temperatures and chemical properties. Therefore, metallurgists have developed so-called cast families, i.e. groups of steel grades that can be produced using the same tundish. In case consecutive ladles are from different cast families, a setup process is also required. There are basically three setup possibilities. One is called a caster turnaround, where the strands are entirely discharged. This process takes between 30 min and 90 min depending on the caster and the steel grade. The second alternative is called composite casting, where the strand is not discharged, but a plate is inserted between consecutive charges, preventing the steel grades to mix up. This procedure only takes between 5 min and 15 min of setup time. The third possibility is called composite casting without tundish change. It is similar to a regular charge transition, but with charges from different cast families. The caster is strongly slowed down so the setup lasts about 15 min. The resulting mixed-zone of the two different steel grades can either be used for an order with low requirements (only when the steel grades are similar enough) or has to be scrapped. Besides the steel grade, the width of the strands also

determines the need and applicable type of setup process. In case consecutive charges contain differing strand widths that exceed the ability of the caster to adjust the moulds, a caster turnaround setup is required, independent of the steel grades (Box and Herbe 1988), (Lee et al. 1996), (Degner et al. 2008).

Adjusting and Slab Yard

In the ideal case, the slabs produced at the continuous casting stage are immediately transported and processed at the hot strip mill. Whenever a steel manufacturer is producing a high variety of products, the inhomogeneous constraints of continuous casting and hot strip milling require an intermediate slab yard to decouple these very asynchronous processes. E.g. some slabs need to be further treated before they are ready for milling. The reasons for adjusting slabs vary. One is due to planning decisions such as combining two narrow slabs to one mother slab and splitting it afterwards. A further reason for slitting occurs when producing wider slabs then required, in order to complement charges. Another reason is unplanned rework in case of quality issues during casting. The adjusting processes include slitting, cut-to-length and scarfing. The adjusting processes involve manual work, therefore slabs need to be stored in the slab yard to cool down to appropriate temperatures (Tang et al. 2001).

Hot Strip Milling

At the hot strip mill stage, the slabs are processed into sheet metal and usually rolled up to steel coils. Rolling the steel at high temperature leads to recrystallization processes that enable a larger degree of deformation at lower forces. An unwanted hardening of the steel can be avoided. A very important constraint for detailed hot strip mill production planning is to schedule the slabs in a certain width pattern (coffin shape). This reduces the wear out of milling rolls which is required to meet quality specifications and reduce costs. Other constraints are e.g. steel grade and thickness (Degner et al. 2008).

Pickling and Cold Strip Milling

Cold strip milling is used to further improve the product quality in terms of smaller dimensions and dimensional accuracy, surface quality as well as strength properties. Before cold milling is possible, the mill scale (thin oxide layer) resulting from hot rolling needs to be removed. In a pickling processes, the sheet metal is dipped into a vat of acid that removes the scale. The two processes can either be carried out separately or interlinked in automated pickling-tandem mills (Degner et al. 2008).

Heat Treatment and Coating

In order to compete in the described challenging market situation, steel manufacturers need to further improve the properties of their final products. Heat treatment and

coating are finishing processes that lead to highly specialized products with superior properties. Heat treatment involves annealing, hardening and tempering. All three processes are used in combination to adjust the micro structure and adjust hardness, toughness and stress levels as required by the customer. Annealing is heating the material to a certain temperature, holding the temperature for a specific time period followed by a controlled cooling process. Hardening is heating above austenitizing temperature and rapid cooling. Tempering involves heating with low temperatures below the austenizing point. In order to avoid corrosion or to improve chemical resistance, different coating processes can be applied. This includes organic coating (e.g. plastics or rubber), inorganic coating (e.g. ceramic, cement motar) or metallic coating (e.g. zinc, cadmium or nickel). Metal coating can be carried out as hotdip or electrolytically galvanizing processes (Degner et al. 2008).

2.2 High Level Steel Planning

The different stages of the steel production process have highly differing requirements on the production sequence. The resulting scheduling problem is too complex for all constraints to be respected within one planning procedure (Voß and Witt 2007), (Missbauer et al. 2009). Therefore, production planning in steel production is executed in hierarchical planning levels (Vanhoucke and Debels 2009). In a higher planning level (rough production planning), the customer orders are scheduled into planning periods. Based on the resulting schedule, delivery dates are communicated to the customers. Within lower planning levels (detailed production planning), the actual production sequence is decided by taking all detailed requirements of the individual production stage into account. In the following, a typical rough production planning is introduced.

There are six planning processes belonging to rough planning as considered in this thesis. These are (1) demand planning, (2) net requirements calculations, (3) rough allocation planning, (4) detailed allocation planning, (5) delivery date calculation, (6) rough scheduling. Figure 2.2 provides on overview of these tasks.

Within (1) demand planning, the sales department is evaluating gross demands based on existing orders and historical data. This information is estimated for a period of approx. 18 months. This forecasting is not possible on a detailed steel product level, therefore certain clusters (planning items) were developed based on product categories (e.g. the input material (hot strip, cold strip) and the executed finishing processes (e.g. coating, annealing). For these clusters gross demands are defined. Within (2) net requirements calculations, the information on booked orders is used to estimate the net requirements on a weekly basis. (3) rough allocation planning is matching these net requirements with the existing capacities under consideration of already allocated work loads. This planning step is still done for a mid-term period of approx. 15 months. The result are

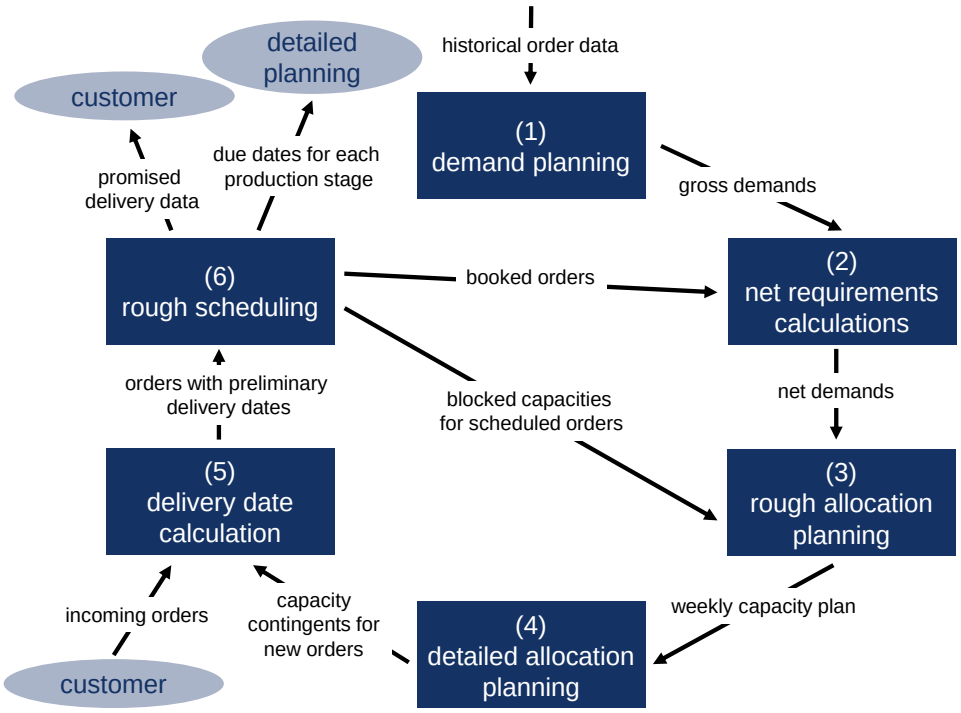

Figure 2.2: Overview of the typical tasks involved in rough planning in the steel industry.

weekly contingents for each planning item. In (4) detailed allocation planning, these contingents are broken down into more precise product clusters (planning products), with a higher level of detail regarding product type and process steps. In the next planning process, (5) delivery date calculation, actual incoming customer orders are processed. The products that can be ordered by the customer can be categorized by the type of steel, the specific treatments and the dimensions of the final product. Whenever a customer requests a certain product, the sales manager first checks whether there is still capacity for the specific contingent and all legal aspects are clarified within commercial order clarification (COC). Based on the capacity contingents of the planning product the customer requests, a preliminary delivery date is calculated as the sum of planned production lead times for each necessary production step. This preliminary date is communicated to the customer. In the next step, the order information is passed on to the technological order clarification (TOC). Based on the product specifications, a routing through production is determined and requirements in terms of intermediate products for each production stage are derived. In the last process of rough planning, the orders are scheduled for the determined production route. This scheduling is referred to as (6) rough scheduling, due to the incomplete consideration of all constraints for each production stage. E.g. for continuous casting, often only the overall capacity of the caster is considered. The necessity to group certain orders together is often not considered in rough planning. As stated above, due to the highly differing constraints on each production stage, no overall procedure is known in the literature that is able to respect all constraints at once. The result of rough scheduling, are due dates for each

order and every production stage. Based on this information, a promised delivery date is communicated to the customer.

For every order, the information on the required intermediate product (from TOC) and the due date are communicated to each necessary production stage. Under consideration of all constraints of the production stage, feasible short-term production schedules are derived while trying to meet the planned due dates. The planning process for the continuous casting stage is described in detail in the next section.

2.3 Continuous Casting Planning Problem

The task of detailed continuous casting production planning is to convert the incoming production orders into a production program for the upcoming planning period, for all casting facilities. The process is known to be highly challenging (Harjunkoski and Grossmann 2001). In contrary to rough planning results described in the previous section, this production program respects all constraints and is feasible to be produced by the casting facilities. Since modern casters are constructed to produce two strands from one tundish (see Section 2.1), the following descriptions are based on twin strand casters.

Tang et al. (2011) describe three decisions that form the continuous casting planning problem: (1) how the slabs created to fulfil a specific customer order should be dimensioned (referred to as *Slab Design* in the following), (2) which slabs should be produced together in one charge (referred to as *Charge Batching* in then following), and (3) how many and which charges should be cast together on one tundish (referred to as *Cast Batching* in the following). Complemented by a preliminary decision (4) on the allocation of orders to casters (referred to as *Caster Selection* in the following) and the concluding decision (5) on how to sequence the generate casts on each caster (referred to as *Cast Sequencing* in the following), the five sub problems involved in continuous casting planning are described. Figure 2.3 depicts the planning elements and decisions of detailed continuous casting production planning. A production program for one caster is described by a sequence of casts.

In the following Sections, each decision is described in detail with the corresponding constraints that limit the decision space. Starting with *caster selection (CAS)* in Section 2.3.1, followed by *slab design (SD)* in Section 2.3.2, *charge batching (CHB)* in Section 2.3.3, *cast batching (CB)* in Section 2.3.4 and finally, *cast sequencing (CS)* in Section 2.3.5. Afterwards, the different planning objectives involved in continuous casting planning are presented within an extra Section 2.3.6. This is motivated, because the objectives are often influenced by more then one decision.

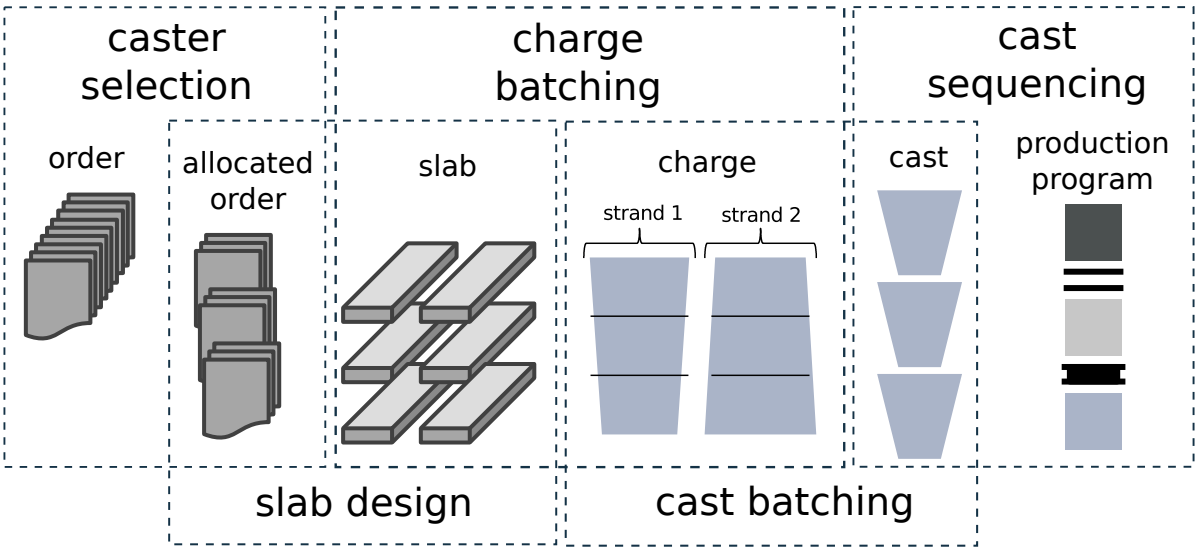

Figure 2.3: The elements and decisions of detailed continuous casting production planning

2.3.1 Caster Selection

In case more than one caster is available for production it has to be chosen which particular work system to use for a production order. There can be a large difference on the suitability of a caster for a certain production order. This is due to varying caster specifications. The most important difference is the ability to squeeze a slab while rolling it. A caster with a large squeezing range of 300 mm could e.g. produce a 1500 mm wide hot strip out of slabs with width between 1500 mm - 1800 mm. The ability to produce certain steel grades and steel qualities can vary significantly between different casters, e.g. influenced by casting speeds and/or the guiding technology of the caster. Finally, load levelling between different casters and providing the right orders to execute the subsequent planning decisions (e.g. to group slabs of equal steel grade together in *Charge Batching*) are important aspects of *Caster Selection*.

Caster Selection is only constrained in case a caster cannot be used to produce a specific production order due to technological requirements or higher level planning decisions that are out of scope for continuous casting planning. The constraints are set within TOC and provided to continuous casting planning (***CAS1:applicable caster***).

2.3.2 Slab Design

As described previously, customers are ordering final products which are most often not slabs but coils. The purpose of *Slab Design* is to create a slab by determining

the dimensions (width, length, thickness, resulting weight) and the steel grade that is needed to produce the final product (Tang et al. 2011). They are specified during TOC. The dimensions of a slab are determined by the mould used in the caster. Since the thickness of the mould can usually not be adjusted, the only way to vary the thickness is to exchange the mould. With investment costs of approx. 1 000 000 EUR for a single mould, the thickness can be considered as a fixed parameter (e.g. 257 mm). The other parameters that determine the dimensions of a slab, length, width and resulting weight can be adjusted and therefore need to be determined for each slab. The requirements for the dimensions of a slab depend on order specification and routing informations (e.g. the described ability of the allocated hot mill to squeeze slabs). Target values, as well as tolerance ranges are provided for length (***SD1:min/max slab length***), width (***SD2:min/max slab width***) and weight (***SD3:min/max slab weight***) for each slab. Further, a slab quality (described by the surface quality, the purity, and the metal micro structure of the slab) as well as a steel quality (described by a required steel grade) are given for each production order.

Within detailed continuous casting production planning the actual dimensions of the slab to be produced are decided. Because of the given flexibilities there are many possible outcomes of *Slab Design* as depicted on an example in Figure 2.4. Only the area determined by length and width is displayed, since the thickness is usually constant (as described above).

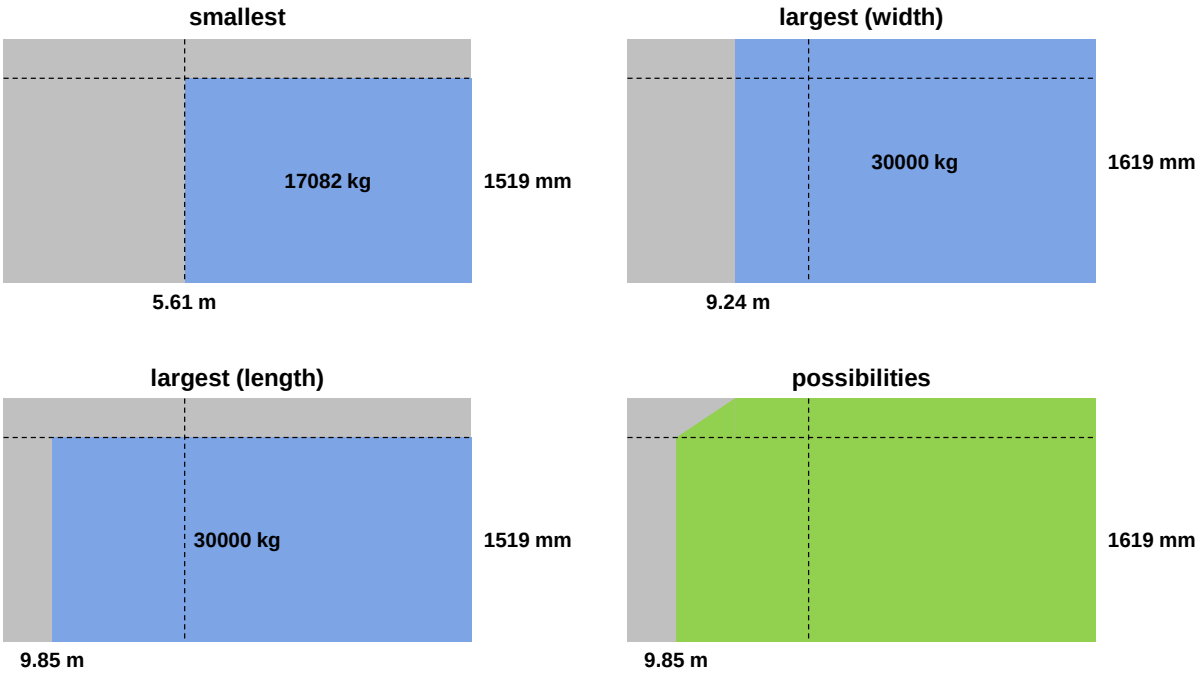

Figure 2.4: An example of the decision space of *Slab Design*.

Considering a sample slab with the following exemplary information provided by TOC:

- thickness (determined by selected caster): 257 mm
- length: [5.61 m - 11.5 m]
- width: [1519 mm - 1619 mm]
- weight: [15 000 kg - 30 000 kg]
- density: 7800 kg/m^3

In this example the smallest possible slab (depicted on the up left corner of Figure 2.4) would be at the minimum length 5.61 m and width 1519 mm with a resulting weight of 17 082 kg. A slab with maximum length 11.5 m and width 1619 mm has a resulting weight of 37 323 kg (not displayed in Figure 2.4) and exceeds the slab weight maximum. Therefore, biggest slabs either are thin 1519 mm with maximal possible length 9.85 m (lower left corner corner of Figure 2.4) or wide 1619 mm with length of 9.24 m (lower right corner corner of Figure 2.4) while not exceeding the maximum weight of 30 000 kg. There are a large number of possible slab dimensions in between those extremes. The actual choice of slab dimension is influenced by multiple objectives and interacts with other decisions of detailed continuous casting production planning (see Section 2.3.6 for more details).

Another constraint for *Slab Design* are prohibited slab length clusters as depicted in Figure 2.5 (***SD4:slab length cluster***).

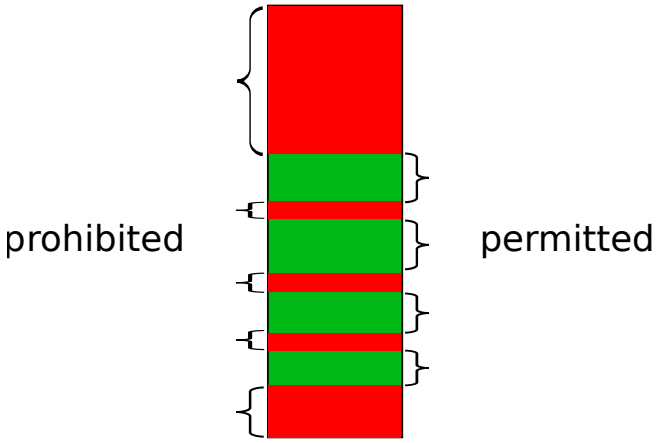

Figure 2.5: Example of different length cluster that are prohibited within *Slab Design*

The slab length cluster constraint is determined by the selected routing for a production order, since its characteristic is depending on the hot strip mill used in further processing of the slab. Before hot strip milling can be carried out, slabs usually need to be reheated when a direct rolling is not executed. Therefore, slabs are stacked on orthogonal beams

in the furnace of the hot strip mill. In order to avoid an unwanted bending of the slabs, certain slab length cannot be used.

Another aspect of *Slab Design* occurs whenever orders require slabs with a small width that is only a fraction of the maximum casting width ability. In that case, it is possible to produce a wide slab (mother slab) and split it after casting into smaller sub slabs (daughter slabs). This could be useful in order to complement very wide orders with very narrow ones in *Charge Batching*. It is also favourable in terms of production output, since this leads to wide slabs and therefore more quantity produced within the same period of time. Whenever mother slabs are composed of slabs from different production orders, difficulties could arise when order parameters (e.g. slab quality, due date or slab length) differ from each other. The ability to combine slabs into a single mother slab is therefore constrained by steel quality compatibility (***CHB1:steel quality compatibility***), slab quality compatibility (***CB4:slab quality compatibility***) and maximum casting width on a strand (***SD5:maximum strand width***). The aspects of steel quality and slab quality compatibility are discussed in more detail in the following sections. The objectives influenced by the generation of mother slabs are discussed in more detail in Section 2.3.6.

2.3.3 Charge Batching

One ladle holding a certain amount of steel (e.g. 250 t) is the elementary production unit of the continuous casting stage as described in Section 2.1. In continuous casting planning, one ladle is referred to as one charge. Since the actual amount of steel that is contained within a ladle can vary depending on the steel mill and on the wear status of the refractory lining, a certain flexibility in the charge size generated in *Charge Batching* is provided. The task of *Charge Batching* is to determine the slabs generated in *Slab Design*, that are grouped together within one charge. Further, the allocation of slabs to the two strands (for twin strand casters as considered in this thesis) and the order in which the slabs are cast in the strands is selected during *Charge Batching*. Figure 2.6 depicts one charge, consisting of four slabs in positions $k = 0, ..., 3$ for each of the two strands $st = 0$ and $st = 1$.

The most important constraint for the *Charge Batching* decision is the steel grade. A single charge is characterized by a specific steel grade realized within the steel making process (see Section 2.1). Only slabs with compatible steel grades can be allocated to that charge (***CHB1:steel quality compatibility***). Since a steel grade is characterized by certain physical and chemical properties, it is possible to meet a given order with a steel grade that outperforms the expectations of the customer (i.e. a higher steel grade). By choosing a high steel grade for a charge, the number of possible slabs available for scheduling increases. This so called quality upgrading can be used in order to complement charges.

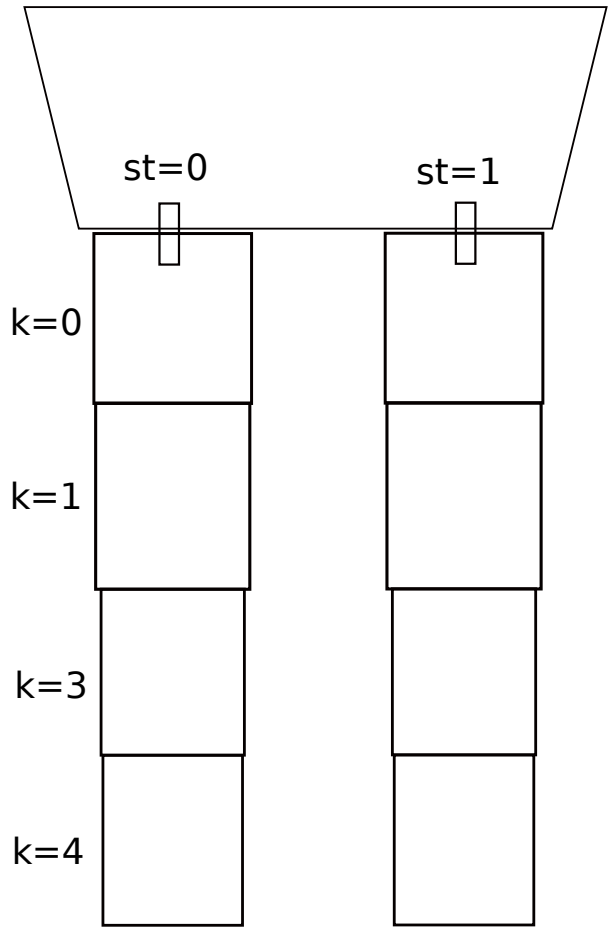

Figure 2.6: Example on how eight slabs are allocated to the two strands of a charge.

The need to complement charges arises from the constraint that only full charges are cast. Whenever not enough slabs are available for a given steel grade (even with quality upgrading), slabs are produced without having a production order allocated to it. These so-called open ordered slabs need to be stored in a slab yard to be reallocated to future production orders, in case of matching specifications. While it is not possible to produce less than a full charge size, it is also not possible that the sum of all slab weights allocated to one charge exceeds the charge size (***CHB2:charge size***).

Another aspect of *Charge Batching* is the width transition between two consecutive slabs on a strand. With modern casters, it is possible to adjust the width of the mould during the casting process. In order to avoid a breakthrough of liquid steel, width changes can only be executed gradually and the maximum possible width change is limited (***CHB3:slab width transition***). Narrowing the width is less dangerous than widening, therefore the direction of width change has to be distinguished. The maximum width difference between two consecutive slabs is influenced by the steel grade, the caster, the slab position within the cast and the direction of width change. The steel grade defines whether a width change in a certain direction is possible or not. The position within the

cast and the caster determine the amount of deviation allowed. Further, for certain steel grades (depending on the carbon content) width changes are not possible. Whenever the width between consecutive slabs is changed, a trapezoidal transition area results, as depicted in Figure 2.7. In case the area is sufficiently small, the hot strip mill is able to compensate for it. Otherwise, an additional production step (slitting) is necessary to straighten the slab before milling (see Section 2.1). In some cases, large width changes are favourable in order to connect customer orders with differing width requirements to meet due date requirements.

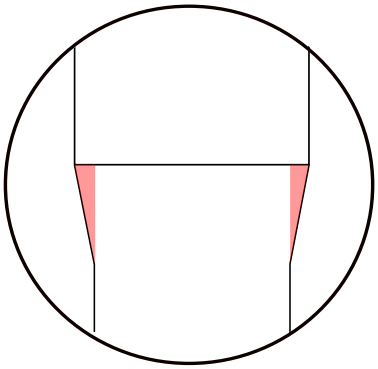

Figure 2.7: Example of the resulting trapezoidal area (depicted in red) when adjusting the slab width during the casting process.

While the maximum width that can be produced on each strand is already considered within *Slab Design*, it is necessary during *Charge Batching* to assure that the sum of the two strand widths does not exceed the total width limit of the caster (***CHB4:maximum total strand width***).

The objectives influenced by the *Charge Batching* decision are described in Section 2.3.6.

2.3.4 Cast Batching

Within *Cast Batching* the number of charges grouped into one cast and the sequence of charges in the cast are determined. A cast is defined as the charges produced using one tundish, i.e. between two setup processes.

When producing consecutive charges using the same tundish, a mixed zone of steel occurs when pouring the subsequent ladle into the tundish. No quality issues occur in case the steel grades of the two tundishes are equal. Similar to the described steel grade compatibility in *Charge Batching*, charges with similar steel grades can also be cast consecutively on the same tundish whenever chemical properties overlap. Metallurgists have defined so-called cast families, that describe the groups of steel grades that fulfil these requirements. Charges that do not belong to the same cast family are not allowed to be grouped together into the same cast (***CB1:cast family compatibility***). Since steel grade and cast families are defined by ranges of technological properties, it is possible that a certain steel grade fulfils the requirements of multiple cast families. In this case, a charge with an overlapping steel grade could be allocated to any cast with respective cast family within *Cast Batching*.

The number of charges that can be produced consecutively within one cast is constrained due to the limited lifespan of the tundish. As described in Section 2.1, the refractory lining of the tundish gets worn out during production and needs to be replaced when a certain wear status is reached. In the literature, the limit is usually described as a time span during which a tundish can be utilized. The casting time is basically determined by the casting width and the casting speed, which is depending on the steel grade. In practice, the above described informations are usually summarized into a maximum number of consecutive charges allowed for each cast family (***CB2:maximum batch size***).

Similar to the width transition requirements described in *Charge Batching*, the connecting strands of two consecutive charges need to be within a maximum difference. Again, trapezoidal slabs occur in case the width needs to be adjusted (***CHB3:slab width transition***).

Another aspect, that is determined within *Cast Batching*, is the slab quality. The slab quality is divided in the three categories purity, internal structure and surface quality. Each category is evaluated on a scale reaching from 1 to 9. Therefore, the slab quality can range from (1/1/1) lowest quality up to (9/9/9) highest quality. The output quality of a produced slab is depending on the position of the slab within the cast, the caster and the affiliation to a certain casting cluster (e.g. interstitial free steel). The flow properties of the liquid steel are the major influencing factor on the achievable slab quality. In case turbulences disturb a smooth flow of liquid steel, the slab quality is decreased. This happens prior at the beginning of a cast, when the steel is poured into an empty tundish. However, when a tundish is running low, or a new charge is poured into the tundish within a running cast, turbulences in the steel flow occur. Metallurgists have defined slab position categories within a cast, and associated achievable slab qualities to each category. Different slab position categories are depicted on an example cast consisting of three charges in Figure 2.8.

There are two possible beginnings of a cast depending on the setup type used. The first

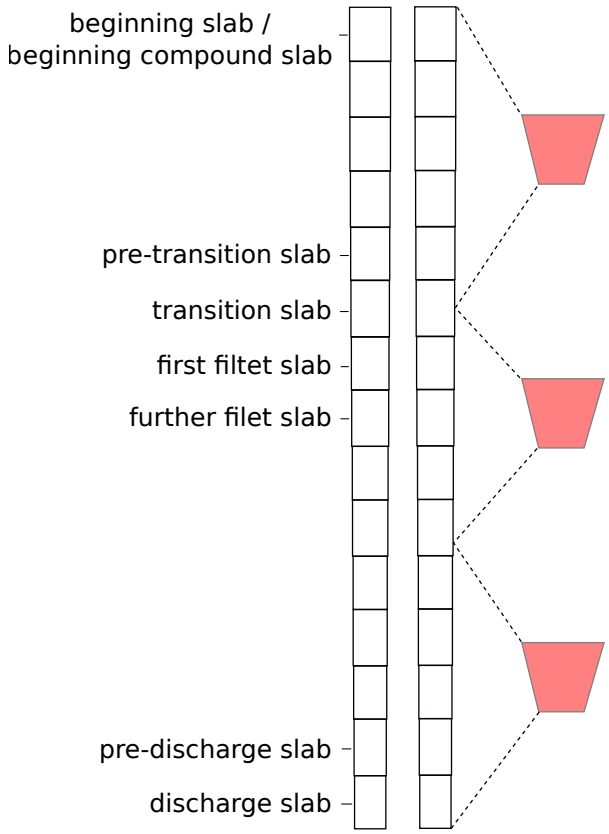

Figure 2.8: Example of slab positions in a cast with three charges.

slab position is either called beginning slab, in case of caster turnaround, or beginning compound slab after a compound setup was carried out. Due to the described turbulences attached to filling in the first ladle, the quality of these first slabs is rather low. Subsequent slabs are of high quality and are named first fillet slab and further fillet slab. At the transition of two charges, it is distinguished between the pre-transition slab before the new ladle is used and the transition slab during or right after the ladle change. Finally, at the end of each cast when the strands are discharged, again turbulences arise leading to reduced slab quality. This affects the pre-discharge slab and the discharge slab at the very end of the cast. For a slab to be allowed at a certain slab position, the customer requested slab quality must be less or equal to the slab quality possible at that position (***CB4:slab quality compatibility***). E.g. if a slab position only generates (3/3/3) quality, slabs from orders where the customer requires higher quality at any category cannot be used in this position. In case no customer order for that slab position exists, open ordered slabs have to be produced. In contrast, higher slab quality than expected by the customer is allowed but should be avoided (see Section 2.3.6 for further discussion of slab quality upgrading).

Finally, the two strands need to be balanced in terms of total strand length, since both strands are fed from the same tundish. This constraint is highly dependent on the *Slab*

Design decision, because the total length on each strand is defined by the sum of slab length of the containing charges. A small tolerance is given as a maximum allowed length difference (***CB5:strand length difference***).

2.3.5 Cast Sequencing

Within *Cast Sequencing* the sequence of casts and therewith the final production program is generated for each caster. This is usually done on a rolling daily basis.

The basic restriction of *Cast Sequencing* is the capacity of the caster. Usually casters are operated continually 24 h per day. However, the available capacity could be reduced by disruptions or scheduled maintenance (***CS1:caster capacity***). The casting time is determined by the caster, the steel grade and the width of the strand. Available production time and casting time per charge determine the total output of the caster.

Another aspect determined within the *Cast Sequencing* is the possible setup type between consecutive casts. As described previously, there are the two main setup options to change from one cast to another, caster turnaround and composite setup. At the caster turnaround, the caster is stopped and the strand is completely discharged. This takes between 30 min and 90 min depending on the caster and the steel grade. During that time no output is produced. In a composite setup, the caster is only slowed down and a plate is inserted between the consecutive charges, protecting the steel grades to mix up. This procedure only takes between 5 min and 15 min of setup time depending on the caster and steel grade. The plate used for a composite setup costs approximately additional 1000 EUR. The caster turnaround is always possible, while the ability to carry out a composite setup is restricted. First, the steel grade of the preceding cast needs to be bondable with a following cast. Second, the succeeding cast also needs to be bondable with a previous cast. These two cases can be discerned for a steel grade. Third, the width requirements described for *Charge Batching* and *Cast Batching* need to be fulfilled for the connection on both strands (***CS2:setup type***).

The total number of setup processes, which contain a tundish change, is limited. Every time a tundish is exchanged, the refractory lining has to be reworked in a special shop. The amount of possible tundish changes that can be carried out in one day is therefore determined by the number of tundishes available for a caster and the time required to rework a used tundish (***CS3:maximum setups***).

A very important constraint for *Cast Sequencing* is to fulfil the hot metal consumption constraint. As described in Section 2.1, hot metal is produced constantly in the blast furnace. Each charge is consuming a certain amount of hot metal depending on the steel grade that is processed out of the hot metal and the ware state of the ladle. The size in terms of number of charges in a cast and the resulting setup process from the

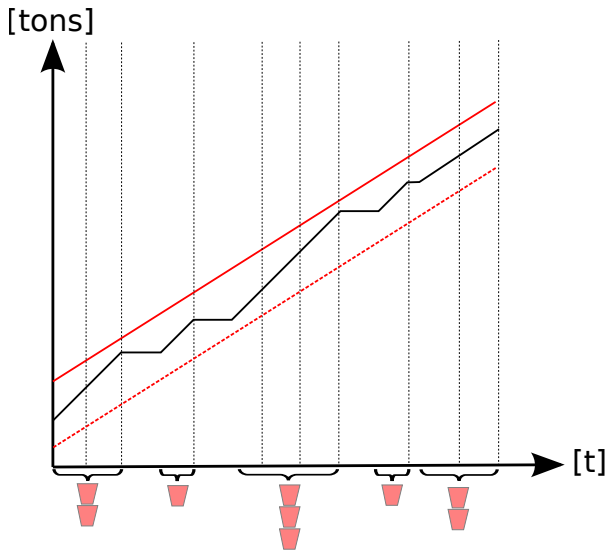

Figure 2.9: Dependencies of *Cast Sequencing* and hot metal consumption for an example production program.

sequence of casts determines the hot metal consumption rate of the caster. At any time, the generated sequence can only consume at most the amount of hot metal that is produced. Further, since the ability to store hot metal is limited (see Section 2.1), the generated production program is forced to consume at least a certain amount of hot metal at any time (***CS4:hot metal consumption***). The number of available torpedo cars is determining the buffer capacity. Figure 2.9 presents the influence of *Cast Sequencing* and hot metal consumption requirements. Below the graph, a sequence of five casts with a total number of (2,1,3,1,2) charges is depicted. The black line represents the hot metal consumption. During a setup process no hot metal is consumed, leading to plateaus in the hot metal consumption. The solid red line shows the constant hot metal supply and the dashed red line, the buffer limit for storing hot metal. The above described possibility to execute a short composite setup process is depicted for the last two casts in the given sequence.

While the steel making production stage is quite flexible in producing different steel grades, one major bottleneck exists. Certain steel qualities require to undergo vacuum treatment to be realised. This is carried out in the vacuum facility of the steel plant. For simplification purposes, the capacity of the vacuum facility is usually given on a daily basis. A generated production program needs to assure that the sum of material that requires vacuum treatment scheduled within one day does not exceed the given capacity. Similar to upstream vacuum treatment capacities, downstream hot transportation capacities can place a constraint for *Cast Sequencing* (***CS5:vacuum treatment capacity***). Certain material needs to be further processed at the hot strip mill directly from casting heat. In order to achieve this, a transportation in special vehicles is required. Since only a certain amount of those vehicles is available, *Cast Sequencing* needs

to assure that the amount of material that requires a hot transport within a planning period does not exceed the given capacity(***CS6:hot transportation capacity***).

Another aspect that constraints the *Cast Sequencing* decision are special casts. Special casts arise from different reasons (e.g. day shift requirement, material required for cleaning in downstream production processes, casts for ramp up processes), but lead to the same planning constraint. They need to be scheduled at a certain point of time and therewith limit the freedom of *Cast Sequencing* (***CS7:special casts***). While this constraint is not justified by technological requirements, higher planning decision can still lead to hard constraints for detailed continuous casting planning.

2.3.6 Planning Objectives

In the following Section the different objectives involved in continuous casting planning are described. Since objectives are often influenced by more than one planning decision, they are separated from the description of decisions and constraints.

There are two basic classes of objectives. The first class is influenced by the set of slabs that have been scheduled within a certain planning period. It is determined by a combination of all planning decisions. The second class of objectives is influenced by the way the slabs are arranged within the charges, casts and production program. Again this is influenced by all planning decisions since the ability to generate a certain arrangement is determined within all decision levels.

Due Date Fulfilment

In order to meet the delivery date communicated to the customer, due dates for each production stage are determined within high level planning (see Section 2.2). The ability to meet these due dates at the continuous casting stage is an important target of detailed continuous casting production planning. The deviation between the actual production date and the due date is referred to as lateness. Positive values of lateness indicate late production (tardiness), whereas negative values indicate early production (earliness). In case of late production, less time is available to process a slab on downstream production stages and the possibility of late deliveries to the customer increases. Producing slabs earlier than planned, also leads to negative effects on the logistics performance. First, the slabs need to be stored and increase the inventory costs. Second, casting capacity used on early production is lost for orders that were originally planned within the planning period. Resulting in late production of the postponed orders.

The target is to minimize the deviation of actual production date and scheduled due

date for each slab produced on the different caster (***O1: duedate fulfilment***). This target is primary effected by the set of slabs scheduled for the planning period under investigation.

Utilization of Upstream and Downstream Facilities

The downstream adjusting facilities, i.e. splitting, scarfing and cut-to-length, are mostly manual activities. Therefore, the capacity is restricted to the amount of workers available in a planning period. Since it is very expensive to have excess capacities, labour requirements are calculated based on average needs. This leads to a certain capacity available in each planning period. The total quantity of material that requires a certain adjusting process, planned for a planning period, should not exceed this capacity to avoid that material stays in inventory for longer than planned. On the other side, not utilizing the given capacities results in opportunity costs.

The capacities for vacuum treatment and hot transportation constrain the *Cast Sequencing* decision as pointed out in Section 2.3.5. Short-term capacity adjustments are very restricted. Therefore, capacity not utilized during one planning period is lost. This could lead to insufficient capacities in other periods when a large amount of orders is in need of the work system. The target of detailed continuous casting production planning is to equally utilize the constrained work systems. The share of orders that require a certain facility within one planning period, should be equal to the average total share of orders for that facility.

To avoid opportunity costs, the target is to maximize the utilization of restricted upstream and downstream capacities (***O2: up/downstream capacity utilization***). This objective belongs to class one and is influenced by the set of slabs contained in the planning period under investigation.

Meet Downstream Demands

Within higher-level planning, certain output rates for each hot strip mill are planned in order to establish an appropriate overall utilization of downstream work systems. These planning decisions include inter-plant transfers of material from steel mills to hot strip mills of different locations. Since slabs stored within the slab yards could also be used to meet hot strip mill demands, a planning department is determining the net requirements and setting targets and tolerance windows for detailed continuous casting production planning. A similar situation occurs for other downstream production facilities, e.g. coating. Higher planning levels determine desired amounts of material for downstream work systems. The product type specified for each order determines the work plan and

therefore the traversed facilities for each slab. This information is used to define targets for material produced at the casting stage in a specific planning period.

The target is to minimize the deviation of produced material and demanded material for specified downstream work systems (***O3: downstream demand fulfilment***). This objective belongs to class one and is determined by the set of set of slabs contained in the planning period under investigation.

Utilization of Slab Quality

The slab position within a cast determines the slab quality, as described in Section 2.3.4. Each cast always consists of certain slab positions. One task of continuous casting production planning is to match the available slabs with those slab positions within a cast. Only when the customer quality requirements behind a set of slabs exactly fit to the slab quality provided by the positions of a cast, no deviation occurs. While it is not possible to meet a customer order with a lower quality slab, a fillet slab with highest quality could be allocated to any order. Meeting an order with slabs of higher quality than requested results in opportunity costs, since the slab produced at this position could have been sold to a customer with higher requirements for a higher price.

The target is to minimize the amount of customer orders that are served with slabs of a higher slab quality then required and thus minimize (***O4: slab quality upgrading***). This objective belongs to class two and is determined by the arrangement of slabs within the casts of a production program.

Utilization of Steel Quality

As described in Section 2.3.3, all slabs within a charge are produced in the same steel grade and it is possible to serve a customer order with a superior steel grade. Whenever a charge cannot be completely filled with slabs that require the same steel grade, one opportunity would be not to produce the charge and wait for matching orders to arrive in the future. However, if orders with compatible but inferior steel grade requirements exists, these orders could be used to complement the charge. Orders that are served with slabs of superior steel grade then requested, result in a quality upgrade that comprises an opportunity cost.

The target is to minimize the amount of customer orders that are served with slabs of a higher steel quality then required and thus minimize (***O5: steel quality upgrading***). This objective belongs to class two and is determined by the assignment of slabs to charges with a certain steel grade.

Open Ordered Slabs

Open ordered slabs are used to complement charges or casts when slabs derived from existing orders cannot be used to fulfil the constraints of *Charge Batching* and/or *Cast Batching*. This can occur whenever width transitions cannot be achieved, not enough slabs of a certain steel grade or cast family exist, not enough slabs for the different slab qualities derived from the cast positions exist, or to level out strand widths in *Cast Batching*. They are slabs produced with no associated customer order and therefore enable the planner to set the dimensions and steel/slab quality specification of the slab as needed to fulfil all constraints. After production the open ordered slabs are stored in the slab yard and wait to be allocated to incoming customer orders. During that time, the slabs create inventory holding costs and there is the risk of never finding an appropriate order. In case an incoming order is eligible to use the slab, often additional adjusting processes are needed to fit the slab dimensions.

The target is to minimize the amount of open ordered slabs used to complement the production program (***O6:open ordered slabs***). This objective belongs to class two and is determined by the interaction of *Slab Design*, *Charge Batching* and *Cast Batching*.

Utilization of Hot Strip Mill Furnaces

The decision of *Slab Design* involves determining the length of each slab. As described in Section 2.3.2, there is a certain flexibility to increase the slab length by reducing the width. Long slabs, close to the maximum furnace length, are desired because they increase the utilization of the hot strip mill furnaces. This is directly influencing the maximum output rate of the hot strip mill production stage.

The target is to maximize the hot strip mill furnace utilization (***O7: HSM furnace utilization***). This objective belongs to class two and is determined within *Slab Design*.

Utilization of Casters

The continuous casting stage is usually the bottleneck in steel production. Therefore, the total output generated is closely related to the casting output. In order to maximize the total production output, the utilization of the casters in terms of effective production time should be as high as possible. There are basically two factors that influence the utilization of the casters. First, the amount of time spend on setup processes and second, the slab width. As described in section 2.1, when changing between different cast families, the width of consecutive strands differ dramatically or whenever the lifespan of

the tundish is reached, a setup process is needed to exchange the tundish. Depending on the setup type, this procedure takes between 5 min and 90 min. Opportunity costs result whenever the maximum number of charges is not exploited.

Another possible way to maximize the caster utilization is to generate maximum slab width during *Slab Design*. The wider the strands, the less time is needed for producing the same amount of material, and the more material can be produced within one planning period.

The target is to maximize the caster utilization (***O8: caster utilization***). This objective primary belongs to class two and is determined by *Slab Design* and *Cast Sequencing*.

Tundish Utilization

Besides the opportunity costs for not utilizing the maximum possible output for the caster, opportunity costs arise whenever the maximum number of charges grouped into a cast is not utilized. This is because each tundish exchange results in labour costs for the setup process and the reworking of the refractory lining of the tundish.

The target is to maximize the tundish utilization (***O9: tundish utilization***). This objective primary belongs to class two and is determined by *Cast Batching*.

Adjusting

Certain adjusting processes like scarfing and slitting can be avoided during detailed continuous casting production planning. The need for scarfing depends on *Caster Selection*. Slitting can be prevented when the width between consecutive slabs is not changed to a higher degree than the hot strip mill is able to squeeze nonparallel slabs. This can be respected at *Charge Batching* as well as *Cast Batching*. Otherwise slitting is needed to prepare the trapezoidal slabs for milling. Another decision is to produce either very wide mother slabs that consist of two slabs but need to be split before milling, or to produce thin slabs without adjusting.

The target is to minimize the required amount of adjusting (***O10: adjusting***). This objective belongs to class two and is determined by *Caster Selection*, *Slab Design*, *Charge Batching* and *Cast Batching*.

Constraint	Planning Decision
CAS1: applicable caster	Caster Selection
SD1: min/max slab length	Slab Design
SD2: min/max slab width	Slab Design
SD3: min/max slab weight	Slab Design
SD4: slab length cluster	Slab Design
SD5: maximum strand width	Slab Design
CHB1: steel quality compatibility	Slab Design, Charge Batching
CHB2: charge size	Charge Batching
CHB3: slab width transition	Charge Batching, Cast Batching
CHB4: maximum total strand width	Charge Batching
CB1: cast family compatibility	Cast Batching
CB2: maximum batch size	Cast Batching
CB3: slab quality compatibility	Slab Design, Cast Batching
CB4: strand length difference	Cast Batching
CS1: caster capacity	Cast Sequencing
CS2: setup type	Cast Sequencing
CS3: maximum setups	Cast Sequencing
CS4: hot metal consumption	Cast Batching, Cast Sequencing
CS5: vacuum treatment capacity	Cast Sequencing
CS6: hot transport capacity	Cast Sequencing
CS7: special casts	Cast Sequencing

Table 2.1: Overview of all constraints involved in Continuous Casting Planning

2.4 Summary of the Continuous Casting Scheduling Problem

The continuous casting production process is highly challenging from a technological perspective. This complexity translates into a large number of constraints that need to be respected during detailed continuous casting production planning. In the previous sections all constraints have been discussed within the sub section of the respective planning decision. Table 2.1 provides an overview of all constraints with a link to the corresponding planning decision. Further, Figure 2.10 illustrates the different constraints graphically.

The different objectives described in Section 2.3.6 are summarized in Table 2.2. For each objective, the class (as defined in Section 2.3.6) and the major influencing decision are listed.

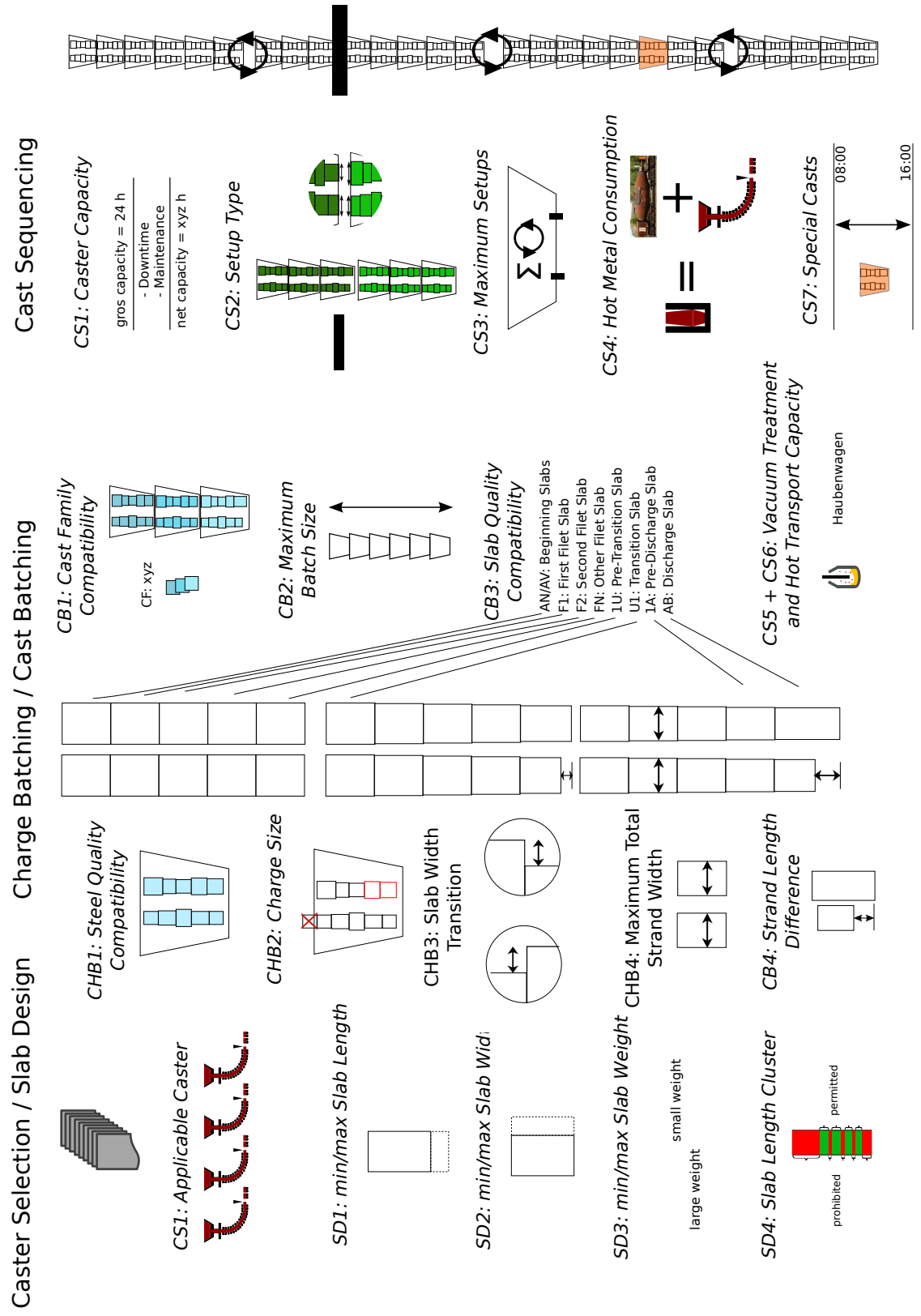

Figure 2.10: Overview of all constraints involved in Continuous Casting Planning

Objective	Class	Major Influencing Decisions
O1: duedate fulfilment	1	all
O2: up/downstream capacity utilization	1	all
O3: downstream demand fulfilment	1	all
O4: slab quality upgrading	2	Caster Selection, Charge Batching, Cast Batching
O5: steel quality upgrading	2	Caster Selection, Charge Batching
O6: open ordered slabs	2	Caster Selection, Slab Design, Charge Batching, Cast Batching
O7: HSM furnace utilization	2	Slab Design
O8: caster utilization	2	Caster Selection, Slab Design, Cast Sequencing
O9: tundish utilization	2	Caster Selection, Cast Batching
O10: adjusting	2	Caster Selection, Slab Design, Charge Batching, Cast Batching

Table 2.2: Overview of all Objectives involved in Continuous Casting Planning

2.5 Research Approach

The target of this thesis is to develop a DSS for the continuous casting planning problem, that enables a decision maker to balance the conflicting targets. Since the problem is too complex to consider all decisions, constraints and objectives within one model, the problem needs to be decomposed. Several decomposition approaches have been published in the continuous casting literature. They differ in the number of decisions contained, the constraints selected and objectives considered (see Section 2.3).

The largest issue with current approaches is the insufficient consideration of the ***CS4: Hot Metal Consumption*** constraint (see Section 3.2 for a detailed discussion). Since the (***CS4: hot metal consumption***) constraint primary influences the decisions *Cast Batching* and *Cast Sequencing*, the approach of this thesis is to decompose the continuous casting problem into two parts. The first part contains the decisions *Caster Selection*, *Slab Design* and *Charge Batching*. The second part contains *Cast Batching* and *Cast Sequencing*. Part 1 can be referred to as the Charge Pool Generation Problem and part 2 as Cast Batching and Sequencing Problem.

The focus of this thesis is set on the Cast Batching and Sequencing Problem. A formal model for this problem is developed in Section 4 and an iterated local search heuristic to solve the problem is presented. In Section 6 a DSS is described to support planners to generate good schedules for the continuous casting planning problem in industry applications. The DSS contains preprocessing procedures to generate solutions to the Charge Pool Generation Problem and utilizes the method generated in Section 4 to generate schedules for the Cast Batching and Sequencing Problem. An approach is developed to

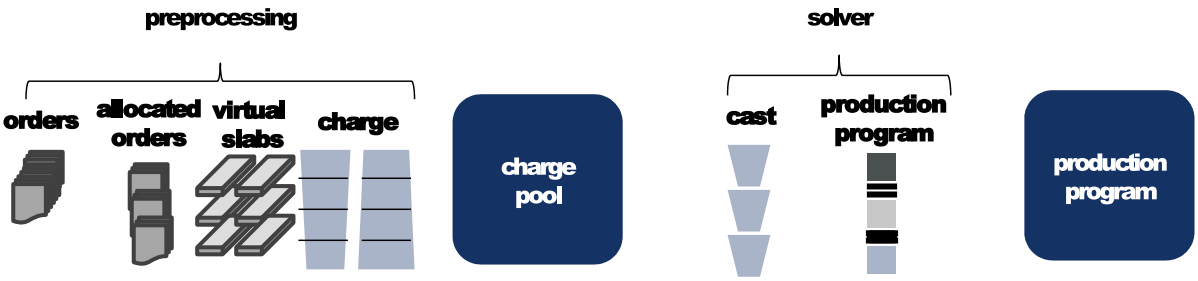

Figure 2.11: Decomposition approach to separate *Caster Selection*, *Slab Design* and *Charge Batching* from *Cast Batching* and *Cast Sequencing* as selected for this thesis

respect the multi-objective character of the problem and provide the decision maker with alternative production programs. This enables him to find a good compromise between conflicting objectives and to derive a balanced target achievement. Figure 2.11 depicts a simplified representation of the approach.

The approach selected to solve the Cast Batching and Sequencing Problem is as follows. Input to the the problem is a set of charges generated by the Charge Pool Generation Problem. One way to represent a solution to the problem is a permutation of all charges available. In case consecutive charges fulfil the batching requirements (constraints of *Cast Batching*), they are considered to be a cast. This representation is similar to family scheduling, where jobs are assigned to specific setup families and consecutive jobs of the same setup family do not require a setup process (see Section 3.3 for the related work). Using this similarity, a formal model for the Cast Batching and Sequencing Problem is developed in Section 4. Besides the (***CS4: hot metal consumption***) constraint, further constraints that can be considered without determining *Slab Design* and *Charge Batching* decisions are respected. For the *Cast Batching* sub problem that is (***CB1: cast family compatibility***). For the *Cast Sequencing* sub problem, (***CS1: caster capacity***) and (***CS2: setup type***). All constraints that involve slab width information and slab position information cannot be respected in the described approach, since this would require a detailed solution to the Charge Pool Generation Problem which is not the focus of this thesis.

The multi-objective character of the problem is separated from the formal model derived for the Cast Batching and Sequencing Problem. Within the model, only the target (***O1: duedate fulfilment***) is respected by minimizing total tardiness of all charges. However, other targets of the detailed continuous casting problem are treated within the DSS.

The following chapter contains the related work for the continuous casting planning problem and the scheduling literature that is related to the modelling approach selected for the Cast Batching and Sequencing problem. Further, the reasoning for selecting the above approach is deduced from limitations in the current continuous casting literature.

Within the scheduling part of Chapter 3, the extensions of existing scheduling approaches to model and solve the Cast Batching and Sequencing problem are pointed out.

3 Related Work

This chapter presents the related work that forms the foundation of this thesis. In Section 3.1, the state of the art in continuous casting planning is presented. Section 3.2 contains a discussion of the limitations of existing literature. Based on these findings, the decision on how to approach the problem described in Section 2.5 was deduced. Section 3.3 presents the state of the art in scheduling relevant to the chosen approach. Finally, Section 3.4 discusses required extensions of existing scheduling approaches in order to realize the attempted approach of this thesis.

3.1 Continuous Casting Planning Problem

As described in Section 2, the detailed continuous casting planning problem consists of the five interrelated sub problems: Caster selection, slab design, charge batching, cast batching, and sequence selection. Because the combined continuous casting planning problem is very complex, available approaches in the literature tend to only focus on parts of the given decisions, constraints and/or objectives. The following sections are structured according to the degree of sub problem considerations within the problems considered in the literature. Since there are no papers explicitly concerned with the caster selection sub problem, it is not presented in an independent section. The first Section 3.1.1 considers the relevant literature on the slab design problem. The second Section 3.1.2 considers literature that focuses on problems including charge batching and potentially slab design. The following Section 3.1.3 presents literature on problems that derive casts, potentially including charge batching and slab design considerations. Finally, Section 3.1.4 provides an overview of existing literature that determines sequences of casts for a caster and therewith final solutions to the continuous casting planning problem.

General surveys that are relevant to the continuous casting problem have been conducted by Lee et al. (1996) and Tang et al. (2001).

3.1.1 Slab Design

There are two streams of literature that focus on the slab design problem. The first stream only considers order weights and slab weights as well as an order type (color) when grouping orders to slabs. The literature is primarily focused on developing new constraint programming solution approaches. The second stream of literature is focused on real world planning problems and considers more constraints and objectives.

All problems of the first stream focus on constraint (***SD3: min/max slab weight***). These problems also consider a color for each order, representing the routing of the order. In order to avoid additional adjusting processes (***O10: adjusting***), the problems in the first stream place a constraint on the number of colors that are allowed to be grouped into one slab. Frisch et al. (2001a) described three models with different choices of decision variables for the steel mill slab design problem. They presented constructive heuristics to solve small problem instances. In a second publication, the authors presented another model that is solved using constraint programming (Frisch et al. 2001b). Gargani and Refalo (2007) modelled the problem using a constraint programming formulation. They presented specified search strategies and combine them with a large neighbourhood search. Hentenryck and Michel (2008) further modified the search strategy and showed that they can achieve better results even without using large neighbourhood search. Schaus et al. (2010) provided an extensive comparison of different approaches developed for the problem based on existing and developed test instances.

The second stream of literature is concerned with solving real industry problems. Dawande et al. (2004) considered (***SD1: min/max slab length***), (***SD2: min/max slab width***), (***SD3: min/max slab weight***), (***CHB1: steel quality compatibility***) and (***CB3: slab quality compatibility***). As targets they considered (***O10: adjusting***) via a color constraint (similar to literature stream one), (***O8: caster utilization***) by generating as few as possible, thus large slabs, and (***O6: open ordered slabs***) by minimizing the surplus material produced when grouping unequal daughter slabs on one mother slab. The authors developed a heuristic that consists of three steps. First a graph is constructed for all orders that represent possible compatible order combinations. They used existing algorithms to estimate the maximum weight matching of orders in the graph to obtain initial slabs. Second, they used a constructive heuristic based on the idea of bin packing to allocate unassigned orders to slabs and reassign orders in existing slabs. While slab weights are maximized in the first two steps, step three consists of resizing slabs to include orders with small weights.

Dash et al. (2007) combined the slab design problem with the problem of generating only a subset of possible slabs depending on the caster capacity. Dimensional constraints (***SD1: min/max slab length***), (***SD2: min/max slab width***) and (***SD3: min/max slab weight***) were taken into consideration. The authors focused on the targets (***O1: duedate fulfilment***) by minimizing unselected rush orders and maximizing

order completeness, (***O6: open ordered slabs***) by minimizing the surplus ratio and (***O8: caster utilization***) by maximizing the slab weight. In their solution approach they used the order geometry information to create possible slab patterns that could be used to fulfil the orders. With this structure they were able to exploit the similarity to the cutting stock problem and solved it using a column generation approach. They provided an integer problem formulation and described different solution approaches to solve the sub problem.

3.1.2 Charge Batching

Lee et al. (2004) developed an algorithm based in interval graphs to perform charge batching for a single strand caster. For each order, a weight and a range of minimum-maximum width requirements were given. They assumed that each charge can only be cast in one single width. The target of their approach was to find a minimum set of charges with respective width, such that all orders are completely covered. The authors derived properties of an optimal solution and presented an exact algorithm for the problem.

Tang and Wang (2008) developed a MIP model for the *Charge Batching* problem of a single strand caster. The problem was modelled as a p-median problem to allocate given slabs to a minimum amount of median slabs to form charges. The objectives minimizing (***O4: slab quality upgrading***) and minimizing (***O6: open ordered slabs***) were considered. Further, the authors translated due date information into priorities and minimized the number of unselected slabs with high priority. This objective contributed to the (***O1: duedate fulfilment***) objective. Regarding constraints, their model considered (***CHB1: steel quality compatibility***), (***CHB3: slab width transition***) and (***CHB2: charge size***). In order to respect higher level constraints and objectives (associated with *Sequence Selection*), the authors defined constraints on the minimum number of charges generated with specific characteristics, i.e. (***CS5: vacuum treatment capacity***), (***O2: up/downstream capacity utilization***). To solve the model they developed a dynamic programming based heuristic that separates slabs into subsets according to steel grades. Afterwards, a shortest path problem was solved considering weights and width information as arc weights.

Tang and Jiang (2009) derived an IP for the *Charge Batching* problem that transfers orders into charges. Using cost penalties in the objective function, the constraints (***CHB1: steel quality compatibility***) and (***CHB3: slab width transition***) were considered. Further they considered the higher planning objective (***O1: duedate fulfilment***) by penalizing due date deviations within a charge. The (***CHB2: charge size***) was respected explicitly as a constraint in the model. In order to model the allocation of every order to exactly one charge they used an assignment constraint. They solved the problem by applying Lagrangian relaxation to the assignment constraint.

3.1.3 Cast Batching

Chang et al. (2000) developed an IP model for the *Charge Batching* problem. They considered the constraint (***CHB3: slab width transition***) as maximum width difference between charges grouped into a single cast. The assumed that all slabs within one charge have the same width. Further (***CB1: cast family compatibility***) was included as maximum carbon and manganese content difference for all charges in a cast, and (***CB2: maximum batch size***) as the total casting time of charges within a cast. The objective of their model was to derive a minimum number of casts for all charges, which is equivalent with (***O9: tundish utilization***) since large casts are generated. They reformulated the problem as a knapsack problem. Afterwards, the problem was solved using a heuristic column generation approach, where fractional solutions were converted into binary solution using a round of scheme.

Tang and Wang (2008) developed a MIP model for the *Cast Batching* problem that allocates charges to casts and determines the sequence of charges within each cast. The authors considered (***CHB3: slab width transition***) as maximum width differences within a cast, (***CB2: maximum batch size***) as total casting time of charges within a cast and (***CB1: cast family compatibility***). Additionally, higher level constraints and objectives (associated with *Sequence Selection*) were treated as constraints, i.e. (***CS5: vacuum treatment capacity***) and (***O3: downstream demand fulfilment***). As further objectives, maximizing (***O9: tundish utilization***), minimizing (***O5: steel quality upgrading***) and maximize (***O1: duedate fulfilment***) as minimizing unselected charges with high priority were used. The input set of charges were derived from their approach presented in Section 3.1.2. More charges were generated than the solution required in order to gain flexibility for the *Cast Batching* problem. The problem was solved using a tabu search based heuristic. First, given charges were grouped into sub sets of similar steel grades. Second, casts were generated by grouping charges according to width requirements and maximum batch size. Third, casts were greedily selected into an initial solution cast list, respecting a defined caster capacity. Afterwards, the solution cast list was modified by applying a tabu search approach. Different local search moves that exchange, insert, delete or swap charges from unselected casts and the casts in the solution cast list were applied. While executing the modifications, feasibility regarding (***CB1: cast family compatibility***), (***CHB3: slab width transition***) and (***CB2: maximum batch size***) was assessed. They implemented an iterated local search approach to escape local optima, but did not describe how the perturbation was executed.

Wang and Tang (2008) formulated a MILP model to derive charge-lots, i.e. multiple charges of the same steel grade, from order information. This can be understood as a combined *Slab Design* and *Charge Batching* approach, that is extended by a preliminary batching of charges. The decisions made in the model consist of assigning order weights to charge-lots, determining the number of slabs generated for each order and to

determine the number of charges in each charge-lot. They considered the objectives to minimize (***O6: open ordered slabs***), (***O1: duedate fulfilment***) as minimizing the amount of unselected order weights and minimizing an assignment costs consisting of (***O5: steel quality upgrading***) and (***O10: adjusting***) resulting from differences in width. As explicit constraints, their model respected the (***CHB2: charge size***) and (***SD3: min/max slab weight***). They solved the problem by developing a methodology combining Lagrangian relaxation on the order assignment constraint, subgradient, dynamic programming and heuristic approaches. The approach was further improved in a later publication by combining Lagrangian relaxation with a column generation to solve the charge-lot batching problem with the same constraints and objectives respected (Tang et al. 2011).

In their most recent paper, Tang et al. (2014) developed a MILP model for the *Cast Batching* problem on a single strand caster, which also considers the sequence of charges within a cast and the determination of casting width into the decision problem. For a given set of charges, their model utilized binary three index decision variables that determined if two charges are cast consecutively with equal or deviating width. They assumed that at most one width change is possible per charge and a specific cost is associated with this width change due to required rework. The chosen objective function tried to maximize a defined reward for producing a cast, i.e. to maximize (***O9: tundish utilization***), reduced by (***O10: adjusting***) penalties resulting from width changes and (***O5: steel quality upgrading***) penalties. They considered the constraints (***CB1: cast family compatibility***), (***CHB3: slab width transition***) and (***CB2: maximum batch size***). The problem was solved using a column generation approach utilizing properties of an optimal solution derived by the authors.

3.1.4 Sequence Selection

Box and Herbe (1988) described a expert system approach for scheduling a twin strand caster based on a given slab pool. In their approach, a decision maker had to specify a cast by choosing starting and ending width for both strands as well as a desired number of charges for a certain cast family. Afterwards, slabs were selected one at a time from a presorted list. The sequence was derived using a penalty evaluation based on (***O10: adjusting***) resulting from slab width transitions, (***O7: HSM furnace utilization***), (***O1: duedate fulfilment***) and (***O9: tundish utilization***). Further, each extension was assessed regarding feasibility based on (***CB3: slab quality compatibility***), (***CS4: hot metal consumption***), (***CHB3: slab width transition***), (***SD4: slab length cluster***) and (***CB4: strand length difference***). Alternative casts were generated and evaluated using the expert system approach. The sequence of casts was derived based on a ranking of generated casts evaluated by the described targets.

Most approaches that consider the sequencing of casts are primarily concerned with

scheduling the steel making stage, which is the predecessor of continuous casting. Bellabdaoui and Teghem (2006) studied the combined steel making continuous casting problem. However, they considered the sequence of charges as given input and concentrated on deriving a complementary production plan for steel making based on a MILP model. The objective was to minimizes total completion times under various steel making constraints. Numao and Morishita (1989) developed an expert system for the combined steel making continuous casting problem, in which the planner can make modification that are evaluated by the system according to feasibility and the performance criteria of minimizing waiting times and maximizing output. Their approach focused on conflicts between different production steps in steel making.

Tang et al. (2000) developed a linear programming model in which continuous casting was considered to be an unconstrained single machine sequencing problem and the sequence was primarily determined by the steel making constraints and objectives.

Harjunkoski and Grossmann (2001) developed a decomposition approach for the combined steel making and continuous casting problem for a single strand caster. In a first step, products (i.e. slabs) were presorted according to charge and cast batching requirements (i.e. steel grades, thickness and width). Second, a MILP model was developed that allocated all given slabs to casts considering (***CHB3: slab width transition***) and (***CHB1: steel quality compatibility***) requirements via precedence constraints and (***CB2: maximum batch size***) of the casts. The objective of the model was to derive a minimum number of casts for the given set of slabs, thus (***O8: caster utilization***). Sorting the generated casts in descending order of containing slabs, a production program for the continuous caster was derived. In the next step, a MILP model for scheduling the steel making facilities was derived based on a job shop scheduling approach. The third step consisted of a MILP model assigning positions for each cast on the continuous caster and the associated steel making production steps. This model considered due dates of the casts (derived from earliest slab in a cast), the makespan and thickness changes between casts as objectives, without considering additional constraints. In a final step, an LP was designed to compress the schedule by solving the MILP model from step two with fixed binary variables according to the sequence of casts generated from the previous MILP model in step four.

Missbauer et al. (2009) developed a scheduling system for the combined steel making and continuous casting problem. They decomposed the scheduling of casts on the caster from scheduling the charges on the steel making facilities. For both problems they present MILP models. The decision they considered for the continuous casting problem was to determine production times as well as starting times for each charge on the caster, whereas the sequence of casts and containing charges was considered as predefined by a higher planning level and not subject of the decision problem. In their model, they were able to consider setup and waiting times between charges (***CS2:setup type***) as well as hot metal supply by the furnace and hot metal consumption of charges (***CS4: hot metal consumption***). Feasibility was achieved by introducing additional waiting

times in case the hot metal consumption exceeded the hot metal supply from the blast furnace. Various additional steel making constraints, such as routing on different facilities, transportation, or crane capacities and scrap supply, were considered. They used forward and backward constructive heuristics to solve the problem.

Wichmann et al. (2014) considered a single strand continuous casting problem. Input to their problem was a set of slabs that needed to be sequenced for a single strand caster. A sequence of slabs automatically presented charges, casts and cast sequences and therefore combined charge batching, cast batching and sequence selection into one problem. They considered the objectives (***O6:open ordered slabs***), (***O8:caster utilization***) and (***O9:tundish utilization***). In a feasible sequence of slabs they considered the constraints (***CHB3: slab width transition***) and (***CS2: setup type***). The authors presented a MILP model for the problem with a weighted target function. Since only small instances were solvable with a commercial solver, they presented a greedy random adaptive search procedure (GRASP) heuristic that consisted of a construction phase to generate an initial feasible solution and a local search phase to improve the solution with regards to the described targets. The construction was established by starting with an empty sequence and iteratively adding slabs that are evaluated best according to an incremental objective function increase. In the local search phase, the two simple operators were used, that (1) try to combine batches with equal steel grade and (2) move batches within the sequence of slabs to increase the objective function.

3.2 Limitations of Current Approaches for the Continuous Casting Planning Problem

A fundamental constraint of the continuous casting problem is to sufficiently consume the hot metal continuously supplied by the blast furnace. The consumption rate of a schedule is primarily determined by the number of setup processes (defined by the number of charges grouped into casts) and the production speed (defined by the strand width and casting speed of scheduled charges).

One stream of existing literature focuses mainly on batching aspects and only implicitly consider the hot metal consumption constraint by maximizing the number of charges in a cast (see Section 3.1.3). The other stream of literature that explicitly considers the hot metal consumption constraint (see Section 3.1.4) and has its primary focus on the steel making scheduling and the sequence of charges and casts which is assumed to be given from a higher planning level.

The only source that includes both aspects, batching and sequencing, is Box and Herbe (1988), and it does not provide a sufficient mathematical description that could be used

to implement their approach as part of a DSS.

3.3 Job Scheduling

The basic scheduling problem consists of a set of jobs $j \in J$ that needs to be sequenced on one or more machines (Conway et al. 2003). Typical parameters given for jobs can be the time that it takes to process one job p_j, a release date r_j when the job is ready to start processing, a due date d_j that represents the time when the job should be produced determined by a higher planning level and/or a priority weight w_j that differentiates jobs according to their importance. The solution to a scheduling problem is a permutation of jobs for the available machines. Depending on the sequencing decision, the completion time C_j of each job $j \in J$ is determined and different objectives can be evaluated.

Based on Graham et al. (1979) classification scheme, scheduling problems can be described by three categories $\{\alpha,\beta,\gamma\}$. The machine layout (α), the problem conditions and constraints (β) and the problem objectives (γ).

α - Machine Layout

The machine layout describes the type of system studied for a problem. It is determined by the number of machines m under consideration and the connection of these machines. Table 3.1 lists different machine layouts that have been studied in the scheduling literature.

β - Characteristics and Constraints

Besides the machine configuration, there are several characteristics and constraints in practical scheduling problems that need to be considered. These include conditions that prohibit certain production sequences or additional information given for jobs that is required for a certain evaluation. Table 3.2 lists different characteristics and constraints that have been studied in the scheduling literature.

The table provides a rough overview and does not claim to be comprehensive. In case of setup time and costs, there are several detailed aspects available, which are relevant for this thesis. They will be discussed in detail in Section 3.3.2. The same holds for scheduling with resource constraints, which is discussed in Section 3.3.3.

Abbreviation	Name	Description
1	single machine	all jobs are produced on one machine
P	identical parallel machines	jobs are produced on m identical machines
U	uniform parallel machines	jobs are produced on m machines with different job processing times on each machine based on the machine precessing speed
R	unrelated parallel machines	jobs are produced on m machines with different job processing times given on each machine
F	flow shop	jobs are produced on s production stages with one machine on each stage
FF	flexible flow shop	job are produced on s production stages with m parallel machines on the different stages
AF	assembly flow shop	jobs are produced on multiple machine on a first stage and on only one machine in the subsequent stage
J	job shop	jobs are produced on s stages with m machines and each job has a unique routing that could leave out stages or require repeated processing on one stage.
O	open shop	jobs are produced on s stages and each job needs to be produced once on each machine with different job routings
SC	supply chain	jobs are produced on multiple facilities that are of any of the described shop types
NC	machine availability	determines the patterns in which the m machines are available

Table 3.1: Selection of machine layouts studied in the scheduling literature

Abbreviation	Name	Description
prec	precedence	rules that predetermine the sequence of certain jobs
M_j	machine eligibility	only a subset of the available machines can be used for job j
pmtn	preemption	temporarily interrupt processing of a job
W_l	workforce	only a limited number of operators are available in pool l for the m machines
r_j	release date	a release date is given for each job and processing cannot start before that date
b	batch size	the maximum number of jobs that can be batched on a machine
s	setup time or costs	a certain setup time or cost can be assigned to a transition between two consecutive jobs on a machine
rs	resource	a supplied resource is consumed by the jobs of a schedule
p_{ir}	learning effects	the processing time of a job decreases depending on the time or position of the job in the sequence

Table 3.2: Selection of characteristics and constraints studied in the scheduling literature

A further distinction between deterministic scheduling where all information (e.g. job processing times) are given as fixed parameters, and stochastic scheduling where certain information could be described as random variables can be made. Further in static scheduling problems, all information is known at once. In dynamic scheduling problems, the information becomes available during the course of scheduling.

γ - performance measures

The solution to a scheduling problem is a permutation of jobs for the available machines. Whenever the constraints are fulfilled, a schedule is considered to be feasible. However, one sequence of jobs could be considered superior to another. Depending on the preferences of the scheduler, different measures have been developed to compare alternative schedules. Table 3.3 lists different performance measures that have been studied in the scheduling literature.

Any performance measure that is job related could also be used as a weighted version using job specific priority weights w_j. Only the unweighted targets are described in the following.

Abbreviation	Name	Description
C_{max}	makespan	the total completion time of the schedule
L_{max}	maximum lateness	the largest lateness of all jobs
$\sum_{j \in J} E_j$	total earliness	the earliness of all jobs in a schedule
$\sum_{j \in J} T_j$	total tardiness	the tardiness of all jobs in a schedule
$\sum_{j \in J}(E_j + T_j)$	total earliness/tardiness	the sum of earliness and tardiness of all jobs in a schedule
$\sum_{j \in J} U_j$	total late jobs	the number of late jobs in a schedule
T_{max}	maximum tardiness	the largest tardiness of all jobs
$\sum_{j \in J} C_j$	work in process	the sum of completion times for all jobs
$\sum_{j \in J} F_j$	total flow time	the sum of completion times when considering release dates for all jobs

Table 3.3: Selection of characteristics and constraints studied in the scheduling literature

The *Cast Batching and Sequencing Problem*, which is the focus of this PhD Thesis, can be described as follows. First, since *Caster Selection* is not considered, the problem is a **(1) single machine** problem. Second, the described continuous casting production process includes **(2) setup processes** whenever changing from one cast family to another. Third, a constant amount of hot metal is supplied as a **(3) resource** to the caster. Fourth, the major objective of continuous casting planning is to produce orders early enough to avoid late delivery. Therefore **(4) total tardiness** has been selected as the performance measure in this thesis.

The methods developed for specific scheduling problems can often be adopted to suit similar problems. In the following, the existing literature on scheduling problems that share at least two out of the four described parts of the *Cast Batching and Sequencing Problem* are presented.

3.3.1 Job Scheduling to Minimize Total Tardiness on a Single Machine

The scheduling objective total tardiness $\sum_{j \in J} T_j$ for a sequence σ is defined as the sum of tardiness of all jobs in the sequence

$$T_\sigma = \sum_{j \in J} T_j$$

.

Depending on the completion time of each job $j \in J$ in a given sequence σ, the tardiness can be calculated as

$$T_j = \max\{0, C_j - d_j\}$$

.

The solution to a single machine scheduling problem is a permutation of all available jobs. This rather simple setting enables polynomial time algorithms for some of the described performance measures. Jackson (1955) derived the earliest due date (EDD) rule, which sorts jobs in order of increasing due dates, for solving $1||L_{max}$. Smith (1956) developed the *shortest processing time (SPT)* (sorting jobs in order of increasing processing times) and *shortest weighting processing time (SWPT)* (sorting jobs in order of increasing ratio $\frac{p_j}{w_j}$ of processing time and priority weight) rules for solving $1||\sum_{j\in J} C_j$ and $1||\sum_{j\in J} w_j C_j$ in polynomial time. The objective $1||\sum_{j\in J} U_j$ can be solved efficiently using the Moore-Hodgson algorithm (Moore 1968). However, the objective to minimize total tardiness $1||\sum_{j\in J} T_j$ is proven to be NP-hard (Du and Leung 1990) and no polynomial time algorithm exists.

Minimizing Total Tardiness on a Single Machine

Minimizing total tardiness is referred to as a regular performance measure since it does not have the property that the objective is non-decreasing in job completion times (Azizoglu and Webster 1997). Therefore, there is no idle time in an optimal sequence (Koulamas 2010). Reviews on scheduling to minimize total tardiness for a single machine are available from Potts and Van Wassenhove (1991), Koulamas (1994), Sen et al. (2003) and Koulamas (2010).

Most exact algorithms and heuristics that are concerned with minimizing total tardiness use properties that have been derived by Emmons (1969) and Lawler (1977). Emmons (1969) described necessary conditions for the relative order of two jobs $i, j \in J$ based on their due dates and processing times. Lawler (1977) presented a way to decompose a total tardiness problem into two sub problems using information on the longest job $j \in J$. Extensions to those properties have been formulated by Potts and Van Wassenhove (1992), Szwarc (1993),Chang et al. (1995), Yu (1996), Tansel and Sabuncuoglu (1997), Della Croce et al. (1998), Szwarc (1998), Szwarc et al. (1999), Szwarc (2007) and Kanet (2007) as described in the survey of Koulamas (2010). More recently, Kanet (2014) developed a fourth theorem on the ideas of Emmons (1969).

An overview of the literature on single machine scheduling to minimize total tardiness is provided in the following. The literature is clustered according to the solution type. Further details can be reviewed in the mentioned survey papers.

Exact solution approaches have been proposed by Shwimer (1972), Fisher (1976), Pi-

card and Queyranne (1978), Potts and Van Wassenhove (1982), Sen et al. (1983), Sen and Borah (1991), Potts and Van Wassenhove (1992),Kondakci et al. (1994), Szwarc and Mukhopadhyay (1996), Della Croce et al. (1998) Hirakawa (1999), Szwarc et al. (1999),Biskup and Piewitt (2000), Szwarc et al. (2001) and Tansel et al. (2001).

Fully polynomial time approximation schemes have been proposed by Lawler (1982), Kovalyov (1995), Koulamas (2009).

Heuristic algorithms based on constructive heuristics have been proposed by Carroll (1965), Holsenback and Russell (1992), Panwalkar et al. (1993), Fadlalla et al. (1994), Holsenback and Russell (1997), Tansel et al. (2001), Naidu et al. (2002) and Panneerselvam (2006).

Heuristic algorithms based on local search methods have been proposed by Wilkerson and Irwin (1971), Fry et al. (1989), Potts and Van Wassenhove (1991), Antony and Koulamas (1996), Ben-Daya and Al-Fawzan (1996), Sabuncuoglu and Gurgun (1996), Bauer et al. (1999) and Cheng et al. (2009).

More recently, Zhou and Liu (2013) developed a new property based on existing heuristics that decreased calculation times in branch and bound procedures for minimizing total tardiness on a single machine.

Minimizing Total Weighted Tardiness on a Single Machine

The similar problem, minimizing total weighted tardiness on a single machine $1||\sum_{j\in J} w_j T_j$, has been surveyed by Abdul-Razaq et al. (1990), Potts and Van Wassenhove (1991), Sen et al. (2003). The problem is known to be NP-hard (Lenstra et al. 1977).

Exact algorithms have been proposed by Emmons (1969), Lawler (1977), Picard and Queyranne (1978), Potts and Van Wassenhove (1991), Szwarc and Liu (1993) and Selim Akturk and Bayram Yildirim (1998).

Heuristic approaches have been proposed by Crauwels et al. (1998) and Volgenant and Teerhuis (1999).

After the last available survey paper by Sen et al. (2003) several additional papers on the $1||\sum_{j\in J} w_j T_j$ have been published.

Congram et al. (2002) developed an iterated dynasearch algorithm that simultaneously executed a combination of basic local search moves. Avci et al. (2003) developed a problem space algorithm that was based on the application of different dispatching rules.

Grosso et al. (2004) also designed a dynasearch algorithm combining pairwise interchanges with job movement operators. Cheng et al. (2005) modified the algorithm of Lawler (1977) and presented a polynomial-time approximation algorithm for the problem. Holthaus and Rajendran (2005) designed an ant colony algorithm, while Bozejko et al. (2006) presented a tabu search approach.

Kolliopoulos and Steiner (2006) described several properties of the problem and derived a quasi-polynomial time approximation scheme. Ergun and Orlin (2006) described acceleration strategies for increasing solution time of local search heuristics based on job interchange and job move from $O(n^3)$ to $O(n^2)$. Tasgetiren et al. (2006) presented two population based heuristics using particle swarm and differential evolution approaches. In combination with variable neighbourhood search, they were able to show that both heuristics performed very good using benchmark instances from the OR library. Bilge et al. (2007) presented new strategies for designing tabu lists in a tabu search heuristic. Kellegöz et al. (2008) analysed the efficiency of several different crossover procedures generated for genetic algorithm (GA)s to solve the problem. Wodecki (2008) developed dominance properties based on breaking the sequence into blocks. Using these properties in a branch and bound method, the author was able to optimally solve problems with up to 80 jobs. He further presented a way to use parallel computing to reduce calculation times.

Most recently, Wang and Tang (2009) developed a population based variable neighbourhood search algorithm. They combined local search moves and greedy procedures to combine solutions in the population. Their algorithm provided better results at the expense of increased calculation times compared to Avci et al. (2003), Tasgetiren et al. (2006) and Bilge et al. (2007).

3.3.2 Job Scheduling with Setup Considerations

In practice, it is often the case that additional time to adjust tooling is necessary when changing from one job to another on a machine. This is considered by setup times or costs within the scheduling model.

In the general case, setup times are given for each pair $i, j \in J$ and a distinction is made between sequence dependent and sequence independent setup times or costs (Allahverdi et al. 2008). A special case is given whenever different jobs share similar tooling requirements and therefore are allocated to so called setup families (Potts and Kovalyov 2000). In this case, no setup time is necessary when producing consecutive jobs from the same setup family, and a major family setup time is required whenever the family changes. Allahverdi et al. (2008) referred to a sequence of jobs from the same family as a batch. While the allocation of jobs to families is given as a parameter, the allocation of jobs to batches for a certain setup family was part of the decision process. Again, it

can be distinguished between sequence dependent and independent setup times or costs. Figure 3.1 depicts the classification of scheduling with setup considerations developed by Allahverdi et al. (2008).

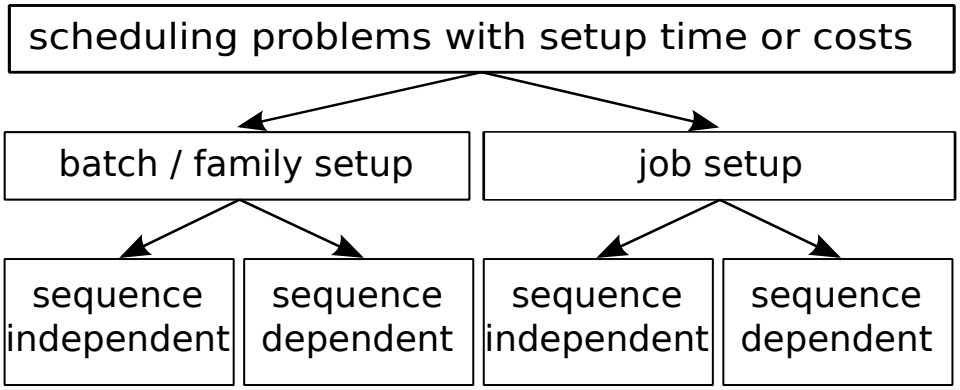

Figure 3.1: Classification of scheduling problems with setup time considerations (according to Allahverdi et al. (2008)

Using the notation scheme from Graham et al. (1979), the different setup considerations can be stated in the β part as follows (Allahverdi et al. 2008). Setup time ST can have index si in case of sequence independence, sd in case of sequence dependence and si, b or sd, b when family/batch setups are considered. The same notation is used for setup costs SC.

For family scheduling, another distinction is made regarding the so called group technology assumption (GTA) (Potts and Van Wassenhove 1992). In case of GTA, all jobs that belong to the same setup family need to be scheduled together. Therefore the problem only consists of finding the sequence of families and the sequence of jobs within each family. In the other case, without GTA, jobs need to be grouped to batches of arbitrary sizes, those batches needs to be sequenced and also the jobs within each batches (Schaller 2007), (Allahverdi et al. 2008).

Family scheduling problems can also be separated into batch availability models and job availability models (Potts and Kovalyov 2000). In the first case, all jobs of the batch become available for further processing only after the entire batch is processed. In the latter case, each job becomes available immediately after it is processed, independent of the completion of other jobs within the batch. A special case of batching occurs in so called batching machines (Potts and Kovalyov 2000). These are e.g. ovens, that are capable of producing multiple jobs simultaneously.

Surveys on scheduling with setup considerations or batch considerations have been conducted by Potts and Van Wassenhove (1992), Webster and Baker (1995), Allahverdi et al. (1999), Potts and Kovalyov (2000) and Allahverdi et al. (2008).

When discussing scheduling with batching, the case of batching machines need to be dis-

tinguished from producing jobs in batches. A batching machine is capable of processing multiple jobs (a batch of jobs) simultaneously, while a regular machine can only process one job of a time. In this thesis, only regular machines are considered. Mathirajan and Sivakumar (2006) provide a survey for problems considering batching machines.

Family Scheduling on a single machine to Minimize Total Tardiness

As described above, in a single machine family scheduling problem to minimize total tardiness $1|ST_b|\sum_{j\in J} T_j$, all jobs are allocated to a certain setup family and no setup time occurs for consecutive jobs of the same family.

Nakamura et al. (1978) considered the problem with GTA and developed an algorithms based on Emmons (1969) conditions.

(Schaller 2007) developed a local search heuristic based on five improvement moves: Console (combining two batches and moving them to position m), Batch Interchange (exchanging pairs of batches), Move (move jobs from one batch to another of equal family), Break (break a batch into two sub batches) and Job Interchange (exchanging pairs of jobs). The author used two procedures to derive initial solutions, one that produced rather large batches and one that was sorting jobs in EDD order. Afterwards, the five improvement moves were applied in a fixed order until no further improvement was achieved.

Gupta and Chantaravarapan (2008) described a MILP model for the problem with GT-assumption. They developed different heuristics, each consisting of a part to schedule jobs within each family and a part to schedule the families. For the sequencing of jobs within each family, the procedure was based on the Panwalker, Smith and Koulamas (PSK) heuristic of Panwalkar et al. (1993) and the net benefit of relocation (NBR) heuristic of Holsenback and Russell (1992). For the scheduling of families, they presented different heuristic approaches based on exchanging and/or moving batches.

Herr and Goel (2014) compared a two-index with a three-index MILP formulation to solve the problem. While the three-index formulation was able to solve problems up to 18 jobs compared to only 9 jobs with 2-index formulation, calculation times dramatically increased for the 3-index case.

Single Machine Scheduling with Sequence-Dependent Job Setup Times to Minimize Total Tardiness

Unlike the family scheduling case, minimizing total tardiness for job setup times on a single machine $1|ST_{sd}|\sum_{j\in J} T_j$ has been studied extensively by various authors.

Rubin and Ragatz (1995) developed a GA and provided benchmark problems with 15-45 jobs. They showed that their GA performs well compared to a previously developed branch and bound heuristic presented by Ragatz (1993). Tan and Narasimhan (1997) developed a simulated annealing algorithm and compared it against enumeration and random sequences. In a later publication, Tan et al. (2000) developed a local search heuristic based on pairwise job interchange. The heuristic was tested on the (15-45 job benchmarks by Rubin and Ragatz (1995)) and performed good in comparison to previously developed simulated annealing (Tan and Narasimhan 1997) and GA (Rubin and Ragatz 1995) methods. Armentano and Mazzini (2000) developed another GA and showed that it performs similar to the one from Rubin and Ragatz (1995) based on the (15-45 jobs) benchmark problems. For larger problems the author was able to present that his GA outperforms the constructive heuristic of Lee et al. (1997). França et al. (2001) developed a GA and a memetic algorithm (combination of the GA with local search). Using the (15-45 jobs) benchmark problems the authors were able to present that their memetic algorithm outperforms previous methods. However, the GA of Armentano and Mazzini (2000) was not considered. Gagné et al. (2002) developed an ant colony heuristic to solve the problem. Not considering the GA of Armentano and Mazzini (2000) and the memetic algorithm of França et al. (2001), the authors showed that their ant colony approach was superior for all benchmark problems (15-45 jobs). Further, the authors provided additional benchmark problems with 55-85 jobs and evaluated different parameter settings for their ant colony heuristic.

Armentano and De Araujo (2006) developed a different GRASP heuristics variants with path re-linking and memory-based construction. Their heuristic outperformed the ant colony heuristic of Gagné et al. (2002) and the memetic algorithm of França et al. (2001).

Gupta and Smith (2006) developed two heuristics. First, a GRASP multi-start heuristic that used a specific cost function to generate initial solutions. Afterwards, a local search phase was executed using job interchange and job backwards as well as job forwards moves. Second, a space-based local search that constructed neighbourhood solutions by modifying job processing times. Using both benchmark problem sets (15-45 jobs and 55-85 jobs), the authors were able to show that the GRASP heuristic outperforms the ant colony approach of Gagné et al. (2002) in terms of solution quality but at the expense of computation time. The space-based local search on the other hand performed similar to the ant colony heuristic but required less calculation time. Liao and Juan (2007) also developed an ant colony heuristic combined with local search elements. Not considering

Gupta and Smith (2006), they presented superior performance to Gagné et al. (2002) for all benchmark problems. Lin and Ying (2008) developed a hybrid approach combining simulated annealing and tabu search for job interchange and job move local search operators. With this approach, they were able to outperform the ant colony heuristic of Liao and Juan (2007) in most cases of the benchmark problems. Ying et al. (2009) developed an iterated greedy heuristic based on local search combined with destruction and construction moves. Again using the benchmark problems, the authors were able to further decrease the best known solution or achieve equally good solutions in almost all instances. Sioud et al. (2012) proposed a hybrid GA based on a new crossover procedure combining achieves from multi-objective evolutionary algorithms with transition rules from ant colony approaches. They tested different variants of their method against recent results from the literature using both benchmark instance sets. In parallel, Akrout et al. (2012) developed a combined a GRASP with a population based differential evolution approach and use an additional variable neighbourhood search to further improve the solution quality. They presented that their approach was able to outperform the existing heuristics.

Besides the attempts to generate efficient heuristics for the $1|ST_{sd}|\sum_{j\in J} T_j$ problem, several authors worked on generating exact solution methods. Building upon Ragatz (1993), Souissi and Kacem (2004) developed a branch and bound heuristic. They presented results for different parameter settings for self-generated test instances. Luo and Chu (2006) described several dominance properties and developed a branch and bound algorithm that is able to solve self-generated problem instances with up to 30 jobs. Bigras et al. (2008) reformulated the problem as a time-dependent travelling salesman problem and present different column generation approaches. With this approach they were able to optimally solve all benchmark instances with up to 35 jobs. Not considering Bigras et al. (2008), Sewell et al. (2012) developed a branch and bound algorithm based on adopted pruning strategies from a successful application to $1|r_j|\sum_{j\in J} T_j$ in Kao et al. (2009). They presented superior performance compared to the results from Ragatz (1993). When using their branch and bound as a heuristic the authors presented superior performance on the two benchmark sets compared to the simulated annealing and tabu search approach of Lin and Ying (2008).

Single Machine Scheduling with Sequence-Dependent Job Setup Times to Minimize Total Weighted Tardiness

A similar number of publications can be found for the weighted version of the single machine scheduling problem with sequence dependent job setup times $1|ST_{sd}|\sum_{j\in J} w_j T_j$.

Lee et al. (1997) developed a constructive heuristic based on a newly designed dispatching rule. The performance is evaluated for different parameters and compared to an existing dispatching rule. Cicirello and Smith (2005) generated a set of 120 test instances with

60 jobs and different parameter settings. The authors implemented several stochastic search heuristics and provided the best results for each of the 120 test instances. In the following, these test instances will be referred to as the benchmark instances (for $1|ST_{sd}|\sum_{j\in J} w_jT_j$). Liao and Juan (2007) developed an ant colony heuristic combined with local search elements. They were able to improve 86% of the benchmark instances provided by Cicirello and Smith (2005).

Lin and Ying (2007) implemented a simulated annealing heuristic, a GA and a tabu search heuristic based on swap and insertion moves. While all three approaches achieved similar performance, the authors were able to show that their methods outperform the results from Cicirello and Smith (2005) provided with the benchmark instances. In a follow up publication Lin and Ying (2008) developed a hybrid approach combining simulated annealing and tabu search. Again, they were able to generate superior results for a large number of test instances. Anghinolfi and Paolucci (2008) developed an ant colony heuristic. Considering all previously published methods (except Lin and Ying (2008)), their heuristic was able to outperform existing upper bounds in almost all 120 problem instances.

Ying et al. (2009) developed an iterated greedy heuristic based on local search combined with destruction and construction moves. Compared to previously developed methods by Lin and Ying (2007) and Liao and Juan (2007), they claimed to decrease the average upper bounds of the benchmark problem instances. Valente and Alves (2008) developed a beam search algorithm that only analyses promising branches in an branch and bound procedure. Their method was tested using self generated instances and compared to Lee et al. (1997), not considering recent results. Anghinolfi and Paolucci (2009) created a population based heuristic motivated by particle swarm optimization. Using the benchmark problems, they were able to present improved upper bounds for a large number of instances.

Tasgetiren et al. (2009) developed a discrete differential evolution algorithm that used different procedures to generate initial solutions and combined it with local search procedures. With their approach the authors reduced the upper bounds of 42.5% of the benchmark instances, while 41.67% where equally good compared to the best solution found of existing methods. Liao et al. (2012) discussed the three local search neighbourhoods job interchange, job insertion/move and twist (i.e. reversing the order of a subset of jobs). They provided theorems to reduce calculation times when evaluating a neighbourhood and presented acceleration strategies to reduce the number of evaluation that needs to be executed when exploring a neighbourhood.

Recently Tanaka and Araki (2013) developed an exact algorithm based on a successive sublimation dynamic programming algorithm originally developed for $1||\sum_{j\in J} T_j$ Tanaka and Fujikuma (2012). Subramanian et al. (2014) developed an iterated local search heuristic for the single machine sequence dependent job setup time problem to minimize weighted tardiness $1|ST_{sd}|\sum_{j\in J} w_jT_j$.

Family Scheduling to Minimize total Weighted Tardiness on Machine Layouts other then Single Machine

This section describes family scheduling problems to minimize total weighted tardiness on different machine layouts $X|ST_b|\sum_{j\in J} w_j T_j$. In a recent paper, Schaller (2014) has developed a GA using uniform order-based crossover and job/batch move local search procedure, for the $P|ST_b|\sum_{j\in J} T_j$ problem.

Minimizing Total Tardiness with Sequence-Dependent Job Setup Times on Machine Layouts other then Single Machine

Naderi et al. (2009) solved $FF|ST_{sd}|\sum_{j\in J} C_j, \sum_{j\in J} T_j$ using a simulated annealing based local search heuristic. Ruiz and Stützle (2008) developed an iterated local search heuristic for the $F|ST_{sd}|C_{max}, \sum_{j\in J} T_j$.

3.3.3 Job Scheduling with Resource Constraints

As described in Section 2, the continuous casting stage is fed with liquid steel that is processed from a constant supply of hot metal within steel making. Each charge produced at the continuous casting stage consumes a certain amount of liquid steel and therewith hot metal. A similar situation occurs for example in assembly processes, where material is produced to stock and assembled to order in the last production stage. While make to stock production can be regulated, the supply of hot metal in steel production cannot (see Section 2.3.5).

Figure 3.2 illustrates the implication of the resource constraint on the example of scheduling with setup times. In the figure, the amount of resources available is shown as a solid line with a constant supply rate. The dashed vertical lines illustrate completion times of individual jobs and the cumulative resource demand over time is shown as a solid piece-wise linear curve. Horizontal segments of this curve illustrate times during which the machine is not processing any job because of a setup or because of necessary waiting time required in order to meet the resource constraint.

Job Scheduling with resource consideration did not receive much attention in the literature. The problem was first mentioned by Carlier and Kan (1982) and a first study on complexity as well as classification is given in Blazewicz et al. (1983). Grigoriev et al. (2005) published a survey paper and distinguished three types of resource constraints. First, when each job is allocated to a unique resource supply. Second, all jobs require a common resource supply. Third, there are multiple raw material supplies that could be

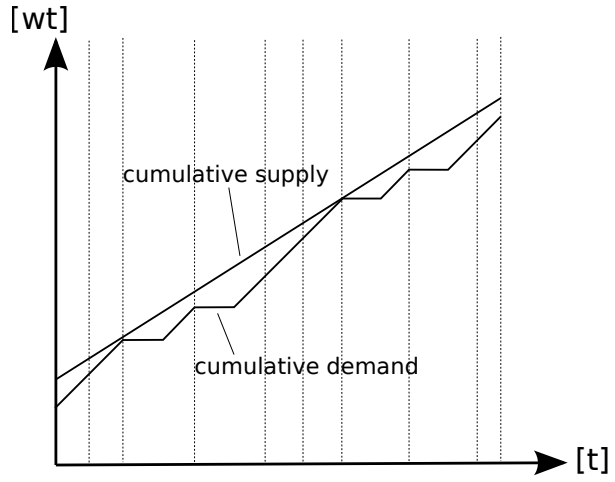

Figure 3.2: Depiction of the resource constraint in the context of scheduling with family setup times.

required by an order. They extended the classification scheme by introducing *ddc* (different dedicated raw materials) for case one, *rm* (m sources of undedicated raw material) for cases two (*rm=1*) and three (*rm=m*).

A different stream of literature that involves resource constraints, deals with complex flexible flow shops with multiple stages, machines, and resources in the process industry (see e.g. Schwindt and Trautmann (2000), Neumann et al. (2005)). Since these sources consider different machine layouts and are usually concerned with minimizing maximum makespan C_{max}, they will not be discussed in detail in this thesis. Resource constraints have also been studied in the context of resource-constrained project scheduling problems (see the review paper of Hartmann and Briskorn (2010) for an overview). Only those problems relevant to the job scheduling problem treated in this thesis will be mentioned below.

Single Machine Scheduling with Resource Constraint and Performance Metrics other then Total Tardiness

Neumann and Schwindt (2002) discussed project scheduling problems with inventory constraints. Instead of considering a time span for tasks, the authors considered to schedule events in a project and modelled the problem as a single machine scheduling problem with resource constraints to minimize makespan $1|rm = m|C_{max}$. Besides the requirement to have sufficient amount of resource available to process a job, they also considered to consume enough resource not to excess an given inventory. They described several properties for the problem and developed a branch and bound algorithm based on these properties.

Grigoriev et al. (2005) proved that the single machine scheduling problem with resource constraints and unit processing times to minimize maximum lateness $1|rm = 1, p_j = 1|L_{max}$ can be solved in polynomial time. Further, the authors showed that $1|rm = 1|C_{max}$ is strongly NP-hard, but in case of equal processing times $p_j = p$ or unit supply of raw material, the problem can be solved in polynomial time.

Briskorn et al. (2009) developed a generic GA and presented different solution representations and evaluation functions to solve $1|rm = 1|\sum_{j\in J} w_j C_j$, $1|rm = 1|\sum_{j\in J} L_{max}$, $1|rm = 1|\sum_{j\in J} w_j U_j$ and $1|rm = 1|\sum_{j\in J} w_j T_j$.

Briskorn et al. (2010) discussed scheduling problems with inventory constraints. They distinguished between the two cases of increasing and decreasing inventories when processing jobs. The latter case is equivalent to consuming a supplied resource. They proved NP-hardness for problems $1|rm = 1|\sum_{j\in J} C_j$, $1|rm = 1|\sum_{j\in J} w_j C_j$, $1|rm = 1|\sum_{j\in J} L_{max}$, $1|rm = 1|\sum_{j\in J} U_j$ and developed polynomial time algorithms for special cases of these problems. The authors further stated, that the single machine scheduling problem with resource constraints has not been studied for the objectives to minimize total tardiness and minimize total weighted tardiness.

In a later publication Briskorn et al. (2013) discussed different properties for the single machine scheduling problem with resource constraints to minimize total weighted completion times $1|rm = 1|\sum_{j\in J} w_j C_j$. Based on these properties the authors developed a branch and bound algorithm that is able to solve with up to 20 jobs.

Drótos and Kis (2013) discussed scheduling problems with inventory releasing jobs. They provide complexity discussions for special cases of the problem and a fully polynomial approximation scheme a special case of the minimize total tardiness problem. Based on these findings, Györgyi and Kis (2014) studied the problem $1|rm = 1|C_{max}$ and proved that there exists no polynomial time approximation scheme for this problem. However, they provided polynomial time algorithms for special cases of the problem (for specific supply patterns of the resource).

3.4 Necessary Extensions of Current Scheduling Approaches for the Cast Batching and Sequencing Problem

The requirements of the (***CS4: hot metal consumption***) constraint have not been sufficiently analysed in the existing scheduling literature. Especially in the combined consideration with family scheduling. Further, benchmark test instances, that enable a

comparison of different solution approaches, have not been published for family scheduling problems.

4 Models for the combined Charge Batching and Sequencing Problem

The combined charge batching and sequencing problem, as studied in this thesis, can be described as a single machine family scheduling problem with upper and lower resource constraints, to minimize total tardiness. Within the following sections, the specific aspect of each constraint is presented by gradually extending a basic single machine family scheduling model. Further, the link between the model variants and the constraints derived from practical requirements are discussed. Models 2 and 3 have originally been presented by Herr and Goel (2015).

4.1 Model 1 - Single Machine Family Scheduling Model

Input to the combined charge batching and sequencing problem is a set of charges that needs to be processed on a single continuous caster. A solution to the problem is given as a permutation of all charges. A charge will be referred to as a job in the following, in order to use common scheduling notation. As described in Chapter 2, setup processes occur during continuous casting whenever two consecutive charges are from different cast families. This requirement can be modelled using a family scheduling approach, where each job is assigned to a specific setup family, and setups occur whenever consecutive jobs are from different setup families.

4.1.1 Notation

Let J denote a set of jobs to be processed by a single machine. Each job is characterised by a due date d_j and a processing time p_j. Furthermore, each job belongs to a given setup family f_j. For any pair of jobs $i, j \in J$ with $f_i \neq f_j$, a setup of duration s_{ij} is required between processing jobs i and j. A single machine is available that can process one job at a time and preemption is not allowed. A solution to the problem is represented by a sequence σ of all jobs. The completion time of a job the solution is represented by C_j.

The goal is to find a sequence that minimises total tardiness, with tardiness calculated as $T_j = \max\{0, C_j - d_j\}$.

Let us now assume that each job has to be processed without interruption and that the machine can be idle for any period of time after completion of one job and before starting to process the next job. For the ease of notation it is assumed that $s_{ij} = 0$ for any pair of jobs $i, j \in J$ with $f_i = f_j$. Furthermore, it is assumed that each schedule begins with a dummy job j^* that is included in J. This job has a due date far enough in the future such as it will never be late, has zero precessing time and resource demand, and no setup time is required before processing any other job. Let x_{ij} denote a binary variable indicating whether job $j \in J$ is scheduled immediately after job i ($x_{ij} = 1$) or not ($x_{ij} = 0$).

4.1.2 MILP Formulation

Model 1 can be described by the following MILP:

$$\text{minimize} \sum_{j \in J} T_j \tag{4.1}$$

subject to

$$\sum_{i \in J \setminus \{j\}} x_{ij} = 1 \text{ for all } j \in J \tag{4.2a}$$

$$\sum_{j \in J \setminus \{i\}} x_{ij} = 1 \text{ for all } i \in J \tag{4.2b}$$

$$x_{jj} = 0 \text{ for all } j \in J \tag{4.2c}$$

$$C_{j^*} = 0 \tag{4.3a}$$

$$C_j \geq C_i + s_{ij} + p_j - (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.3b}$$

$$T_j \geq 0 \tag{4.4a}$$

$$T_j \geq C_j - d_j \text{ for all } j \in J \tag{4.4b}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j \in J \tag{4.5}$$

The objective (4.1) is to minimize the sum of tardiness of all jobs in the production schedule. Constraints (4.2a), (4.2b) and (4.2c) require each job to be fulfilled exactly once. Constraint (4.3a) and (4.3b) require that the completion time of a job must be at least as large as the the completion time of the preceding job plus the setup time – if a setup is required – and the processing time. Constraints (4.4a) and (4.4b) restrict the tardiness of each job. Finally, constraint (4.5) restricts the domain of the binary decision variables.

4.2 Model 2 - Single Machine Family Scheduling with Upper Resource Constraint

As described in Section 2.3.5, the continuous caster is fed with a supply of hot metal with a constant rate. The hot metal is processed into the required hot steel and transferred to the casting staqe in charges, consuming a certain amount of hot metal.

Translated into the scheduling problem, a resource is supplied at a constant rate. Each job consumes a certain amount of resource whenever it is produced. Jobs can only be produced whenever a sufficient amount of resource is supplied. Whenever a job in a given sequence is short of resource, the production has to wait.

4.2.1 Notation

Again, let J denote a set of jobs to be processed by a single machine. Each job is characterised by a due date d_j, a processing time p_j, a job family f_j, a completion time C_j and a tardiness T_j. A setup time s_{ij} is defined for each pair of $i, j \in J$, with $s_{ij} = 0$ for $f_i = f_j$ and a binary variable x_{ij} indicating whether job $j \in J$ is scheduled immediately after job i ($x_{ij} = 1$) or not ($x_{ij} = 0$).

The resource required is supplied with a constant rate of r per unit of time, and initially an amount of r^* of the resource is available. For each job, the quantity q_j of a resource required by the machine to process the job is given. It is assumed that the resource is consumed at a constant rate q_j/p_j. For each job $j \in J$ let Q_j denote the accumulated amount of the resource requirements.

The fundamental difference of this problem to the case without resource constraints is, that it may be necessary that the machine is idle because the required resource for the next job is net yet available, whereas in the case without resource constraints the machine is only idle for the time of the setups that may be required. In the case without resource constraints the optimal duration of any subsequence of jobs is always the sum

of all processing times and the required setups. Figure 4.1 shows an example where the minimum duration of a sequence is larger than the sum of processing and setups times because the machine has to wait for the resource required. In the example shown in the figure, it is not possible to schedule job j immediately after job i and the completion of the setup, because the cumulative resource requirements exceeds the cumulative resource supply if no additional waiting time is scheduled.

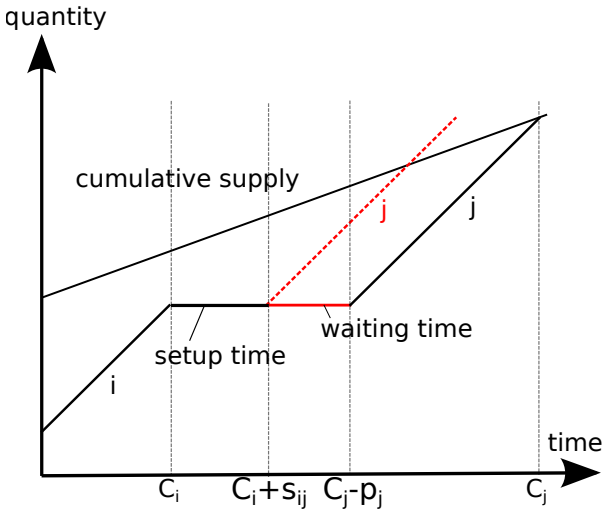

Figure 4.1: Example how required waiting time can result in an increased completion time.

Without resource constraints, any sequence of jobs of the same family can be reordered without impacting the completion time of this sequence and the tardiness of subsequent jobs. Therefore, it is possible to locally optimise the order, in which jobs of the same family are scheduled. In the presence of resource constraints, however, any permutation of jobs may lead to an increase or decrease of the cumulative duration due to necessary waiting times. Therefore, the tardiness of subsequent jobs may change if the order of jobs within a subsequence is modified.

Because of the resource constraints it is necessary to be able to efficiently determine completion times and tardiness of all jobs in a sequence.

4.2.2 MILP Formulation

Model 2 can be described by the following MILP:

$$\text{minimize} \sum_{j \in J} T_j \tag{4.6}$$

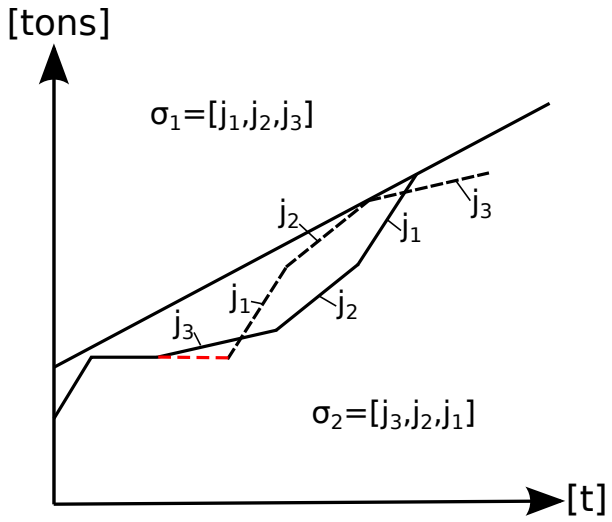

Figure 4.2: Example on how the sequence of jobs within a batch can influence the feasibility of a sequence.

subject to

$$\sum_{i \in J \setminus \{j\}} x_{ij} = 1 \text{ for all } j \in J \tag{4.7a}$$

$$\sum_{j \in J \setminus \{i\}} x_{ij} = 1 \text{ for all } i \in J \tag{4.7b}$$

$$x_{jj} = 0 \text{ for all } j \in J \tag{4.7c}$$

$$C_{j^*} = 0 \tag{4.8a}$$

$$C_j \geq C_i + s_{ij} + p_j - (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.8b}$$

$$T_j \geq 0 \tag{4.9a}$$

$$T_j \geq C_j - d_j \text{ for all } j \in J \tag{4.9b}$$

$$Q_{j^*} = 0 \tag{4.10a}$$

$$Q_j \geq Q_i + q_j - (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.10b}$$

$$Q_j \leq r^* + rC_j \text{ for all } j \in J \tag{4.11}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j \in J \tag{4.12}$$

The objective (4.6) is to minimize the sum of tardiness of all jobs in the production schedule. Constraints (4.7a), (4.7b) and (4.7c) require each job to be fulfilled exactly once. Constraint (4.8a) and (4.8b) require that the completion time of a job must be at least as large as the the completion time of the preceding job plus the setup time – if a setup is required – and the processing time. Constraints (4.9a) and (4.9b) restrict the tardiness of each job and constraints (4.10a) and (4.10b) the cumulative resource requirements before any job. The cumulative resource requirements must not exceed the amount available until completion of the job as required by constraint (4.11). Finally, constraint (4.12) restricts the domain of the binary decision variables.

4.3 Model 3 - Single Machine Family Scheduling with Upper Resource Constraint and Setup Constraint

The ability to wait after any charge until enough resource is supplied, as used in Model 2, is not valid for continuous casting production. When producing charges from the same cast family, consecutive charges need to be produced without interruption. This is due to the inability to stop the caster during production, because of quality and technological requirements.

Translated into the scheduling problem, waiting time between two jobs can only be included whenever a setup process is executed.

4.3.1 Notation

Again, let J denote a set of jobs to be processed by a single machine. Each job is characterised by a due date d_j, a processing time p_j, a quantity of resource requirement q_j, a job family f_j, a completion time C_j and a tardiness T_j. The resource is supplied at rate r and initial amount $r^\star$ of the resource is available. A setup time s_{ij} is defined for each pair of $i, j \in J$, with $s_{ij} = 0$ for $f_i = f_j$ and a binary variable x_{ij} indicating whether job $j \in J$ is scheduled immediately after job i ($x_{ij} = 1$) or not ($x_{ij} = 0$).

Because of operational requirements, it is not always possible that a machine can be idle for an arbitrary period of time after completion of one job and before starting to process the next job. This is the case, if a machine must be brought to a state in which it can remain idle, e.g. if cleaning is required. For such cases, it can be assumed that

the duration of a setup that may be required between jobs of different families can be increased by an arbitrary amount of time, however, an additional setup between jobs of the same family is required if the job does not start immediately after completion of the preceding job. Model 3, is therefore extended to have $s_{ij} > 0$ for jobs $i, j \in J$ with $f_i = f_j$. It is assumed that for each family, the setup time is identical for any pair of jobs in the family. As this setup between jobs of the same family is not always necessary, a binary variable y_{ij}, indicating whether a setup is conducted between jobs i and j, or not, is introduced to the model.

Obviously, any feasible solution for Model 3 is also a feasible solution for Model 2. However, the sequences of jobs that are optimal for the two Models may differ. Let us consider the example with two jobs belonging to the same setup family with parameters $d_1 = 0$, $p_1 = 14$ and $d_2 = 15$, $p_2 = 12$. Furthermore, assume that at the beginning of the planning horizon there is an initial inventory that is sufficient to process either of the jobs without delay, but not both, and that after 30 units of time the resource supply reaches a level that is sufficient to process both jobs. Furthermore, let us assume that for Model 3, the duration of a setup between the two jobs is sufficiently large, so that an optimal solution will not have a setup between the two jobs. Figure 4.3 illustrates the optimal sequences for both Models. The optimal sequence is illustrated by a solid line, whereas the inferior sequence by a dashed line. For Model 2, where the machine may be idle for any period of time after completion of one job and before starting to process the next job, the optimal sequence is to process job 1 before job 2 with $C_1 = 14$, $T_1 = 14$, $C_2 = 30$, $T_2 = 15$ and a total tardiness of 29. If job 2 is processed before job 1, there is $C_1 = 30$, $T_1 = 30$, $C_2 = 12$, $T_2 = 0$ and a total tardiness of 30. For Model 3, where an additional setup between jobs of the same family is required if the job does not start immediately after completion of the preceding job, it is better to delay the start of the first job in the optimal sequence instead of adding an additional setup. The optimal sequence is to process job 2 before job 1 with $C_1 = 30$, $T_1 = 30$, $C_2 = 16$, $T_2 = 1$ and a total tardiness of 31. If job 1 is processed before job 2, there is $C_1 = 18$, $T_1 = 18$, $C_2 = 30$, $T_2 = 15$ and a total tardiness of 33.

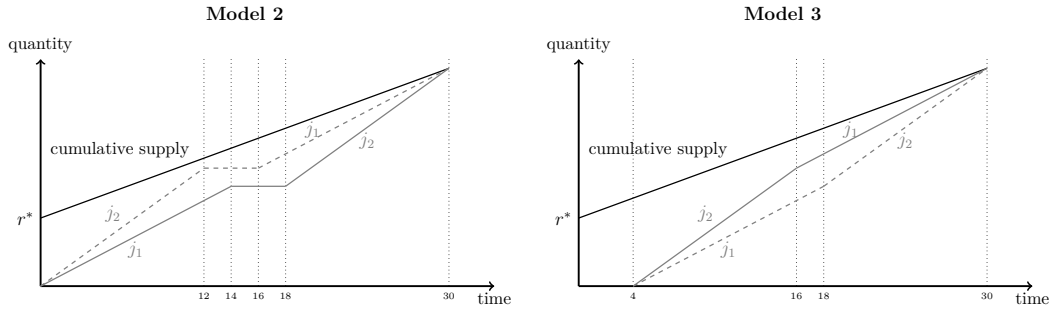

Figure 4.3: Example on how the optimal sequences can differ for both problem variants.

As the above example and the example of Figure 4.2 illustrate, the optimal sequence depends on both the resource availability as well as the operational details determining

when necessary waiting times can be scheduled. Thus, the previously developed optimality conditions for minimising total tardiness, e.g. those presented by Schaller (2007), are not valid for scheduling with resource constraints. As a result, a solution approach cannot exploit these properties.

4.3.2 MILP Formulation

Due to the described extensions of Model 3, constraints need to be added to the MILP formulation to respect additional setups between jobs from the same family. Further, the introduction of additional setups possibly changes the completion times and therefore constraints (4.8a) and (4.8b) need to be modified.

Model 3 can be described by the following MILP:

$$\text{minimize} \sum_{j \in J} T_j \tag{4.13}$$

subject to

$$\sum_{i \in J \setminus \{j\}} x_{ij} = 1 \text{ for all } j \in J \tag{4.14a}$$

$$\sum_{j \in J \setminus \{i\}} x_{ij} = 1 \text{ for all } i \in J \tag{4.14b}$$

$$x_{jj} = 0 \text{ for all } j \in J \tag{4.14c}$$

$$C_{j^*} = 0 \tag{4.15a}$$

$$C_j \geq C_i + y_{ij} s_{ij} + p_j - (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.15b}$$

$$C_j \leq C_i + p_j + (1 - x_{ij} + y_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\}. \tag{4.15c}$$

$$T_j \geq 0 \tag{4.16a}$$

$$T_j \geq C_j - d_j \text{ for all } j \in J \tag{4.16b}$$

$$y_{ij} = x_{ij} \text{ for all } i, j \in J : f_i \neq f_j \tag{4.17a}$$

$$y_{ij} \leq x_{ij} \text{ for all } i, j \in J : f_i = f_j \tag{4.17b}$$

$$Q_{j^*} = 0 \tag{4.18a}$$
$$Q_j \geq Q_i + q_j - (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.18b}$$

$$Q_j \leq r^* + rC_j \text{ for all } j \in J \tag{4.19}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j \in J \tag{4.20a}$$
$$y_{ij} \in \{0, 1\} \text{ for all } i, j \in J \tag{4.20b}$$

The objective (4.13) is to minimize the sum of tardiness of all jobs in the production schedule. Constraints (4.14a), (4.14b) and (4.14c) require each job to be fulfilled exactly once. Constraints (4.15a) and (4.15b) require that the completion time of a job must be at least as large as the the completion time of the preceding job plus the setup time, if a setup is conducted, and the processing time. Furthermore, constraint (4.15c) ensures that job j is processed immediately after completing a preceding job i if no setup is conducted. Constraints (4.16a) and (4.16b) restrict the tardiness of each job. Constraint (4.17a) ensures that a setup is conducted if jobs of different families are processed after another, whereas constraint (4.17b) allows a setup to be conducted if jobs of the same family are processed after another. Constraints (4.18a) and (4.18b) restrict the cumulative resource requirements before any job. The cumulative resource requirements must not exceed the amount available until completion of the job as required by constraint (4.19). Finally, constraint (4.20a) and constraint (4.20b) give the domain of the additional binary decision variables.

4.4 Model 4 - Single Machine Family Scheduling with Upper and Lower Resource Constraint and Setup Constraint

Within the previous models, the constant supply of hot metal was taken into consideration in terms of minimum hot metal required for producing a charge. There, setup processes are introduced into the sequence, in order to increase the amount of available hot metal until it meets demand. In case a large number of setup processes is scheduled, because the sequence of charges consists of a large number of charge family changes, a corresponding large amount of hot metal is supplied. The ability to store hot metal is

limited as described in Section 2.3.5. Therefore, a schedule must also assure that enough hot metal is consumed.

Translated into the scheduling problem, the accumulated amount of resource requirements, at any point within a given sequence, needs to be at least the accumulated resource supplied subtracted by the maximum buffer capacity to store the resource.

4.4.1 Notation

Again, let J denote a set of jobs to be processed by a single machine. Each job is characterised by a due date d_j, a processing time p_j, a quantity of resource requirement q_j, a job family f_j, a completion time C_j and a tardiness T_j. The resource is supplied at rate r and initial amount $r^\star$ of the resource is available. A setup time s_{ij} is defined for each pair of $i, j \in J$, with $s_{ij} > 0$ for $f_i = f_j$. Binary variable x_{ij} indicating whether job $j \in J$ is scheduled immediately after job i ($x_{ij} = 1$) or not ($x_{ij} = 0$) and binary variable y_{ij} indicates whether a setup is conducted between jobs i and j, or not.

Within Model 3, a common resource that is consumed by each job was introduced. This extension is forcing the schedule to execute setups or introduce waiting times whenever the supply of the common resource is not sufficient for production. Now let us consider a schedule that accumulates a large number of setups or waiting times. This schedule results in a situation where more resources are supplied then consumed by the jobs. The excess material needs to be stored until it is consumed. Since the available capacity for buffering surplus inventory is limited in practical applications, the generated sequence could need to locally reduce the use of setups and waiting times to avoid infeasible schedules.

The problem formulated in Model 3 needs to be extended by the constant parameters B, indicating the available buffer inventory capacity. Figure 4.4 illustrates a situation, where a large number of setups leads to a buffer overflow.

While insufficient supply could always be compensated by introducing additional waiting times in Model 3, the a maximum buffer level extended in Model 4 could lead to infeasible solutions. This makes the problem studied in Model 4 fundamentally different. In Model 3 there is no interdependency between different setup processes and as long as enough resource is supplied, necessary setups are scheduled towards the end of the sequence to minimize the sum of completion times and the resulting total tardiness. In Model 4, the accumulation of setups towards the end of the sequence could lead to infeasibility. Therefore, an optimal sequence could require an equal distribution of setup processes that leads to a larger sum completion times and a higher total tardiness. Figure 4.5 depicts the described situation. The left part shows an infeasible schedule where setups

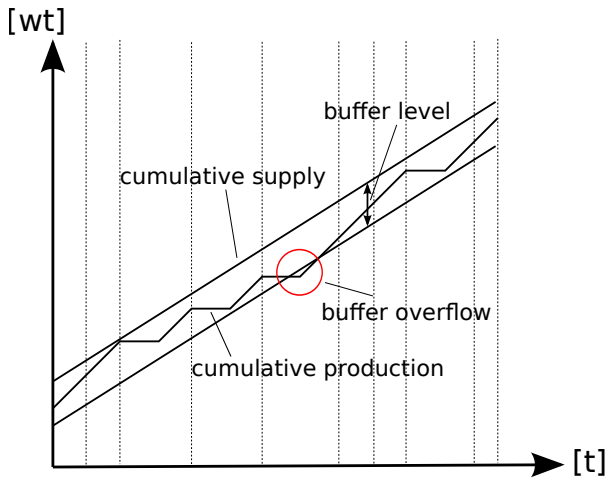

Figure 4.4: Example on how a large number of setups leads to an infeasible solution.

are accumulated at the end of the sequence. The right part shows a feasible schedule that shifted part of the setups towards the beginning of the sequence.

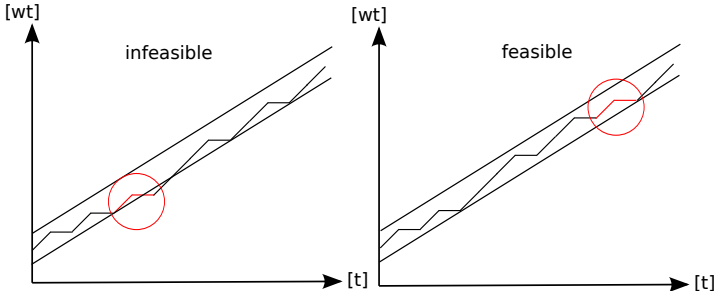

Figure 4.5: Example on how a limited inventory capacity increases the total tardiness of a sequence.

4.4.2 MILP Formulation

The described extension requires a modification of the MILP formulation in Model 4. An additional constraint needs to be added to the cumulative resource requirement, forcing the schedule to consume enough resource such that the buffer is not exceeded.

Model 4 can be described by the following MILP:

$$\text{minimize} \sum_{j \in J} T_j \tag{4.21}$$

subject to

$$\sum_{i \in J \setminus \{j\}} x_{ij} = 1 \text{ for all } j \in J \tag{4.22a}$$
$$\sum_{j \in J \setminus \{i\}} x_{ij} = 1 \text{ for all } i \in J \tag{4.22b}$$
$$x_{jj} = 0 \text{ for all } j \in J \tag{4.22c}$$

$$C_{j^*} = 0 \tag{4.23a}$$
$$C_j \geq C_i + y_{ij} s_{ij} + p_j - (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.23b}$$
$$C_j \leq C_i + p_j + (1 - x_{ij} + y_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\}. \tag{4.23c}$$

$$T_j \geq 0 \tag{4.24a}$$
$$T_j \geq C_j - d_j \text{ for all } j \in J \tag{4.24b}$$

$$y_{ij} = x_{ij} \text{ for all } i, j \in J : f_i \neq f_j \tag{4.25a}$$
$$y_{ij} \leq x_{ij} \text{ for all } i, j \in J : f_i = f_j \tag{4.25b}$$

$$Q_{j^*} = 0 \tag{4.26a}$$
$$Q_j \geq Q_i + q_j - (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.26b}$$
$$Q_j \leq Q_i + q_j + (1 - x_{ij})M \text{ for all } i \in J, j \in J \setminus \{j^*\} \tag{4.26c}$$

$$Q_j \leq r^* + rC_j \text{ for all } j \in J \tag{4.27a}$$
$$Q_j \geq r^* + r(C_j) - B \text{ for all } j \in J \tag{4.27b}$$
$$Q_i \geq r^* + r(C_j - p_j) - B - (1 - x_{ij})M \text{ for all } i, j \in J \tag{4.27c}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j \in J \tag{4.28a}$$
$$y_{ij} \in \{0, 1\} \text{ for all } i, j \in J \tag{4.28b}$$

The objective (4.21) is to minimize the sum of tardiness of all jobs in the production schedule. Constraints (4.22a), (4.22b) and (4.22c) require each job to be fulfilled exactly once. Constraints (4.23a) and (4.23b) require that the completion time of a job must be at least as large as the the completion time of the preceding job plus the setup time, if a setup is conducted, and the processing time. Furthermore, constraint (4.23c) ensures that job j is processed immediately after completing a preceding job i if no setup is conducted. Constraints (4.24a) and (4.24b) restrict the tardiness of each job. Constraint (4.25a) ensures that a setup is conducted if jobs of different families are processed after another, whereas constraint (4.25b) allows a setup to be conducted if jobs of the same family are processed after another. Constraints (4.26a) and (4.26b) constraint the cumulative resource requirements before any job and (4.26c) constraints the increase in cumulative resource to be exactly the resource requirement of job j. The cumulative resource requirements must not exceed the amount available until completion of the job as required by constraint (4.27a). On the other side, enough resource needs to be consumed not to exceed the buffer limit, both after a potential setup process (4.27b) and after the production of any job (4.27c). Finally, constraint (4.28a) and constraint (4.28b) give the domain of the additional binary decision variables.

5 Solution Approach

The problem studied in this thesis combines family scheduling (see Section 3.3.2) with resource constraint scheduling (see Section 3.3.3) and scheduling to minimize total tardiness (see Section 3.3.1). All three problems are known to be NP-hard (e.g Cheng et al. (2003), Briskorn et al. (2010), Du and Leung (1990)). Therefore, solving the problem using a MILP-solver is in general too time consuming for problem instances with larger numbers of jobs.

The approach of this thesis is motivated from the literature on heuristics to solve scheduling problems that share similar characteristics (family scheduling, resource constrained scheduling and/or scheduling to minimize total tardiness). As described in Section 3.3, good results have been reported by using simple operators to modify the sequence of jobs in combination with a meta-heuristic framework to escape local optima. The approach in this thesis consists of an iterated local search approach as used e.g. in Subramanian et al. (2014). The general framework of the heuristic is presented in Section 5.1. An important part of the heuristic is the evaluation of a given sequence. Section 5.2 describes how total tardiness can be evaluated for the different models presented in Section 4 and how feasibility considerations are respected. The different local search operators selected for this thesis are presented in Section 5.3.In order to escape local optima, perturbation has been found to be a good approach (Lourenco et al. 2003). Section 5.4 describes the details on how sequences are perturbated in this thesis.

5.1 Heuristic Framework

The pseudocode of the heuristic used within this PhD thesis is shown in Figure 5.1.

The approach begins by generating an initial solution which can be any sequence of jobs. The procedure to generate initial solutions for the different models is presented in Section 5.5. It then initialises an iteration counter and repeats the same steps until the iteration counter has reached a given limit. In each iteration the approach generates a random sequence of operators, which are subsequently applied. For each operator, all possible moves are examined. If an improving solution is found, this solution is accepted as the new incumbent solution. If within an iteration run, no improvement can be obtained

```
1: set s := initialsolution()
2: set i := 1;
3: while i ≤ k do
4:   repeat
5:     select neighbourhood operator Ω
6:     for all possible applications of Ω
7:       if tardiness(Ω(s)) < tardiness(s) then
8:           set s := Ω(s)
9:       end if
10:    next
11:  until s is locally optimal for all operators
12:  set s := perturbation(s)
13:  set i := i + 1;
14: end while;
```

Figure 5.1: Heuristic

by any of the operators, a locally optimal solution is found. Following the iterated local search framework of Lourenco et al. (2003), the search process is repeated in order to be able to escape from local optima of poor quality. Before continuing with the next iteration, the incumbent solution is perturbed in order to obtain a new solution that potentially can be improved using the operators.

The solution approach comprises three main components: the evaluation of tardiness of a given sequence of jobs, different neighbourhood operators to modify sequences, and a method to perturb a solution.

5.2 Tardiness and feasibility evaluation

To evaluate the tardiness of a particular sequence, the approach has to compute completion times and tardiness values for each job in the sequence. The tardiness evaluation considers waiting times that may be required due to limited resource availability as described below. Depending on the model under consideration, an evaluation of the feasibility of the given sequence is also taken into account.

5.2.1 Model 1

For the simple single machine family scheduling problem, completion times and tardiness for each job in a sequence can be calculated very easy.

The first job in the sequence has completion time

$$C_j = p_j.$$

For all other jobs j the completion time is

$$C_j = C_i + s_{ij} + p_j,$$

where job i is the predecessor of j. The tardiness of any job j is

$$T_j = \max\{0, C_j - d_j\}.$$

5.2.2 Model 2

In the case of the single machine family scheduling problem with upper resource constraints, where the machine may be idle for any period of time after completion of one job and before starting to process the next job, optimal completion times for a given sequence of jobs can still be computed easily. For any job j, the amount of resources required Q_j is the cumulative quantity of resource required by all jobs in the sequence up to job j. The completion times of the jobs in the sequence can be computed as follows. The first job in the sequence has completion time

$$C_j = \max\{p_j, \frac{Q_j - r^*}{r}\}.$$

and cumulative quantity

$$Q_j = r_j$$

For all other jobs j the completion time is

$$C_j = \max\{C_i + s_{ij} + p_j, \frac{Q_j - r^*}{r}\},$$

the cumulative quantity is

$$Q_j = Q_i + r_j,$$

where job i is the predecessor of j. The tardiness of any job j is

$$T_j = \max\{0, C_j - d_j\}.$$

5.2.3 Model 3

Now let us consider the case of the single machine family scheduling model with upper resource constraints and setup constraints, where an additional setup between jobs of

the same family is required if the job does not start immediately after completion of the preceding job and no waiting times are allowed. If the values y_{ij} indicating whether there is a setup between i and j are known, completion time and tardiness can be scheduled analogously to the method described above. The solution approach used in this thesis, however, is only based on operators changing the sequence of jobs and the decision whether a setup is taken or not is not explicitly taken by the operators. Instead, this decision is taken when calculating the completion times of the jobs in the sequence. The cumulative resource requirement Q_j is calculated as in Section 5.2.2. The first job in the sequence is tentatively given the completion time

$$C_j = \max\{p_j, \frac{Q_j - r^*}{r}\}.$$

This value may have to be increased if a subsequent job without intermediate setup cannot be processed due to resource constraints.

For any other job j, it has to be distinguished between several cases. If $f_i \neq f_j$, where job i is the predecessor of j, then a setup is required and the completion time job j is tentatively set to

$$C_j = \max\{C_i + s_{ij} + p_j, \frac{Q_j - r^*}{r}\}.$$

If $f_i = f_j$ then two alternatives must be considered. In the first alternative an additional setup is included and the completion time of job j is tentatively set to

$$C_j = \max\{C_i + s_{ij} + p_j, \frac{Q_j - r^*}{r}\}.$$

In the second alternative, no setup is made between i and j and the completion time of job j is tentatively set to

$$C_j = \max\{C_i + p_j, \frac{Q_j - r^*}{r}\}.$$

To eliminate possible idle time between jobs, the completion time of all jobs prior to j which are not separated by a setup is increased by

$$\Delta = \max\{0, \frac{Q_j - r^*}{r} - (C_i + p_j)\}.$$

As above, the tardiness of any job j is

$$T_j = \max\{0, C_j - d_j\}.$$

The approach can efficiently be implemented as follows. For any job j in a given sequence of jobs let $l = (l^{\text{time}}, l^{\text{tardiness}}, l^{\text{lastsetup}})$ be a label where l^{time} denotes the completion time of job j, $l^{\text{tardiness}}$ denotes the cumulative tardiness until completion of job j and $l^{\text{lastsetup}}$

denotes the sequence position of the last setup that was executed. The approach begins swith a label $l = (0, 0, 0)$. Then it iterates through the given sequence of jobs and calculates completion times and tardiness values as described above. In case a setup occurs, the sequence position of the last setup is updated. The information on the sequence position of the last setup is used when eliminating idle times, as described above. For each different alternative a new label is generated. A tree of alternatives is generated by extending each alternative label.

To reduce the number of alternative labels to be considered the following dominance criteria are used to reduce the number of labels.

Proposition 1. *Let l_1 and l_2 denote labels associated to schedules up to the same point i in a sequence of jobs and let j denote the next job in the sequence. Label l_1 dominates label l_2 if*

$$l_1^{\text{time}} + s_{ij} \leq l_2^{\text{time}}$$

and

$$l_1^{\text{tardiness}} \leq l_2^{\text{tardiness}}.$$

Proof: Let $\hat{l}_1$ denote the label associated to the schedule obtained by adding job j to the schedule associated to l_1 at the earliest time after a setup of duration s_{ij} and let $\hat{l}_2$ denote the label associated to any schedule obtained by extending the schedule associated to l_2 by job j. We have $\hat{l}_1^{\text{time}} \leq \hat{l}_2^{\text{time}}$ and $\hat{l}_1^{\text{tardiness}} \leq \hat{l}_2^{\text{tardiness}}$. Furthermore, for any extension of $\hat{l}_1$ and $\hat{l}_2$ adding setups at the same positions, every job in the schedule associated to the former will be completed earlier or at the same time compared to the same job in the schedule associated to the latter. Thus, the total tardiness of the former will be smaller or equal to the latter and l_1 dominates l_2. □

By eliminating all dominated labels, the size of the search tree build to evaluate total tardiness is reduced effectively.

It must be noted that total tardiness can similarly be evaluated for non-constant supply and demand rates. By replacing all occurrences of the term $\frac{Q_j - r^*}{r}$ with a function that calculates the earliest point in time when the job j can be completed subject to a sufficient resource availability, arbitrary supply and demand patterns can be considered.

5.2.4 Model 4

For the single machine family scheduling model with upper and lower resource constraints and setup constraints, a lower limit on the cumulative quantity constraint is introduced. Consequently, a sequence could possibly be infeasible as described in Section 4.4.

There are two possible cases that could lead to infeasibility. First, when a setup process is required. In this case, additional resource $s_{ij}r$ is supplied during the setup process. Since no consumption takes places during setups, the buffer before starting the setup process, given as $Q_i - (C_i r + r^\star - B)$, needs to greater than or equal to the additionally supplied resource:

$$Q_i - (C_i r + r^\star - B) \geq s_{ij} r.$$

The second case occurs whenever the consumption rate given as q_j/p_j is smaller than the supply rate r. In this case, the additional resource supply during the production process is the difference of supply $p_j r$ and consumption q_j and again the buffer needs to be able to assimilate it:

$$Q_i - (C_i r + r^\star - B) \geq (p_j r - q_j).$$

Figure 5.2 illustrates the two cases.

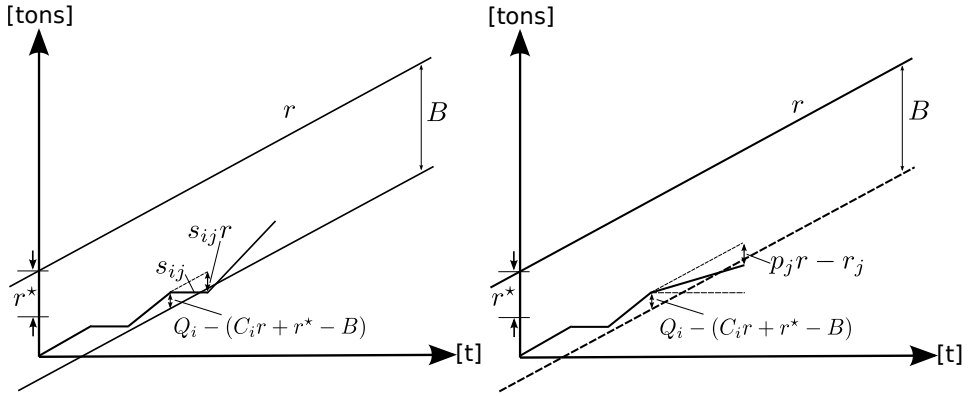

Figure 5.2: Two cases that could lead to infeasible sequences in Model 4.

The feasibility check is implemented as part of the labeling approach described in Section 5.2.3. Whenever a setup occurs ($f_i \neq f_j$) or the consumption rate is less then the supply rate, feasibility is evaluated using the equations above. In case the sequence is infeasible, the tardiness calculation terminates and the move that triggered the evaluation is rejected.

5.3 Local Search Operators

This section describes the set of operators used by the local search approach proposed in this thesis. It consists of job move (Section 5.3.1), job exchange (Section 5.3.2), batch move (Section 5.3.3), batch exchange (Section 5.3.4), batch combine (Section 5.3.5) and batch break (Section 5.3.6). The first two operators are directly based on the sequence of jobs, whereas the other operators are based on batches, i.e. subsequences without a setup. The advantage of such batch-based operators is that some structural properties of the current solution are maintained and unnecessary setups can be avoided.

5.3.1 Job Move

The "Job Move" operator selects a job and inserts it at another position in the sequence. Figure 5.3 illustrates all possible operator moves for a given job. After selecting a job, the operator iterates through the sequence of jobs and evaluates the insertion of the job at all possible positions. If total tardiness can be reduced, the job is moved to the position leading to the lowest total tardiness. This operator is a generalization of the *Move* operator used by Schaller (2007), who limited the search to only move jobs within batches of the same family. By also moving jobs to other positions, sequences are generated that are not covered by the other operators.

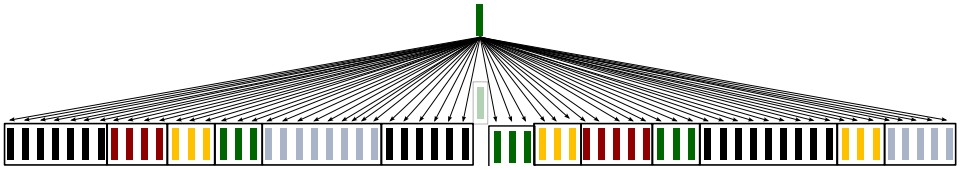

Figure 5.3: Example of all possible job move operations for a selected job.

5.3.2 Job Exchange

The "Job Exchange" operator selects a job and and exchanges its position with another job in the sequence. Figure 5.4 illustrates a single operator move. After selecting a job, the operator iterates through the sequence of jobs and evaluates the exchange of the job with any other job in the sequence. If total tardiness can be reduced, the positions of the two jobs leading to the lowest total tardiness reduction are exchanged. This operator is equivalent to the *JI procedure* proposed by Schaller (2007).

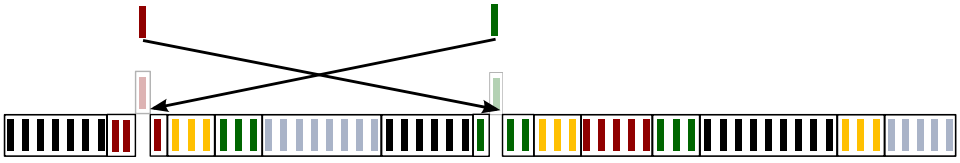

Figure 5.4: Example of exchanging two jobs. In this example the two jobs belong to different setup families and are located in the middle of a batch. The job exchange in the example therefore results in two additional setup processes.

5.3.3 Batch Move

The "Batch Move" operator selects a batch and inserts it at another position in the sequence. Figure 5.5 illustrates the operator move. After selecting a batch, the operator

iterates through the sequence of batches and evaluates the insertion of the complete batch at any sequence position. If a reduction of the total tardiness can be achieved, the batch is moved to the position leading to the lowest total tardiness.

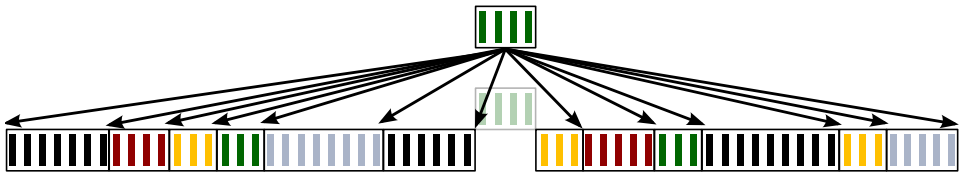

Figure 5.5: Example of moving a selected batch to all possible sequence positions.

5.3.4 Batch Exchange

The "Batch Exchange" operator selects a batch and and exchanges its position with another batch in the sequence of batches. Figure 5.6 illustrates a single operator move. After selecting a batch, the operator iterates through the sequence of batches and evaluates the exchange of the positions of the complete batches. If total tardiness can be reduced, the positions of the two batches leading to the lowest total tardiness reduction are exchanged. A similar operator (SW), that only considers neighbouring batches or randomly selected batches was proposed by Gupta and Chantaravarapan (2008). The operator selected for this thesis is equivalent to the *BI procedure* proposed by Schaller (2007), which can be considered a generalization of the previous operators.

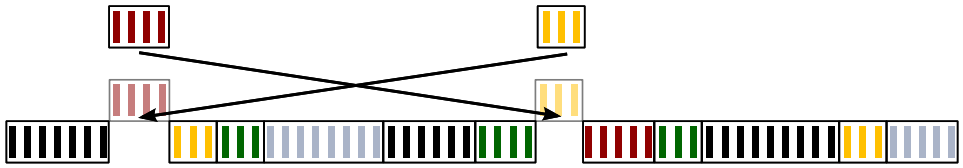

Figure 5.6: Example of exchanging two batches. In this example, the exchange of the two batches result in the reduction of two setup processes, since the moved batches are inserted right before batches from the same setup family.

5.3.5 Batch Combine

The "Batch Combine" operator selects a batch and combines the batch with the next batch of the same family. Figure 5.7 illustrates a single operator move. It removes the jobs of both batches and iterates through the sequence to evaluate whether reinsertion of the combination of both batches can reduce total tardiness. If total tardiness can be reduced, the combined batch is inserted at the position leading to the lowest total tardiness. Baker (1999) proposed an operator that combines two batches from the same

family and sorts the new sequence in EDD order of batches. The operator selected for this thesis is equivalent to the *CONSOL procedure* proposed by Schaller (2007), which can be considered a generalization of the previous operators.

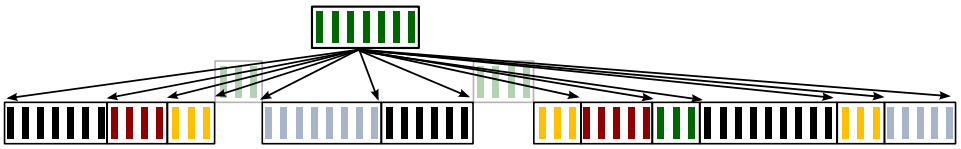

Figure 5.7: Example of combining two batches from setup family "green" and evaluating the insertion at any sequence position in between the original sequence position of both batches

5.3.6 Batch Break

The “Batch Break” operator selects a batch, breaks the batch into two parts and inserts both parts at a new position in the sequence. Figure 5.8 illustrates a single operator move. After selecting a batch, the operator sorts all jobs within the batch according to their due date and divides the batch at the position of the largest due date difference. The operator then iterates over all possible sequence positions for both parts and evaluates the total tardiness. If total tardiness can be reduced, the parts are inserted at the positions leading to the lowest total tardiness.

Other then the *Break* operator in Schaller (2007), all sequence positions of both parts of the broken batch are examined. This is motivated by possible inhomogeneous due dates within a batch, which broken sub parts could be better placed further up and down the sequence. The *S-Neighbourhood* operator of Baker (1999) first tries to split the last job of a batch and move it to a subsequent batch of equal setup family (similar to *Move* of Schaller (2007)). In case this does not lead to improvements, generating an additional batch (breaking the former batch) is considered, too.

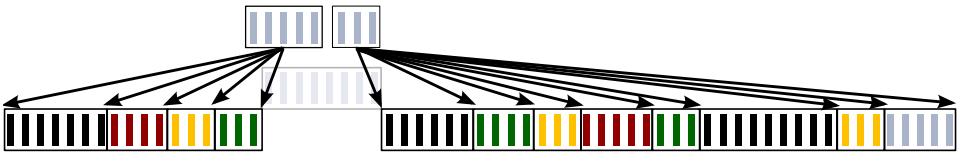

Figure 5.8: Example of breaking a "blue" batch and inserting each broken part at different sequence positions.

5.4 Perturbation Strategy

A well known problem of local search techniques is to get stuck in local optima. Although the described structure of using different neighbourhood types already contributes to approach the global optimum, large travel in search space is not present. An easy but powerful approach to break out of local optima is called iterated local search (Lourenco et al. 2003). In this procedure, local search is used to arrive at a local optimum. Afterwards, perturbation is used to modify the locally optimal sequence in order to reach a different area of the solution space. After perturbation, local search is used again to find the best local solution in the new solution space area. These two steps (perturbation and local search) are repeated until a termination condition is met.

The approach for perturbation selected for this thesis is motivated by Lü and Hao (2009). Their critical element guided perturbation (CEGP) strategy investigates a current best solution and calculates a scoring variable for each job. Based on the scores, which are derived from problem characteristics, a new initial sequence is derived.

The idea of the perturbation approach used in this thesis is to evaluate the most promising sequence position in terms of tardiness minimization for each job in a given current best sequence σ. First, the lateness is calculated as

$$L_j^\sigma = C_j^\sigma - d_j$$

for each job in the current best sequence. A positive lateness indicates that this job should ideally be shifted towards the beginning of the sequence, whereas a negative lateness indicates a backwards shift. In order to estimate a promising shift in sequence positions, each lateness is divided by the average job processing time

$$\Delta_j^{pos} = \frac{L_j^\sigma}{\sum p_j}.$$

Finally, the calculated shift is subtracted from the original sequence position in the current best sequence σ to get a value for the most promising sequence position

$$k_j^\star = k_j^\sigma - \Delta_j^{pos}.$$

The new initial solution sequence σ' after perturbation is a sequence of jobs in order of most promising sequence position $k^\star$. The former position in the current best sequence k_j^σ serves as a tie breaker.

5.5 Experiment Design and Test Instances

This section presents the setup of computational experiments conducted to evaluate the proposed approach for all model variants of the problem. The approach is evaluated using artificially generated instances motivated from the real-world scheduling problem of the case study (see Chapter 6). The generated instances share common characteristics of the real-life problem, however, for confidentiality reasons, these characteristics cannot be described in detail.

Unfortunately, there are no public problems available for family scheduling with or without consideration of additional constraints. In order to enable a comparison of alternative approaches, as possible for other scheduling problems (e.g. single machine scheduling to minimize total tardiness: Section 3.3.1), the test instances used in this thesis can be downloaded under: http://www.telematique.eu/research/download.

The txt files contains the following information:

- NUMBER_OF_JOBS
- NUMBER_OF_FAMILIE
- DUEDATES
- PROD_WEIGHT
- SETUPFAMILY
- SETUPMATRIX
- SETUPMATRIX_F
- INIT_BUFFER
- MAX_BUFFER
- SupplyRate
- START

The following subsections present a detailed description on how the test instances have been generated for the different model variants (Sections 5.5.1 - 5.5.3). Afterwards, the setup for the experiments is explained in Section 5.5.4.

5.5.1 Model 1

For each instance, a set of jobs J and a set of families F are generated. Each family is randomly assigned one job and the remaining jobs are distributed across the families according to an exponential distribution as illustrated in Figure 5.9.

In the real-life problem, processing times of a job depends on its setup family and the production width. Therefore, for each setup family $f \in F$ processing times p_f are randomly generated in the range $[2400; 3000]$. Then, for each job j in family f, the processing time p_j is randomly generated in the range $[\max(0.9p_f, 2400); \min(1.1p_f, 3000)]$. This way, processing times are similar for jobs of equal setup family, but still can vary as a result of the production width.

For any pair of families, setup times between jobs of these families are randomly selected

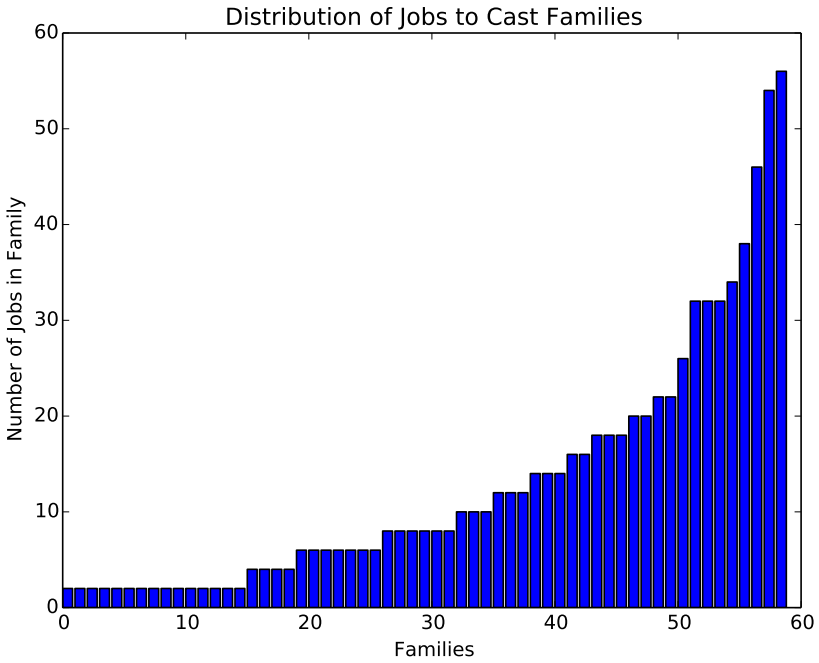

Figure 5.9: Distribution of jobs per setup family

to be either 900 or 2700. This is also motivated from real world instances, where a reduced setup time is possible in case certain characteristics are given.

The due date d_j of any job j is set as follows. First, orders are sequenced under GTA, that is all orders belonging to one family are grouped together in a batch and batches are sequenced in order of descending size. The minimum total time t_1 required to produce all jobs can then be calculated for this sequence. The due date for each job is randomly selected from range $[-0.25t_1; 1.25t_1]$, however, any negative value is set to zero. Thus, as in real-life, instances contain some jobs which will be tardy for any position in the sequence.

5.5.2 Model 2 and Model 3

In Model 2 and 3, an upper bound on the cumulative resource consumption is introduced as described in Section 4.2. The test instances used for Model 2 and Model 3, share the same requirements. Motivated from the real problem instances, the resource demand q_j of job $j \in J$ is set to a random number in range $[250, 270]$.

The due date of jobs are set as for Model 1.

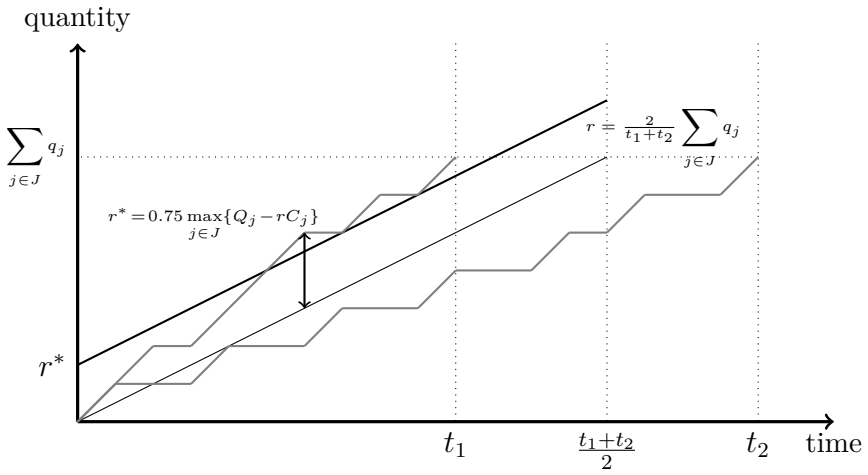

Figure 5.10: Supply rate and initial inventory are derived from two reference sequences.

In order to generate instances, in which the resource constraint is not satisfied trivially and does not always require waiting times, the supply rate r and the initial inventory r^* are determined as follows. Another sequence is generated, in which jobs are ordered according to their due dates, with the earliest due date first. The minimum total time t_2 required without resource constraints is calculated for this sequence. For all instances $t_1 < t_2$ is given because more setups are required. The supply rate r is chosen as $r = \frac{2}{t_1+t_2}\sum_{j\in J} q_j$. As shown in Figure 5.10, the resource supply (without initial inventory) grows at a rate that is between the average demand rates of the two sequences generated. The initial inventory is now set to 75% of the largest difference between the cumulative demand of the first sequence and the supply curve without initial inventory. As a result, the first sequence, which minimises the total number of setups, would be infeasible without additional waiting times. When searching for a solution with low total tardiness, a good tradeoff between minimising the number of setups and minimising waiting times thus has to be found.

5.5.3 Model 4

The extension of Model 4 consists of a lower bound on the cumulative resource consumption (see Section 4.4). Within instance generation, the parameter B, describing the maximum buffer capacity, needs to be defined. The task is to define B such that not all sequences fulfil the requirements and at the same time, some sequences do.

A similar approach to the one used for the upper constraint limit is used (see Section 5.5.2). This time, the largest difference between the EDD sequence σ^{EDD} and the supply curve is estimated. Afterwards, 75% of this largest difference plus the initial inventory

level is defined to be the buffer capacity:

$$B = 0.75 \max_{j \in \sigma^{EDD}} rC_j - Q_j$$

The above described procedure is motivated by excluding a certain share of sequences by cutting away σ^{EDD} and similar sequences. Besides that, the buffer level B must be sufficiently large to contain enough resource to produce any job in order to generate feasible instances. Therefore, the maximum buffer is set to be the maximum of both conditions:

$$B = \max \left\{ 0.75 \max_{j \in \sigma^{EDD}} rC_j - Q_j, \max_{j \in J} \{r_j - p_j r\} \right\}$$

Figure 5.11 depicts both conditions on an example test instance.

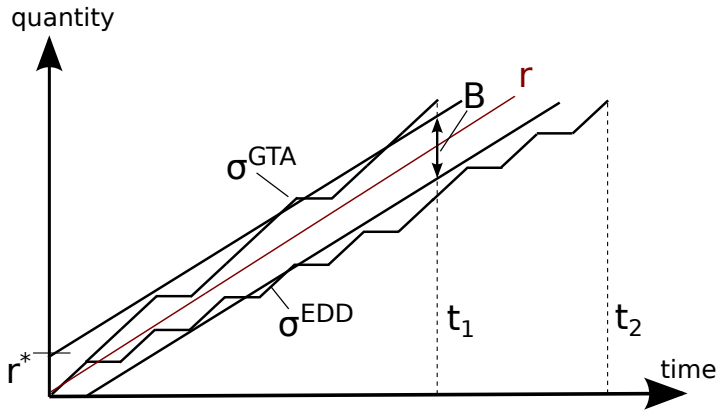

Figure 5.11: Example on how the buffer capacity B is generated.

Especially for very small test instances (e.g. 8 Jobs from 2 setup families), the differences between σ^{GTA} and σ^{EDD} is very small. In this case, the above procedure could not be sufficient to guarantee feasible test instances. Therefore, a feasibility check is executed within instance generation. Whenever the σ^{GTA} cannot be produced with the given buffer capacity, the buffer capacity is extended by the required amount.

5.5.4 Experiment Design

In order to evaluate the developed heuristic for the different model alternatives, test instances have been generated using the described procedure.

For small test instances, the number of families n_f were selected from set {2,3,4} and the number of jobs n_j from set {8,10,12,15}. For each combination of n_f and n_j, five instances were randomly generated. Additionally, medium sized instances were generated with n_f from set {4,5,6} and n_j from set {20,30,40,50}.

Two different methods are compared to each other: an exact solution approach and the developed heuristic. First, the described MILP model (see Section 4) was solved using the commercial solver CPLEX with version 12.6.0, 3600 s CPU calculation time limit, 2 GB RAM, 10 GB tree limit and 1 thread on an Intel Xeon(R) CPU W3530 @ 2.80GHz X 4 with UBUNTU 14.04 64-bit operating system. The second method tested, is the iterated local search heuristic described in Section 5. In order to establish a fair comparison, each method is started from an equal initial feasible solution. For models 1-3, this solution is generated by ordering jobs in order of earliest due date and inserting waiting time according to the tardiness evaluation scheme presented in Section 6.2.1. For model 4 the initial feasible solution is generated by sorting jobs according to their families, generate batches for each jobs and sorting the batches in decreasing batch size.

5.6 Results

This section describes the results obtained by running the experiments described in Section 5.5. For all models, five experiments have been conducted for each number of jobs from set {8,10,12,15,20,30,40,50}. Further, the number of families has been varied from set {2,3,4} for instances with up to 15 jobs and from set {4,5,6} for larger problem instances. This results in a total of 15 instances for each problem size. In the following Sections, the average results from all instances are presented for each problem size. Detailed results can be found in the Appendix.

In the result overview tables, provided for each model variant in the following Sections, the first column indicates the number of jobs. The second column gives the average computation time (in seconds) required by the MIP solver. The third column gives the average degree to which the gap between solution value of the initial solution to the best lower bound has been closed (GAP). This value is calculated as follows:

$$GAP = \frac{TT_{init} - UB_{method}}{TT_{init} - LB_{MIP}} \tag{5.1}$$

A value of 100 indicates that the gap is closed to 100 per cent, i.e. that the solution is optimal. The next two columns give the same information for the heuristic. The last

column indicates the ratio between the GAP value for the heuristic divided by the GAP value for the MIP. A ratio larger than 1 indicates that the heuristic outperforms the MIP.

5.6.1 Model 1

This Section discusses the results derived by solving the test instances using model 1. Table 5.1 presents an overview with average values for all 15 test instances for each problem size. Detailed results for all 120 test instances solved using model 1 can be found in the Appendix.

	MIP		Heuristic		
Jobs	**CPU**	**GAP**	**CPU**	**GAP**	**Ratio**
8	5.59	100.00	0.01	100.00	1.00
10	373.39	100.00	0.02	100.00	1.00
12	2096.72	81.18	0.04	80.51	0.99
15	3360.45	55.97	0.07	55.96	1.00
20	3360.09	52.92	0.19	52.81	1.00
30	3600.00	56.21	0.72	57.59	1.02
40	3600.00	46.21	1.75	59.44	1.29
50	3600.00	42.41	4.01	66.59	1.57

Table 5.1: Average results for Model 1

The results in Table 5.1 show that CPLEX is able to provide optimal solutions (the GAP presented in column 2 is 100 % closed) for all problem instances with up to 10 jobs, given the calculation time limit of 3600 s. With increasing problem sizes, the ability of CPLEX to close the gap between the initial solution value and the best lower bound obtained is strongly diminished.

Looking at the ratio between gap closing ability of CPLEX and the heuristic, it can be observed that on average CPLEX and the heuristic are able to derive similar results for problem instances with up to 30 jobs. However, the average calculation time required by CPLEX is 2132 s compared to only 0.18 s used by the heuristic.

For problem instances with 40 jobs and more, the heuristic is able to outperform CPLEX, still requiring only 2.88 s calculation time.

In the following, the statistics of the detailed results presented in the Appendix are displayed.

For Model1, CPLEX was able to solve 40 out of 120 problem instances with up to 20

jobs. The developed heuristic was able to solve 39 or 97.5 % of those instances as well. However, the heuristic only required an average of 0.028 s while CPLEX was running 299 s on average to solve the instances.

For 17 out of 120 problem instances, the upper bound of CPLEX was equal to the solution obtained by the heuristic. This can be observed for problem instances with up to 15 jobs. In this case, it is likely that the upper bound of CPLEX is the optimal solution, but optimality could not be proven within the 1 h time limit. The average calculation time required by the heuristic is 0.07 s compared to 3600 s run time of CPLEX.

In 52 out of 120 problem instances, the heuristic outperformed CPLEX within the given time limit. The average calculation time required by the heuristic for those rather large instances was 1.86 s, which is still significantly smaller then the CPLEX time limit of 3600 s.

5.6.2 Model 2

This Section discusses the results derived by solving the test instances using model 2. Table 5.2 presents an overview with average values for all 15 test instances for each problem size. Detailed results for all 120 test instances solved using model 2 can be found in the Appendix.

	MIP		**Heuristic**		
Jobs	**CPU**	**GAP**	**CPU**	**GAP**	**Ratio**
8	8.72	100.00	0.02	99.97	1.00
10	624.70	100.00	0.03	100.00	1.00
12	2387.21	76.38	0.06	76.07	1.00
15	3364.40	55.65	0.11	55.62	1.00
20	3361.35	51.84	0.32	53.26	1.03
30	3600.00	48.44	1.18	58.17	1.20
40	3600.00	35.02	2.95	59.79	1.71
50	3600.00	32.57	6.80	67.27	2.07

Table 5.2: Average results for Model 2

The results in Table 5.2 show that CPLEX is able to provide optimal solutions for problem instances with up to 10 jobs, given the calculation time limit of 3600 s. Again, with increasing problem sizes, the ability of CPLEX to close the gap between the initial solution value and the best lower bound obtained is strongly diminished.

Looking at the ratio between gap closing ability of CPLEX and the heuristic, it can be observed that on average CPLEX and the heuristic are able to derive similar results for

problem instances with up to 20 jobs. However, the average calculation time required by CPLEX is 1949 s compared to only 0.11 s used by the heuristic.

For problem instances with 30 jobs and more, the heuristic is able to outperform CPLEX, still requiring only 3.64 s calculation time on average.

In the following, the statistics of the detailed results presented in the Appendix are displayed.

For Model2, CPLEX was able to solve 39 out of 120 problem instances with up to 20 jobs. The developed heuristic was able to solve 35 or 89.7 % of those instances as well. However, the heuristic only required an average of 0.04 s while CPLEX was running 425.5 s to solve the instances.

For 21 out of 120 problem instances, the upper bound of CPLEX was equal to the solution obtained by the heuristic. This can be observed for problem instances with up to 15 jobs. In this case, it is likely that the upper bound of CPLEX is the optimal solution, but optimality could not be proven within the 1 h time limit. The average calculation time required by the heuristic is 0.12 s compared to 3600 s run time of CPLEX.

In 55 out of 120 problem instances, the heuristic outperformed CPLEX within the given time limit. The average calculation time required by the heuristic for those rather large instances was 3.04 s, which is still significantly smaller then the CPLEX time limit of 3600 s.

5.6.3 Model 3

This Section discusses the results derived by solving the test instances using model 3. Table 5.3 presents an overview with average values for all 15 test instances for each problem size. Detailed results for all 120 test instances solved using model 3 can be found in the Appendix.

The results in Table 5.3 show that CPLEX is able to provide optimal solutions for problem instances with up to 10 jobs, given the calculation time limit of 3600 s. With increasing problem sizes, the ability of CPLEX to close the gap between the initial solution value and the best lower bound obtained is strongly diminished.

Looking at the ratio between gap closing ability of CPLEX and the heuristic, it can be observed that on average CPLEX and the heuristic are able to derive similar results for problem instances with up to 20 jobs. However, the average calculation time required by CPLEX is 2051 s compared to only 1.92 s used by the heuristic.

	MIP		Heuristic		
Jobs	**CPU**	**GAP**	**CPU**	**GAP**	**Ratio**
8	11.46	100.00	0.25	100.00	1.00
10	766.85	100.00	0.47	100.00	1.00
12	2539.22	70.06	0.98	70.06	1.00
15	3576.96	54.85	1.82	55.46	1.01
20	3361.76	52.33	6.09	53.97	1.03
30	3600.00	51.19	23.89	58.20	1.14
40	3360.00	35.16	67.88	57.26	1.63
50	3360.00	28.67	169.39	63.30	2.21

Table 5.3: Average results for Model 3

For problem instances with 30 jobs and more, the heuristic is able to outperform CPLEX, still requiring only 87.05 s calculation time.

In the following, the statistics of the detailed results presented in the Appendix are displayed.

For Model3, CPLEX was able to solve 38 out of 120 problem instances with up to 20 jobs. The developed heuristic was also able to solve 38 or 100 % of those instances. However, the heuristic only required an average of 0.58 s while CPLEX was running 543 s to solve the instances.

For 23 out of 120 problem instances, the upper bound of CPLEX was equal to the solution obtained by the heuristic. This can be observed for problem instances with up to 15 jobs. In this case, it is likely that the upper bound of CPLEX is the optimal solution, but optimality could not be proven within the 1 h time limit. The average calculation time required by the heuristic is 2.18 s compared to 3600 s run time of CPLEX.

In 57 out of 120 problem instances, the heuristic outperformed CPLEX within the given time limit. The average calculation time required by the heuristic for those rather large instances was 69.98 s, which is still significantly smaller then the CPLEX time limit of 3600 s.

5.6.4 Model 4

This Section discusses the results derived by solving the test instances using model 4. Table 5.4 presents an overview with average values for all 15 test instances for each problem size. Detailed results for all 120 test instances solved using model 4 can be found in the Appendix.

	MIP		Heuristic		
Jobs	**CPU**	**GAP**	**CPU**	**GAP**	**Ratio**
8	10.04	100.00	0.23	100.00	1.00
10	997.90	100.00	0.52	99.88	1.00
12	2560.05	79.53	1.14	79.45	1.00
15	3528.85	72.00	1.98	72.58	1.01
20	3362.36	60.46	6.41	61.56	1.02
30	3600.00	64.35	30.52	70.24	1.09
40	3360.00	52.29	74.75	66.43	1.27
50	3360.00	48.30	221.88	71.32	1.48

Table 5.4: Average results for Model 4

The results in Table 5.4 show that CPLEX is able to provide optimal solutions for problem instances with up to 10 jobs, given the calculation time limit of 3600 s. With increasing problem sizes, the ability of CPLEX to close the gap between the initial solution value and the best lower bound obtained is strongly diminished.

Looking at the ratio between gap closing ability of CPLEX and the heuristic, it can be observed that on average CPLEX and the heuristic are able to derive similar results for problem instances with up to 20 jobs. However, the average calculation time required by CPLEX is 2092 s compared to only 10.28 s used by the heuristic.

For problem instances with 30 jobs and more, the heuristic is able to outperform CPLEX, still requiring only 109.05 s calculation time on average.

In the following, the statistics of the detailed results presented in the Appendix are displayed.

For Model4, CPLEX was able to solve 39 out of 120 problem instances with up to 20 jobs. The developed heuristic was able to solve 37 or 94.87 % of those instances as well. However, the heuristic only required an average of 0.59 s while CPLEX was running 700 s on average to solve the instances.

For 18 out of 120 problem instances, the upper bound of CPLEX was equal to the solution obtained by the heuristic. This can be observed for problem instances with up to 15 jobs. In this case, it is likely that the upper bound of CPLEX is the optimal solution, but optimality could not be proven within the 1 h time limit. The average calculation time required by the heuristic is 1.68 s compared to 3600 s run time of CPLEX.

In 60 out of 120 problem instances, the heuristic outperformed CPLEX within the given time limit. The average calculation time required by the heuristic for those rather large instances was 83.35 s, which is still significantly smaller then the CPLEX time limit of

3600 s.

6 Decision Support System for the Continuous Casting Planning Problem

Generating a production program, i.e. a schedule to the continuous casting planning problem, is very time consuming in practice. As described in detail in Chapter 2, the problem consists of five interrelated sub decisions that are limited by 21 constraints. When selecting a schedule, ten different and partly conflicting objectives need to be taken into consideration. With order books containing hundreds of customer orders that result in even more slabs that need to be produced, the solution space of the problem cannot be accessed entirely. Current planning procedures concentrate on generating feasible schedules that are able to fulfil all constraints of the problem. The ability to compare alternative production programs and therewith balance between the conflicting objectives, is limited with current mostly manual planning procedures. The DSS described in this Chapter supports planners to find more balanced production programs by generating a larger number of planning alternatives automatically.

As described in Section 3.1, the continuous casting planning problem cannot be solved efficiently with current procedures. Since a planner in practice has only limited time to find production programs, an approach is required that is able to find good schedules regarding solution quality in very short calculation times. The approach of this thesis is a novel decomposition of the problem into two parts. In the first part, *preprocessing*, the sub problems caster selection, slab design and charge batching are considered. The second part consists of cast batching and cast sequencing and is called the *scheduler* part of the DSS. The decomposition is depicted in Figure 6.1.

During cast batching, the batch size of a cast and therewith the total number of setups is determined. This has a major influence on the objective (***O1: Duedate Fulfilment***), since it directly affects the available production time. The most important constraint for the number of setups is the (***CS4: Hot Metal Consumption***) constraint, which is dependent on the sequence of casts. With the selected decomposition, these two aspects can be considered simultaneously within the *scheduler*. In Chapter 4, an efficient procedure for the combined cast batching and cast sequencing problem was developed based on a family scheduling model. In order to be used in practice, the developed

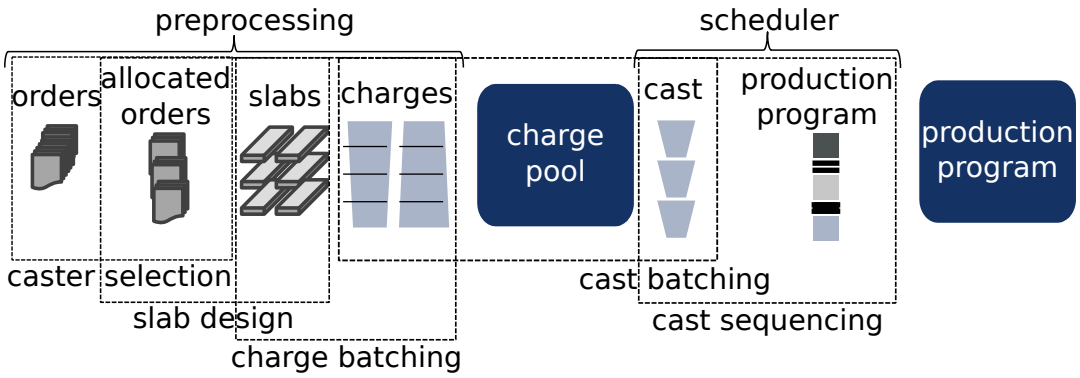

Figure 6.1: Decomposition of the five sub problems of continuous casting planning into the two parts preprocessing (caster selection, slab design, charge batching) and scheduler (cast batching, cast sequencing).

method needs to be embedded into the DSS to generate a production program, i.e. a solution for the complete continuous casting problem.

Starting with given customer orders, the *preprocessing* selects a caster, generates slabs to serve the customer orders and groups slabs together into charges. Within *preprocessing*, the constraints associated with the three sub problems caster selection, slab design and charge batching need to be considered. Since the focus of this thesis is set on the cast batching and cast sequencing sub problems, the *preprocessing* is simplified by only considering the steel grade and weight of slabs but not the width and actual position of slabs within a charge. Therewith, constraints concerned with slab width do not need to be respected when generating production programs. A detailed description of included and excluded constraints is given in Section 6.1. The result of *preprocessing* is a pool of charges. The *scheduler* transforms the pool of charges into a final production program by batching charges into casts and simultaneously sequencing the casts. In order to meet practical requirements, the model described in Section 4.4 needs to be further extended. While the major constraint (***CS4: Hot Metal Consumption***) as well as (***CS1: Caster Capacity***), (***CS2: Setup Type***) are respected, (***CS3: Max Setups***) and (***CB2: Max Batch Size***) are not considered in the formal family scheduling model within this thesis (see Section 6.2 for a detailed discussion).

As described, the continuous casting problem is subject to multiple objectives. Within the formal models presented in Chapter 4, only the objective (***O1: Duedate Fulfilment***) is respected. In order to support a planner with balancing conflicting targets, the DSS needs to be able to generate different production programs that represent decision alternatives when evaluated according to the ten objectives. To realise this requirement, different procedures to generate differing solutions have been developed together with experienced planners from practice (see Section 6.1.5 for procedures within *preprocessing* and Section 6.2.3 for procedures within the *Scheduler* part of the DSS). By adjusting defined parameters during the *preprocessing* and/or *scheduler*, the planner is able to

generate alternative charge pools and in the end alternative production programs that are evaluated for the different planning objectives. This provides him with a decision basis for a more balanced target achievement. As described above, not all constraints are respected within the DSS. Especially the width of individual slabs are not determined within the DSS. In order to create a production plan that can be used for the actual production execution, the selected alternative generated with the DSS needs to be finalized, i.e. dimensions of the slabs need to be set by another planning procedure. This is out of scope of this thesis. The general structure of the DSS is presented in Figure 6.2.

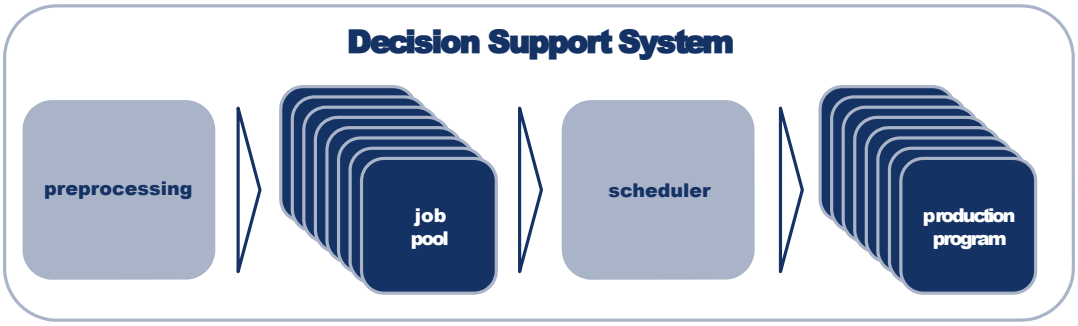

Figure 6.2: The general structure of the DSS including the preprocessing to generate alternative charge pools and the scheduler to generate production programs that strive to minimize total tardiness for each alternative job pool.

The rest of this chapter is organised as follows. First, Section 6.1 describes the procedures used to derive results for caster selection, slab design and charge batching. Second, Section 6.2 describes how the heuristic developed in Chapter 5 is modified to be used in the DSS. Finally, Section 6.3 presents how the DSS is applied in practice based on a case study conducted with an industry partner.

6.1 Preprocessing

The *preprocessing* procedure of the DSS is used to derive a pool of jobs for the *scheduler* of the DSS, as well as to generate alternative pools in order to balance conflicting targets. Input to the *preprocessing* procedure is the pool of known customer orders. A certain share of customer orders has due dates that are weeks ahead. Because the scheduling horizon of continuous casting is usually between 1 and 7 days, only overdue and due orders, i.e. within the selected planning horizon, are considered within the DSS. Exceptions are made whenever non due orders are required in order to fulfil constraints or facilitate objectives. In this case, non due orders are used for complementation. Figure 6.3 depicts the basic steps of the *preprocessing* to generate the job pool for the *scheduler* from existing customer orders. The different sub procedures are displayed from top to bottom.

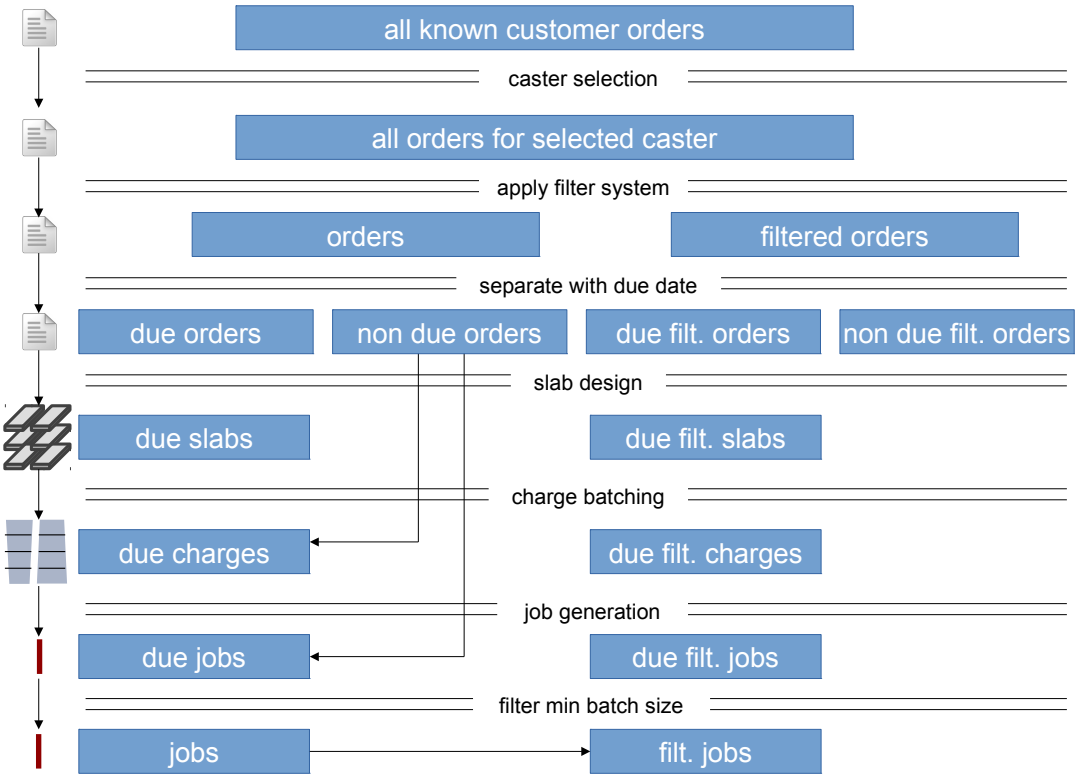

Figure 6.3: Preprocessing steps to generate job pools from production orders

In the first step, the planner selects a caster. Within caster selection (see Section 6.1.1), orders for the caster are extracted for the chosen caster. In the next step, possibly filters can be applied in order to generate alternative production programs. With this step, certain selected orders are forced to be scheduled at the end of the sequence (see Section 6.1.5 for a detailed discussion of this possibility to generate alternative production programs using filters). The result is a set of orders and a set of filtered orders for the selected caster. As described, the problem size is reduced by only considering due orders, i.e. orders within a selected planning period. Both sets, orders and filtered orders are therefore further separated into the sets of due orders, non due orders, due filtered orders and non due filtered orders. In the slab design step, due orders and due filtered orders are transferred into sets of due slabs and due filtered slabs using the slab design procedure described in Section 6.1.2. Since only due orders need to be respected for generating production programs, slab design is only executed for the due share of orders and filtered orders. Within the next step, charge batching 6.1.3, due slabs and due filtered slabs are converted into due charges and due filtered charges. Within charge batching non due orders are used for complementation as depicted with an arrow in Figure 6.3. The reasons are discussed in Section 6.1.3. The next step in preprocessing is to convert due charges and due filtered charges into due jobs 6.1.4. In order to be able to fulfil the hot metal consumption constraint, large batch sizes might be necessary in order to reduce the amount of setups. Therefore, within job generation, additional jobs

are generated from non due orders in case the total number of orders for a given setup family is less then the maximum batch size (see Section 6.1.4 for further details).

Besides the possibility to filter orders, planners can generate alternative solutions by forcing the DSS to only consider jobs with a minimum batch size to facilitate the (***O9: tundish utilization***) objective (see Section 6.1.5 for a detailed discussion). This is realized by applying a filter on the list of due jobs that do not reach the minimum number, leading to a final set of jobs and filtered jobs as the result of *preprocessing*.

6.1.1 Caster Selection

For each order, a list of possible continuous caster is given, which is derived from technological order requirements and caster specification. Therewith, constraint (***CAS1: Applicable Caster***) is considered. Further a preferred caster is marked based the calculation of a technological priority. Both steps are carried out outside the DSS and are considered as given. The DSS by default uses the preferred caster as the decision of caster selection. However, this data field can be adjusted by the planner to influence the allocation of orders to the different continuous caster.

6.1.2 Slab Design

The slab design procedure generates slabs to serve customer orders. As described in Section 2.3.2 this consists of determining the slab dimensions (width, length, thickness, weight) as well as the steel grade of the slab. Since the focus of this thesis is the *scheduler* part, only the weight of slabs is determined, to reduce the problem complexity. Therefore constraint (***SD3: min/max slab weight***) is respected whereas (***SD1: min/max slab length***), (***SD2: min/max slab width***), (***SD4: slab length cluster***) and (***SD5: maximum strand width***) are not considered and need to be respected when the selected production program is finalised by the planning department.

The algorithm to execute the slab design is based on the information of minimum and maximum slab weight, order target slab weight as well as the total weight requested for each order. The target of slab design is to allocate the total weight requested on different slabs, such that each slab is within the defined minimum and maximum range (***SD3: min/max slab weight***) while maximizing the individual slab weight (***O8: caster***

utilization). The algorithm used for slab design is presented in the following:

```
for every due order i available for the selected caster do
    if order requested weight < max slab weight then
        if order requested weight < min slab weight then
            no slab is generated
        else
            generate one slab with the requested order weight
        end
    else
        generate (order requested weight/maximum slab weight) slabs with maximum
        slab weight
    end
    if left over material > min slab weight then
        generate one slab with left over material
    else
        generate one slab with minimum slab weight
    end
end
```

Algorithm 1: Slab Design

6.1.3 Charge Batching

Within charge batching, slabs are grouped together to form charges. Constraint (***CHB2: charge size***) is described by a given minimum, maximum and target charge size. In order to respect (***CHB1: steel quality compatibility***), the given slabs are separated by steel grade and charges are generated for each steel grade independently. In order to facilitate good results for the (***O1: due date fulfilment***) objective, charge batching strives to minimize the deviation of due dates within a charge. At the same time, as few charges as possible are generated in order to minimize objective (***O6: open ordered slabs***). In order to realize this, slabs are first sorted according to the due dates and afterwards subsequently assigned to charges that offer enough space to add the given slab. The algorithm for charge batching is displayed below:

```
for each steel grade do
    get a list of available due slabs
    sort the slabs according to their due dates
    initialize an empty set of temporary charges
    initialize an empty set of finalized charges
    for each slab do
        set placed marker to FALSE
        for each charge in the set of temporary charges do
            if weight of charge + weight of slab < target size then
                insert slab into charge
                leave charge open and continue with next slab
            else if weight of charge + weight of slab < max size then
                ins ert slab into charge
                close charge and continue with next slab
            else if weight of charge < min size AND weight of charge + minimum
            possible weight of slab < max size then
                decrease slab size and insert slab into charge
                close charge and continue with next slab
            open new charge and append it to the temporary charge list
            insert slab into new charge
        end
        if placed marker is FALSE then
            generate new charge and append it to the temporary charge list
            place slab into new charge
        else
            continue with next slab
        end
    end
end
```

Algorithm 2: Charge Batching - Allocation of Slabs

After allocating all due slabs, it is possible that not all charges have been filled up to the minimum charge size. Since only full charges can be produced (***CHB2: charge size***), this would lead to the production of open ordered material which should be minimized (***O6: open ordered slabs***). Therefore another algorithm is executed to fill up charges

using slabs generated from non due orders:

```
for each charge in the set of temporary charge do
    if charge weight < minimum charge size then
        complement charge with non due orders using slab design algorithm
        if charge weight < minimum charge size then
            fill up charge with open ordered material
            add charge to finalized charge list
        else
            add charge to finalized charge list
        end
    else
        add charge to finalized charge list
    end
end
```

Algorithm 3: Charge Batching - Complement Charges

Since the width of slabs is not determined during slab design, the constraints (***CHB3: slab width transition***) and (***CHB4: maximum total strand width***) are not taken into consideration.

6.1.4 Job Generation

After distributing all slabs into charges, the charges need to be transferred into jobs for the *scheduler*. A job used in scheduling is characterized by a processing time (the time required to produce the charge), a due date (a cumulated due date for the entire charge derived from included orders), a cast family (the setup family of the steel grade), a weight (the total weight of the charge) and a maximum batch size (maximum number of consecutive charges based on technological requirements). Most of these parameters can directly derived from the charge to the jobs. The maximum batch size and the cast family are retrieved from a knowledge database. In order to calculate the processing time of a job, additional calculations are necessary. First, an average strand width needs to be derived from the orders contained in the charge

$$\bar{w} = \sum_{slab \in charge} \frac{slab^{maxWidth} + slab^{minWidth}}{2}.$$

As described in Section 2.3.2, it is possible to combine multiple thin slabs into one mother slab in case the sum of slab width is less then the caster's width constraint. In order to respect this within the DSS, for each caster the average strand width $\bar{w}$ is

compared to the maximum cast width of the caster and in case it is possible, the average strand width of the charge is reduced accordingly.

With the additional parameters weight of the charge wt_{ch}, slab thickness produced by caster t_{ca}, casting speed of steel grade and caster $v_{ch,ca}$ and specific weight of the steel grade γ_{sg} derived from knowledge database, the processing time of the job p_j is given as:

$$pj = \frac{wt_{ch}}{2\bar{w}t_{ca}v_{ch,ca}\gamma_{sg}}$$

As described above, the approach of the DSS is to only consider due orders to reduce problem complexity. However, large batch sizes might be necessary within the *scheduler* in order to fulfil the (***CS4: hot metal consumption***) constraint. To facilitate this, additional jobs are generated from the set of non due orders to complement setup families to be able to produce maximum batch sizes if necessary. Further, if the decision maker chose to only accept minimum batch sizes, all jobs of a given setup family are filtered out in case they do not meet the requirements. The following algorithm is used for the job fill up and filter tasks:

```
for each setup family do
    get a list of available jobs
    if number of jobs < maximum batch size then
        complement list of jobs with non due orders using slab design and charge
        batching algorithm
        if number of jobs < minimum batch size then
            filter out all jobs
        else
            add additional jobs to the list of available jobs
            continue with next setup family
        end
    else
        continue with next setup family
    end
end
```

Algorithm 4: Job Generation - Complementation for Hot Metal Consumption Constraint

A final step of job generation is used to support the alternative generation described in Section 6.2.3. In order to improve the (***O3: downstream demand fulfilment***) objective, it needs to be assured that a sufficient amount of material is available in the set of jobs. The following algorithm is implemented to complement the list of jobs by generating additional jobs from non due orders that contain material requested by the

planner:

```
for each user defined target do
    calculate the total required amount over all planning periods
    calculate the total available amount in the list of jobs
    if total required amount < available amount then
        for each steel grade do
            get list of non due orders that could complement to the target
            sort orders according to due date
            complement list of jobs with non due orders using slab design and charge
            batching algorithm
            update available amount
            if total required amount < available amount then
                continue with next steel grade
            else
                continue with next target
            end
        end
    else
        continue with next target
    end
end
```

Algorithm 5: Job Generation - Complementation for Target Achievement

6.1.5 Alternative Generation

One way to generate good production program alternatives regarding objectives other then (***O1: due date fulfilment***) is the filter option implemented within the *preprocessing* of the DSS. By selecting certain parameters, the planner is able to separate a specific set of orders and postpone their production. As depicted in Figure 6.3, the filtered orders result in a filtered set of jobs that is separated from the regular set of jobs. In the evaluation of the production program alternative, the filtered orders are sequenced at the end of the regular orders. This way, different objectives can be influenced as described in the following sub sections.

Sink For each order, the destination after continuous casting is given. This destination could be a specific hot strip mill, the inventory stock or it could be sold directly as a slab without further processing. Within higher planning levels, certain destinations are considered to be less important. Therefore a production program alternative consists of excluding orders for the given destination in order to increase the (***O3: downstream demand fulfilment***) objective.

Devaluation Material Material specified as devaluation material has the property that it is often produced unintended when quality requirements for a superior product are not met during the casting process. Therefore an alternative planning decision is not to focus on orders requesting this type of material. With this option planners are able to improve the (***O5: steel quality upgrading***) target.

Type The type of order determines which production stages are required during further processing. An alternative planning scenario is to only feed a certain production stage from a specific caster. Therefore all other casters need to exclude the orders of the given type. This filter option enables planners to generate solutions that increase the (***O2: up/downstream capacity fulfilment***) objective.

Steelgrade Certain steelgrades are considered unfavourable within higher planning levels (e.g. if the profit margin is very low). This provides another possibility to generate an alternative production programs and improve the target of (***O5: steel quality upgrading***).

Slab Quality Orders that require a very low slab quality can be fulfilled using slabs, which have intentionally be produced for other orders with higher slab quality. Therefore a planning decision is to focus on higher quality slabs and to exclude certain slab qualities from the planning basis. This option again contributes to the (***O5: steel quality upgrading***) objective.

Width The production width has a large impact on the production time of a charge. Together with the number of setups scheduled, the width is the major influencing factor for fulfilling the (***CS4: hot metal consumption***) constraint. By filtering certain width ranges, the planner is able to generate alternative scenarios that use more or less setup processes.

Vacuum Treatment Vacuum treatment is an additional production step that could be required for an order. In order to reduce costs, a planning alternative is not to utilize this production stage at every shift.

Hot Production Hot production requires to roll slabs directly after the continuous casting stage. In order to do so, either a direct hot transport needs to be organized, or an intermediate heating is required. Similar to vacuum treatment, a planning decision is not to produce hot production material at a certain time.

Scorch Degree The scorch degree of an order determines the necessary slab treatment after the casting stage. Certain scorch degrees need to undergo manual treatment before milling can be carried out. Again, excluding such material is an alternative planning decision and contributes to the (***O10: adjusting***) objective.

6.2 Scheduler

As described above, the scheduler receives a job pool as input and generates a production program based on the heuristic developed in Chapter 5. Depending on the chosen number of planning periods (days to be planned) and the number of due orders available, real industry problems can contain hundreds of jobs. In order to be able find schedules to these problems in computation times required in practice, adjustments to the heuristic are necessary.

Section 6.2.1 describes how the tardiness evaluation procedure of the heuristic is adopted to cope with industry size problems. Besides the aspect of calculation times, the heuristic used in practice needs to respect additional constraints not implemented in the formal model in Section 4.4. A second modification to decrease calculation times is described in subsequent Section 6.2.2. Assumptions are formulated that allow the design of acceleration strategies to decrease the number of evaluations required within a move. Finally, Section 6.2.3 presents possibilities to generate alternative production programs, that are associated with the *scheduler* of the DSS.

6.2.1 Tardiness Evaluation

In order to decrease the calculation time for the tardiness evaluation within the DSS, the evaluation as used for Model 4 (see Section 5.2.4) is modified. Instead of applying the labelling algorithm developed in Model 3 (see Section 5.2.3), the evaluation is carried out as if waiting times are permitted in between any two jobs (as used for Model 2 in Section 5.2.2). When converting a sequence into a production program, the "mistake" is corrected by summing up all waiting times in between consecutive jobs of the same setup family and shifting them to the setup process at the beginning of the batch. The error made during this procedure only effects jobs within the respective batch and the influence on the evaluation of an improvement move is negligible.

Different from the models described in Chapter 4, the real industry case has a known starting job 0 derived from the previous planning period. Further, the planning period starts at a certain user specified date t^{start}. Since the calculation of cumulative resource and completion times are defined to start at zero , the tardiness evaluation needs to

respect the actual staring date of the planning. In order to implement the additional constraint (***CB2: maximum batch size***), batch counter CTR_{batch} is used to determine the number of consecutive jobs that are produced without a setup process. Whenever two consecutive jobs are from the same setup family $f_i = f_j$ and the maximum batch size the job b_j is not reached, the counter is incremented by one. Otherwise, either an additional setup process is introduced or a regular setup takes place and the counter is reset to one:

$$CTR_{batch} = \begin{cases} CTR_{batch} + 1 & \text{if } f_i = f_j \text{ and } CTR_{batch} + 1 < b_j \\ 1 & \text{else} \end{cases}$$

The completion times and resource consumptions of the jobs in the sequence can be computed as follows. The first job in the sequence has completion time

$$C_j = \max\{s_{f_0,f_j} + p_j, \frac{Q_j - r^*}{r}\}.$$

and cumulative quantity

$$Q_j = r_j$$

For all other jobs j the completion time is

$$C_j = \begin{cases} \max\{C_i + p_j, \frac{Q_j - r^*}{r}\} & \text{if } f_i = f_j \text{ and } CTR_{batch} + 1 < b_j \\ \max\{C_i + s_{f_i,f_j} + p_j, \frac{Q_j - r^*}{r}\} & \text{else} \end{cases}$$

the cumulative quantity is

$$Q_j = Q_i + r_j,$$

where job i is the predecessor of j.

The tardiness of any job j is

$$T_j = \max\{0, t^{start} + C_j - d_j\}.$$

There are different conditions that need to be fulfilled in a feasible schedule. Due to the described procedures to complement charges and jobs in Section 6.1.4, usually more jobs are available for scheduling then required for the user specified number of planning periods. Since the feasibility requirements are only of interest for the desired planning periods, the following test are only executed for jobs within this time span.

First, the aspect of maximum buffer capacity could lead to an infeasible schedule as described in Section 4.4 due to constraint (***CS4: hot metal consumption***). This is evaluated by calculating the buffer level before and after the execution of a job

$$buffer_{before} = (C_i + s_{ij})r + r^\star - Q_i$$
$$buffer_{after} = C_j r + r^\star - Q_j$$

and comparing both buffer levels to the maximum allowed buffer level B specified by the decision maker. In case this maximum buffer level is exceeded, either before or after the completion of a job, the schedule is considered to be infeasible.

A second feasibility check is implemented within the DSS that has not been addressed in the models described in Chapter 4 to respect the (***CS3: max setups***). Each continuous caster has only a limited number of tundishes available (see Section 2.3.5). One tundish is used for the production of a single batch and another tundish is required whenever a setup process occurs. This translates into the scheduling problem as a restriction on the number of setups that can be carried out during one day. In order to implement this constraint into the sequence evaluation, a setup counter $CTR_{setups,day}$ is used for each day of the specified planning periods. After each job, it is evaluated in which day index *day* the completion of the job is located. This is done by dividing the completion time of the jobs, given in seconds from the beginning of the sequence, by the number of seconds in one day and applying a floor function to the results:

$$day = \left\lfloor \frac{C_j}{86400} \right\rfloor$$

Now each time a setup takes place, i.e. the batch counter CTR_{batch} is reset to 1, the setup counter $CTR_{setups,day}$ for the given day index *day* is increased by one. In case $CTR_{setups,day}$ is greater then the number of ladles available for the given caster (given as a constant in a knowledge data base), the sequence is considered to be infeasible.

6.2.2 Acceleration Strategies

This section describes acceleration strategies that have been implemented for the different improvement operators described in Section 5.3 to decrease the required calculation time. The idea of the acceleration strategies is that a lot of evaluations executed in the operators are not leading to improvements. The goal is to use general knowledge on tardiness minimization to create rules that reduce the number of evaluations executed for each move. In the following, the different acceleration strategies are described for each improvement operator. Two types of acceleration are distinguished in each Section. First, when using the operator to minimize total tardiness and second, when using it for any other target (see Section 6.2.3 for a discussion of different targets).

Job Move

The general job move operator attempts each and every sequence position for a given job, leading to $n - 1$ evaluations per job.

In case of total tardiness evaluation, the lateness of jobs in the sequence can be used. When moving a given job j forward, i.e. towards the beginning of sequence, the completion times of all jobs that are skipped is increased. Therefore, only in case job j is late, the move could contribute to minimize total tardiness. Moving a non late job j forward is similar to moving many late jobs i backwards and skipping job j. Therefore an acceleration can be achieved when only evaluating one direction of movement. The two aspects are described in the following acceleration rule:

- only evaluate forward movements of job j
- only evaluate job j in case it is late

In case of accessing a target different then total tardiness, the lateness cannot be used for excluding evaluations. Instead, moves that do not effect the selected planning period, do not influence the target fulfilment.

- only evaluate moves for jobs within the planning period or move positions that insert jobs into the planning period

Job Exchange

The job exchange operator attempts to exchange the position of a job j with every other job in the sequence, leading to $n-1$ evaluations.

In order to reduce the amount of necessary evaluations, properties of optimal solutions regarding total tardiness as described in Section 3.3 can be used. Emmons (1969) showed for a single machine scheduling problem to minimize total tardiness, that in an optimal sequence j_1 is sequenced before j_2 in case $d_{j_1} \leq d_{j_2}$ and $p_{j_1} \leq d_{j_2}$. Although this aspect is not valid when considering family scheduling and resource requirements, it could be used as an acceleration strategy to increase solution time while allowing optimality violations.

- only exchange jobs j_1, j_2 with $j_1 \prec j_2$, when $p_{j_1} \leq p_{j_2}$ or $d_{j_1} \leq d_{j_2}$

Similar to the job move operator, for evaluating non total tardiness targets, an exchange of two jobs i and j is only considered when at least one of the jobs is scheduled within the planning period.

- only exchange jobs j_1, j_2 when at least one job is within the planning period

Batch Move

The batch move operator evaluates all $n_{batches}$ positions for a selected batch. The acceleration strategy consists of removing unpromising evaluations.

Since all batches are evaluated, only the evaluation of moving in one direction is necessary. The backward movement, towards the beginning of the sequence is chosen due to the following reasons. A backward move has two effects. First, the completion times of skipped batches is increase, what possibly increase total tardiness. Second, the completion time of the moved batch is reduced, what could decrease total tardiness in case the batch is late in the current sequence position. Using this information, an acceleration strategy is defined, that only moves batches backwards, that have at least one late job.

- only evaluate backward movements of batch b
- only evaluate batch b in case at least one containing job is late $\sum_{j \in b} T_j > 0$

Batch Exchange

Similar to the job exchange operator, the properties for minimizing total tardiness derived by Emmons (1969) can be used to reduce the number of necessary evaluations for the batch exchange move. Let us consider to exchange two batches b_1 and b_2, with batch b_1 sequenced before batch b_2 in a given sequence. It is unlikely to improve the solution whenever batch b_2 has a larger batch processing time and at the same time larger average due dates then batch b_1. In case of larger batch processing time, all batches in between the two batches will suffer from prolonged completion times, which has a negative impact on the total tardiness. This could only be compensated when the due dates of batch b_2 are smaller then of batch b_1. An acceleration strategy for batch exchange is defined by only exchanging two batches in case at least one of the two properties are fulfilled.

- only exchange batches b_1, b_2 with $b_1 \prec b_2$, when $\sum_{j \in b_2} p_j < \sum_{j \in b_1} p_j$ or $\frac{\sum_{j \in b_2} p_j}{n_{b_2}} < \frac{\sum_{j \in b_1} p_j}{n_{b_1}}$

Batch Combine

The batch combine operator evaluates all possible sequence positions for the combined batch. This leads to $n_{batches}$ evaluations.

A simple way of reduce the number of evaluations is to only assess positions in between the two former sequence positions of the involved batches b_1 and b_2. Assuming that the batch move operator has been applied already, the sequence positions of the individual batches already reflect the due dates. When combining the two batches, a setup is saved on the expense of forcing jobs with potentially inhomogeneous due dates to be sequenced together. It is unlikely that the combined sequence position is outside the former positions of the individual batches.

- only evaluate sequence positions between batches b_1 and b_2 for the combined batch

Batch Break

The batch break operator evaluates all combinations of sequence positions before and after the broken batch. This leads to $(\frac{n}{2})^2$ evaluations.

One way to improve the calculation speed of the operator is to assess the forward shift and the backward shift of the two broken batch parts separately while leaving the other part at a fixed position. This reduced the number of possible evaluations to n.

- evaluate backwards and forwards movement separately for the two broken parts

Analysis of Acceleration Strategies

The target of the DSS is to analyse different planning alternatives to aid the daily planning process. An analysis consists of multiple solutions that need to be generated to be compared. In order to be applicable in the real industry planning process, a maximum calculation time of 5 min or 300 s needs to be achieved. At the same time, the solution quality is not allowed to drop too far from the best solution that can be generated using the heuristic.

In order to evaluate the applicability of the acceleration strategies describe above and to compile a set of operators that fulfil the requirements stated above, the following analysis has been carried out. For a sample test instance from the industry case study with 308 jobs out of 68 families, each improvement move is executed on the initial solution. Each move is carried out exhaustively, i.e. for the entire sequence (e.g. job move is executed for every job in the sequence). To assess the performance, total tardiness reduction [%] as well as computation time [s] to excessively apply the move are examined. Table 6.1 displays the results for all move operators and the corresponding accelerated move operators.

Table 6.1 displays that in general accelerated moves are not able to achieve the same improvements as non accelerated moves. E.g. the ""job move"" operator is able to decrease total tardiness by 11.44 % when applied excessively to the initial solution of the test instance, whereas the accelerated variant is only able to achieve a total tardiness reduction of 10.01 %. However, the reduction in calculation time is magnitudes higher for accelerated versions. The job exchange and job move operators are the most effective, but naturally tend to require larger calculation times because more moves need to be evaluated. The break operator without applied acceleration has by far the largest calculation time and therefore is unlikely to be useful for an effective and efficient heuristic.

Move Name	Total Tardiness Reduction [%]	Calculation Time [s]
Job Move	11.44	193.71
Job Move Acc.	10.01	17.45
Job Exchange	11.28	177.25
Job Exchange Acc.	11.07	62.61
Batch Move	9.76	17.91
Batch Move Acc.	8.59	3.12
Batch Exchange	8.21	15.23
Batch Exchange Acc.	8.26	7.74
Batch Break	9.91	496.05
Batch Break Acc.	8.33	8.95
Batch Combine	2.93	1.53
Batch Combine Acc.	0.53	0.66

Table 6.1: Performance comparison of different improvement moves with acceleration strategies applied

The relative poor performance of the batch combine operators can be explained by the generation of large batching in the initial solution generation procedure.

In order to find a combination of move operators to be used in the DSS, the following experiment has been conducted. Four different combinations of job moves are defined and evaluated for the test instance using the heuristic framework described in Chapter 5. The first variant **(1)** consists of all move operators in the accelerated version described in this section (i.e. Job Move Acc., Job Exchange Acc., Batch Move Acc., Batch Exchange Acc., Batch Break Acc. and Batch Combine Acc.). The second version **(2)** consists of all accelerated moves on batch level (i.e. Batch Move Acc., Batch Exchange Acc., Batch Break Acc. and Batch Combine Acc.). This version is motivated by the relative small calculation times of batch level operators compared to job level operators. Variant **(3)** is the opposite of variant **(2)** and consists of the two accelerated operators on the job level (i.e. Job Move Acc. and Job Exchange Acc.). Finally, variant **(4)** consists of all non accelerated moves, except the very time consuming batch break move (i.e. Job Move, Job Exchange, Batch Move, Batch Exchange and Batch Combine). Table 6.2 displays the results for running the four variants on the test instance with a 5 min time limit. Column 1 identifies the version, column 2 presents the reduction in total tardiness obtained [%] and column 3 depicts the calculation time [s].

Heuristic Version	Total Tardiness Reduction [%]	Calculation Time [s]
1	14.01	300
2	13.95	105
3	11.71	172
4	14.01	300

Table 6.2: Tested heuristic compositions using the described accelerated and unaccelerated local search operators

The results in Table 6.2 show that the largest decrease in total tardiness can be obtained by applying either all accelerated or non accelerated moves (variants 1 and 4). However, similar good results could be achieved by applying only local search moves on the batch level (variant 2). Relative poor results where obtained by not using operators on the batch level (variant 3). Since the solution time is significantly smaller for variant 2 with only a small reduction in solution quality, this variant is used within the *scheduler* of the DSS.

6.2.3 Alternative Generation

Besides the filter option to generate alternative production programs described in Section 6.1.5, the DSS has three additional options that are associated with the *scheduler*. The first option presented in the first paragraph of this section describes an approach to generate production programs that improve the (***O3: downstream demand fulfilment***) objective by allowing the planner to define targets for certain downstream work systems. The second option described in the second paragraph of this section contributes to the (***O9: tundish utilization***) objective by restricting generated cast sizes to minimum batch sizes defined by the planner. Finally, a third option is presented in the third paragraph of this section that is associated with the (***O1: due date fulfilment***) objective.

Downstream Demand

Derived from higher planning levels, certain daily demands are defined for downstream production systems. One target of continuous casting planning is to fulfil these demands with the scheduled production program (***O3: downstream demand fulfilment***). In order to realize a production program that performs good regarding this target, the following procedure is implemented within the DSS.

The planner can define target values for hot strip mills and production types (that determine the use of downstream work systems) for a production program alternative. For each target $tgt \in TGT$, a desired daily amount is specified $wt^{\star}_{tgt}$. Now this information is used within the evaluation part of the *scheduler*. While the day index is calculated as described in Section 6.2.1, the actual amount for each target and the given day $wt_{tgt,day}$ is increased by the job's contribution to the given target tgt_j:

$$wt_{tgt,day} = wt_{tgt,day} + tgt_j$$

To evaluate the target achievement for the entire sequence, the absolute deviation from

desired daily amount $wt^{\star}_{tgt}$ and actual amount $wt_{tgt,day}$ is calculated for every target and every day. The sum of all absolute deviations is used as the key performance indicator *total target deviation (TTD)* to evaluate the target achievement:

$$TTD = \sum_{tgt \in TGT, day \in PP} (wt_{tgt,day} - wt^{\star}_{tgt})$$

In case an improvement operator move is able to decrease the deviation and the feasibility conditions descried in Section 6.2.1 are fulfilled, the move is executed.

When only *TTD* is considered as a target to guide the search of the heuristic, it is likely that poor tardiness performance results. Therefore, two strategies are implemented to avoid this effect. First, the heuristic builds upon the reference solution generated for minimizing total tardiness. This way, all jobs that are not effected by changes stay in the order that was found best regarding total tardiness. Second, after the target improvement heuristic was executed, an adopted version of the total tardiness heuristic is applied. In this version, reductions in total tardiness are only executed in case the value of *TTD* is not increased.

Minimum Batch Size

In order to increase the performance of the (***O9: tundish utilization***) objective, planners try to avoid the production of small batches. One production program alternative is to only allow batch sizes larger then or equal to defined minimum batch sizes. Within the DSS, the planner has the possibility to define these minimum batch sizes and to force the *scheduler* to respect them.

In order to implement minimum batch sizes, the tardiness evaluation described in Section 6.2.1 needs to be extended. An additional feasibility check is necessary in case the user defined to generate only batches that fulfil at least minimum batch size requirements b_j^{min}. Whenever two consecutive jobs are from different families $f_j \neq f_{j+1}$, the current batch counter CTR_{batch} is compared to the minimum requirement b_j^{min} and the sequence is considered to be infeasible in case $CTR_{batch} < b_j^{min}$.

Hot Metal Supply

Another option to generate alternative production programs is the distribution of hot metal to the different casters. In a situation where hot metal is produced for multiple caster, the amount of hot metal dedicated to each caster is a planning decision. The

daily amount of hot metal determines the hot metal supply rate described in model 4 (see Section 4.4). This has a major influence on the required batch sizes to fulfil the (***CS4: hot metal consumption***) constraint and therefore on the performance of a production program (especially on the (***O1: due date fulfilment***) objective).

Within the DSS the planner has the option to define the amount of hot metal dedicated to a certain caster and therewith can analyse the effect of different distributions of hot metal on the target achievement (see Section 6.3 for an example).

6.3 Industry Case Study

This section describes the use of the developed DSS on three exemplary analysis conducted within an industry case study. In Section 6.3.1 the DSS is used to evaluate alternative planning programs regarding (***O1: due date fulfilment***) and (***O3: downstream demand fulfilment***). In Section 6.3.2 the hot metal supply option of the DSS described in Section 6.2.3 is used to evaluate alternative distributions of hot metal on two different casters regarding the (***O1: due date fulfilment***) objective. Finally, in Section 6.3.3 the DSS is used to evaluate how to balance (***O1: due date fulfilment***) and (***O3: downstream demand fulfilment***) when a hot strip mill could be served by two different casters.

6.3.1 Analysis 1 - Higher Planning Level Decisions

The first application of the DSS describes the evaluation of higher planning level decisions on the performance of a certain continuous caster (called caster 1 in the following). The total daily amount of hot metal available for this caster is 6000 t and a production program should be generated for 2 days. In this example, three different higher planning decisions are assessed. First, the requirement to produce a certain amount of material for a specific target hot strip mill (*Target Hot Strip Mill*). This decision is made at the master planning department to level hot strip milling capacities. In the example discussed in this section, a total of 4000 t of material for hot strip mill 1 is assumed. Second, the constraint not to produce any material that requires vacuum treatment. At the steel mill planning department, the staffing of vacuum facilities is decided upon. Because of cost savings, or the allocation of vacuum capacity to another caster that utilizes the facility, it could be set out of service for a given planning period (*Vacuum Treatment*). Third, the requirement to produce a certain amount of material of a specified product type that is produced on a certain coating work system. This could result from the desire to maximally utilize the coating work system (*Target Product Type*). In this example, product type 1 requires a daily production of 1000 t. While the general target of

continuous casting is to produce optimally regarding due date performance, the ability to meet the higher level planning decisions also contributes to the overall continuous casting performance. Therefore a compromise needs to be found regarding the different targets. In this analysis, a two day planning period is selected for evaluation.

The effect of the higher planning decisions on the performance of the caster can be analysed using the developed DSS. Besides generating a reference solution, that does not consider any higher planning level decisions, parameter adjustments are used to generate seven additional scenarios representing the higher planning level decisions, as presented in Table 6.3. The "X" in the table indicates that the target is active in the scenario.

Number	Alternative Name	Hot Strip Mill	Vacuum Treatment	Product Type
1	Reference Solution	O	O	O
2	HSM	X	O	O
3	Type	O	O	X
4	HSM+Type	X	O	X
5	NOVAC	O	X	O
6	NOVAC+HSM	X	X	O
7	NOVAC+Type	O	X	X
8	NOVAC+HSM+Type	X	X	X

Table 6.3: Different scenario alternatives for analysis 1

Figure 6.4 depicts a comparison between the eight planning alternatives described in Table 6.3, regarding the targets due date fulfilment (upper part of the spider web), ability to meet the requested hot strip mill demand (right part of the spider web), the ability not to include vacuum material (lower part of the spider web) and the ability to meet a requested amount of material of the specified type (left part of the spider web). All four targets are normalized using the best value achieved by any solution as the reference. The first eight radar charts display the performance of each of the alternative solutions, whereas the last radar chart is used as the legend for the Figure.

As depicted in Figure 6.4, there is no alternative that is Pareto optimal, i.e. it is best regarding all performance measures. When considering due date performance, measured as the percentage of due orders produced within the planning period and scaled by the maximum value reached, *(1) Reference*, *(5) NOVAC* and *(7) NOVAC+Type* perform best. This is not surprising, since the heuristic is set to only focus on tardiness minimization for *(1) Reference*. In solutions *(5) NOVAC* and *(7) NOVAC+Type*, all material that requires vacuum treatment is not considered for the planning periods. However, enough due material is available to reach a similar good due date performance as shown in solution *(5) NOVAC*. Solution *(7) NOVAC+Type* has an equal performance because there is no material of the desired type that does not need to undergo vacuum treatment and therefore both scenarios are equal in this case.

The best performance regarding the target to equally supply the selected hot strip

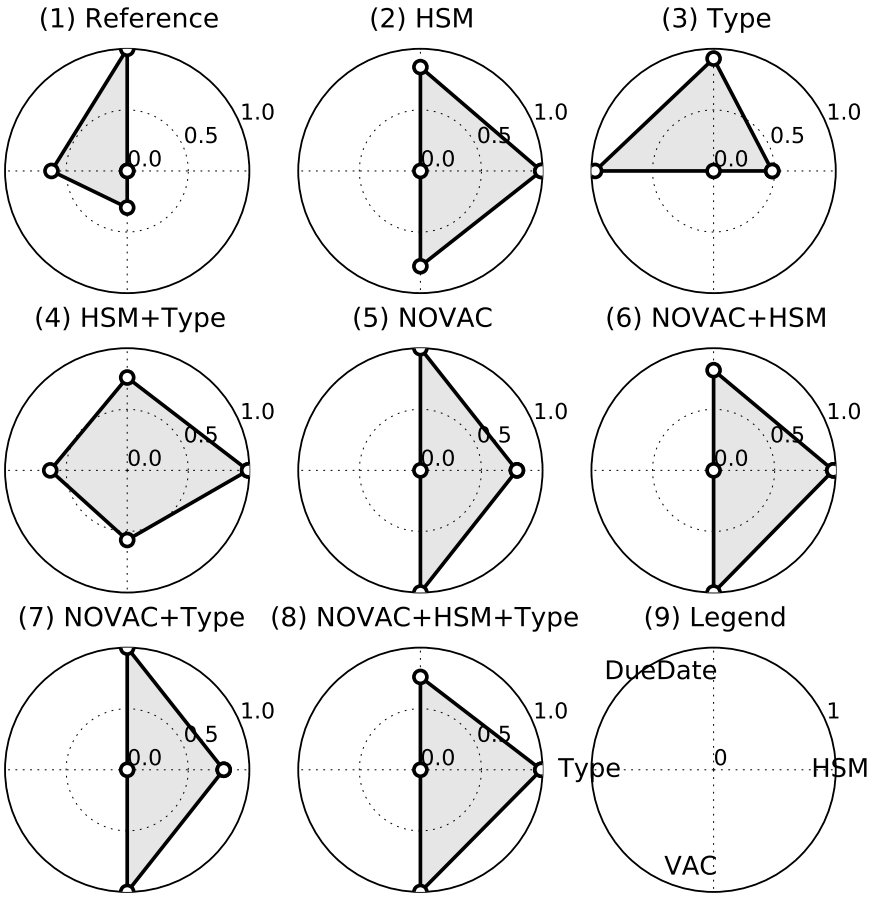

Figure 6.4: Depiction of target fulfilment influenced by higher planning decisions.

mill was reached by solutions *(2) HSM*, *(4) HSM+Type*, *(6) NOVAC+HSM* and *(8) NOVAC+HSM+Type*. This shows that the available orders are sufficient to meet the hot strip mill target. Even when only material that does not need vacuum treatment is considered (*(6) NOVAC+HSM* and *(8) NOVAC+HSM+Type*) or other targets are considered simultaniuously (*(4) HSM+Type* and *(8) NOVAC+HSM+Type*).

Regarding the equal supply of material for the specified production type, only solution *(3) Type* is able to reach a good performance. While solutions *(1) Reference* and *(4) HSM+Type* at least are able to contribute to the target achievement, all other solutions do not include any material for the selected product type.

Solution *(4) HSM+Type* can be considered as the most even compromise, because all targets are at least partially considered. However, when renouncing a target, e.g. the

product type fulfilment in solution *(2) HSM* all other target fulfilments could be increased. The overview in Figure 6.4 can be used to negotiate with all higher planning departments in order to reach a compromise, that best fits the overall company targets at the time of planning.

The selected production program can be used as a specification for the planning department to finalize an actual plan for the execution in the production department.

6.3.2 Analysis 2 - Distribution of Hot Metal on Casters

Whenever more then one caster is fed from the same blast furnace or transport of hot metal/hot steel is possible, a decision on how to allocate the hot metal on the different caster is necessary. Depending on the order structure and the distribution of orders to the casters, a different performance can be achieved when alternating the allocation of hot metal on the casters. The numbers presented in the analysis have been scaled for confidentiality reasons.

In the industry case study, 10 000 t of hot steel (that is processed from hot metal) need to be allocated to two different casters. Table 6.4 depicts possible alternative scenarios considering minimum and maximum production amounts for each caster.

Number	Hot Steel Caster 1	Hot Steel Caster 2
1	3300	6700
2	3600	6400
3	4000	6000
4	4300	5700
5	4600	5400

Table 6.4: Different solution alternatives for Analysis 2

The results for analysis 2 are displayed in Figure 6.5. The blue bars display the due date performance, measured in percentage of due orders produced within the planning period, for caster 1. The same measure is depicted in orange bars for caster 2. The red curve displays the average due date performance for the different solution alternatives.

From Figure 6.5 it can be seen that in general, due dates can be better met in case smaller hot metal amounts are processed. For caster 1, 98 % can be achieved when generating a production program for 3300 t daily hot metal in seven planning days. This value drops to 92 % when 4600 t of daily hot metal need to be consumed. The same effect is shown for caster 2, with a best performance of 94 % with 5400 t and only 90 % with 6400 t. No result is shown for caster 2 and 6700 t, because it was not possible to construct a production program that is able to consume this large amount of daily hot metal with the given set of orders for a period of seven days. This also explains the

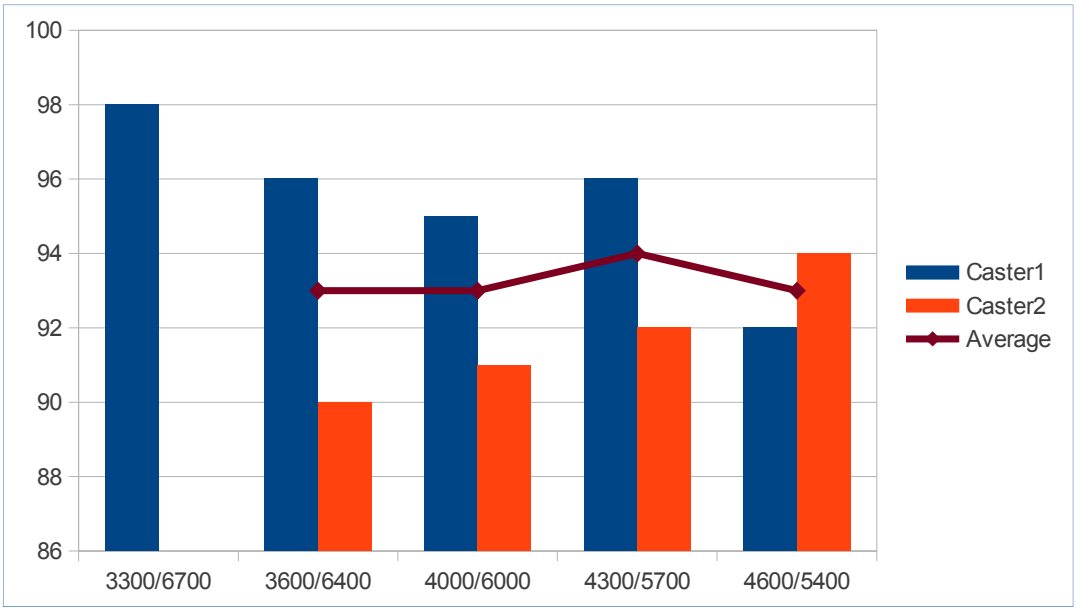

Figure 6.5: Depiction of target fulfilment influenced by higher planning decisions.

performance decrease with increasing hot metal supply. In order to be able to consume the hot metal, a production program needs to contain very large batches. However, the available due material might not be sufficient for that and non due orders are needed to complement the batches.

The difference in average due date performance for the solution alternatives presented in Table 6.4 is negligible as depicted in Figure 6.5. For the given set of orders, the decision on how to distribute the available hot metal on the two caters is independent from the due date performance. However, the analysis revealed, that the developed DSS is applicable to evaluate the distribution of hot metal. The planning department can use this analysis prior to the actual generation of production programs.

6.3.3 Analysis 3 - Distribution of Hot Strip Mill Requirements

As already described for Scenario 1 (see Section 6.3.1), a target amount of material that needs to be produced for a given hot strip mill is determined within higher level planning. Since slab transports between different plant locations are carried out, continuous casting planning can decide to produce certain amounts of material for the respective hot strip mill. Again the presented numbers have been scaled for confidentiality reasons.

In the case of analysis 3, 12 000 t of material needs to be produced at location 1 and transported to the hot strip mill at location 2. Since two casters are available in location 1, a planning decision is to decide how much material for location 2 is produced on each

caster. The analysis is conducted for a planning period of seven days. Table 6.5 depicts different scenarios that will be evaluated in the following. In scenario 1, e.g. the total amount of 12 000 t demanded for the hot strip mill in location 2 is produced at caster 1 and 0 t are produced on caster 2.

Number	Caster 1	Caster 2
1	12000	0
2	10000	2000
3	8000	4000
4	6000	6000
5	4000	8000
6	2000	10000
7	0	12000

Table 6.5: Different scenarios for the production of 12 000 t material for a hot strip mill in location 2 using two possible casters at location 1

The results for analysis 3 are displayed in Figure 6.6. The bars display the average due date performance (***O1: due date fulfilment***) for both casters (right axis) and the orange line depicts the average amount of material that is daily scheduled for the considered hot strip mill (left axis) contributing to the (***O3: downstream demand fulfilment***) objective.

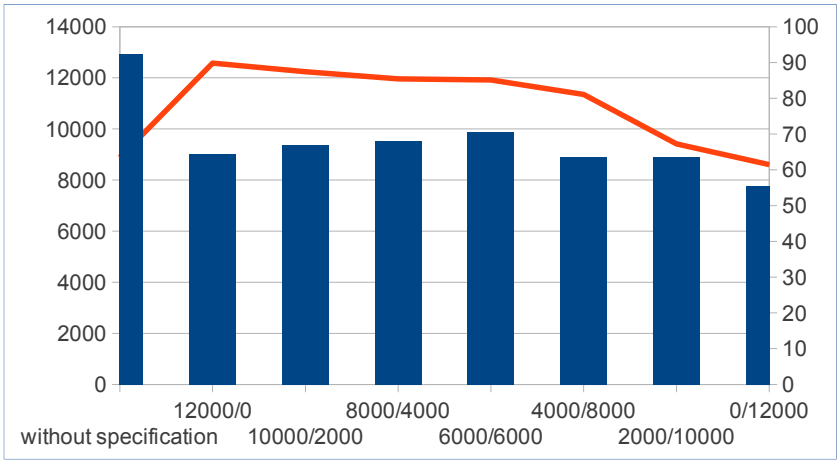

Figure 6.6: Depiction of target fulfilment influenced by higher planning decisions.

The results in Figure 6.6 show that it is possible to supply the desired amount of material for the selected hot strip mill using different distributions of hot metal. The two best distributions (8000/4000 and 6000/6000) also only differ slightly in due date performance. However, the due date performance of any solution that is forced to realize a certain slab supply for the hot strip mill, is much worse in (***O1: due date fulfilment***) compared to the reference solution without considering (***O3: downstream demand fulfilment***) targets.

Again this analysis can be used by the planning department to find a compromise between the conflicting targets. The amount of material produced on the casters is used as a specification for the actual generation of executable production programs.

7 Conclusions

7.1 Summary

Within this PhD thesis, a DSS for the detailed continuous casting planning problem has been developed. The DSS is able to efficiently planning alternatives and supports the planner to find better compromises for conflicting planning objectives.

In Chapter 2, the thesis provides a detailed overview of the continuous casting planning problem. First, the planning problem is set into context of overall steel production planning. By briefly introducing the production processes of the different stages of steel manufacturing, the conflicting requirements for detailed planning are revealed. The description of the higher planning levels of steel production planning demonstrate a main source of conflicting objectives in detailed continuous casting planning. Afterwards, the continuous casting planning problem is classified by the decision problems involved in the problem as well as the planning objectives. Each of the five interrelated problems *Caster Selection*, *Slab Design*, *Charge Batching*, *Cast Batching* and *Cast Sequencing* is discussed and all relevant constraints are worked out. In total 21 constraints have been identified and associated with the different decisions. Additionally, ten different planning objectives where detected and explained. Based on the classification of the planning problem, the existing research was surveyed and the decomposition approach selected for this thesis was deduced (see Chapter 3). When combining *Cast Batching* and *Cast Sequencing* into the *Cast Batching and Sequencing Problem*, the important constraint of hot metal consumption can be considered simultaneously with the batching decision.

For the selected decomposition approach, single machine job scheduling has been identified to deliver promising solution procedures. After identifying the constraints required to model the *Cast Batching and Sequencing* problem as a single machine job scheduling problem, the relevant literature has been surveyed in Chapter 3. Local search heuristics based on simple exchange, move, break and combine operators where successfully implemented for similar problems in the scheduling literature. However, the required model extensions have not been treated sufficiently in the literature and needed to be developed within this thesis.

Chapter 4, presents five different model extensions to the basic job scheduling model.

For each model extension, a detailed description, the formal notation and a MILP model are presented. Model 1 includes the family scheduling character of producing multiple charges from the same tundish. Model 2 extends the basic family scheduling model by introducing a constraint handling the limited supply of hot metal. In model 3, the disability to introduce waiting times after producing a charge within a cast is included. Finally, Model 4 respects maximum buffer capacities for storing surplus hot metal in case a production program is consuming less hot metal then supplied.

In Chapter 5, the heuristic to solve the models is presented and evaluated. An iterated local search framework has been selected based on good performance on similar problems reported in the job scheduling literature. Six different local search operators where selected. The two operators job exchange and job move work on the job level and are the most used operators in heuristics for single machine scheduling problems. In order to exploit the problem characteristic of having jobs grouped together into batches of equal setup families, four additional operators have been selected on the batch level. These are batch move, batch exchange, batch combine and batch break. The local search operators are randomly applied within a single improvement iteration. In order to escape local optima, a problem specific perturbation strategy has been developed. For each of the model extensions described in Chapter 4 a modified evaluation of total tardiness was required. While models 1 and 2 could been evaluated rather easily, models 3 and 4 required a labelling algorithm for the evaluation.

In order to test the effectiveness and efficiency of the developed heuristic, 120 test instances with job sizes between 8 and 50 jobs and distributed on 2 to 6 families have been generated for each model type. The experiments revealed that the heuristic was generating near optimal solutions for all instances that could been solved by CPLEX. Further, the heuristic was able to outperform CPLEX for larger instances that could not been solved within a 1 h time limit.

Within Chapter 6, the DSS has been developed. The general structure of the DSS decomposes the five sub problems of detailed continuous casting planning into a *preprocessing* step (*Caster Selection*, *Slab Design*, *Charge Batching*) and a *scheduler* step (*Cast Batching and Sequencing*). Within the *preprocessing* step, developed procedures transform a given set of customer orders into a pool of charges. This pool of charges is afterwards translated into a set of jobs and planning alternatives are generated using the developed heuristic. In order to respect the multi-objective character of the detailed continuous casting planning problem, three different approaches have been implemented in the DSS. First, a filter possibility to exclude certain orders from being considered early in the schedule. Second, the problem parameters hot metal supply and a minimum batch size can be adjusted to change the character of the problem and thus, generate an alternative. Third, targets for specific downstream work system can be defined and the result is modified to meet these demands. On the example of three possible analysis that arise in practical planning situations, the application of the developed DSS was demonstrated.

7.2 Scientific Contribution

The work carried out during this PhD thesis contributed to the continuous casting research, the scheduling research and delivered a transfer of scientific results into the practise.

As a benefit to the continuous casting research, a detailed description of all relevant decisions, constraints and objectives for the continuous casting planning problem was presented. The developed classification enables researchers in the field to identify research gaps and simplifies the comparison of different approaches. Further, the presented decomposition fills a gap in the existing literature, by including the hot metal consumption constraint into the batching decision of continuous casting. The developed heuristic is able to efficiently generated production programs from an input set of charges. Further simple procedures are presented to generate charges from order information. This presents a good foundation for further developing methodologies to assists the continuous casting planning.

As a benefit to the scheduling research, the basic family scheduling model has been extended to resource constraint scheduling and continuous casting specific setup requirements. For all problem types, benchmark test instances have been generated that are made public under http://www.telematique.eu/research/download. As presented in Chapter 3, large improvements regarding solution quality and calculation time could be realized, when benchmark instances for a scheduling problem are available. By publishing the instances used in this PhD thesis I am hoping to facilitate the generation of solution procedures for family scheduling problems and it's extensions.

As a benefit to the practise, the developed DSS was implemented within an industry case study. Besides the analysis presented in Chapter 6, the planners in the continuous casting department are now able to generate alternative planning solutions quickly and therewith are able to compare a larger set of possible solutions regarding different planning objectives.

7.3 Future Research

As discussed above, this thesis extended the existing single machine job scheduling models by including family scheduling, resource constraints and continuous casting specific setups. For each extension, benchmark test instances have been generated and are now available to use. Similar to the $1|ST_{sd}|\sum_{j\in J} T_j$ problem, a promising improvement could be achieved by testing heuristic approaches, other then the iterated local search heuristic described in this thesis. Further, more advanced exact solution approaches like

column generation could be used to solve larger problem instances and enable a better assessment of heuristic quality.

In the field of continuous casting planning, a novel decomposition approach has been presented combining *Cast Batching* with *Cast Sequencing*. For this combined problem, an efficient heuristic was presented. The other decision problems, *Caster Selection*, *Slab Design* and *Charge Batching* have been treated within a preprocessing to generate a pool of charges. This preprocessing was executed sequentially with rather simple procedures. Within the preprocessing, the practical constraints concerned with the width of slabs have not been respected within this PhD thesis. The procedures that are described in the literature that focus on this aspect of the problem, only consider single strand caster. In the continuous casting problem concerned in this thesis, twin-strand casters are assumed. Since the complexity to generate feasible charges from slabs for this situation is very complex, it was excluded for this thesis. However, the consideration of slab width and the corresponding constraints is very important for the practice and should be subject of future research.

Using a procedure to generate feasible charges from slabs would highly improve the solution quality of the presented DSS, since all constraints would be considered and the results are directly producible. Further, a large number of objectives for the continuous casting planning problem can only be evaluated when the exact dimensions of slabs within a charge are specified. Therefore, the DSS could again be extended to include more objectives into the comparison of solutions.

Bibliography

Abdul-Razaq, T.S., C.N. Potts, and Luk N. Van Wassenhove. "A survey of algorithms for the single machine total weighted tardiness scheduling problem." In: *Discrete Applied Mathematics* 26 (1990), pp. 235–253.

Akrout, Hanen, Bassem Jarboui, Patrick Siarry, and Abdelwaheb Rebaï. "A GRASP based on DE to solve single machine scheduling problem with SDST." In: *Computational Optimization and Applications* 51.1 (2012), pp. 411–435.

Allahverdi, Ali, Jatinder N D Gupta, and Tariq Aldowaisan. "A review of scheduling research involving setup considerations." In: *Omega* 27 (1999), pp. 219–239.

Allahverdi, Ali, C.T. Ng, T.C.E. Cheng, and M.Y. Kovalyov. "A survey of scheduling problems with setups times or costs." In: *European Journal of Operational Research* 187 (2008), pp. 985–1032.

Anghinolfi, Davide and Massimo Paolucci. "a New Ant Colony Optimization Approach for the Single Machine Total Weighted Tardiness Scheduling Problem." In: *International Journal of Operations Research* 5.1 (2008), pp. 44–60.

– "A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times." In: *European Journal of Operational Research* 193 (2009), pp. 73–85.

Antony, Solomon R. and Christos Koulamas. "Simulated annealing applied to the total tardiness problem." In: *Control and Cybernetics* 25.1 (1996), pp. 121–130.

Appelqvist, Patrik and Juha-Matti Lehtonen. "Combining optimisation and simulation for steel production scheduling." In: *Journal of Manufacturing Technology Management* 16.2 (2005), pp. 197–210.

Armentano, Vinícius Amaral and O. C. B. De Araujo. "Grasp with memory-based mechanisms for minimizing total tardiness in single machine scheduling with setup times." In: *Journal of Heuristics* 12.6 (2006), pp. 427–446.

Armentano, Vinicius a. and Renata Mazzini. "A genetic algorithm for scheduling on a single machine with set-up times and due dates." In: *Production Planning & Control* 11.March 2015 (2000), pp. 713–720.

Atighehchian, Arezoo, Mehdi Bijari, and Hamed Tarkesh. "A novel hybrid algorithm for scheduling steel-making continuous casting production." In: *Computers & Operations Research* 36.8 (Aug. 2009), pp. 2450–2461.

Avci, Selcuk, M Selim Akturk, and Robert H Storer. "A problem space algorithm for single machine weighted tardiness problems." In: *IIE Transactions* 35 (2003), pp. 479–486.

Azizoglu, M. and S. Webster. "Scheduling job families about an unrestricted common due date on a single machine." In: *International Journal of Production Research* 35.5 (1997), pp. 1321–1330.

Baker, Kenneth R. "Heuristic procedures for scheduling job families with setups and due dates." In: *Naval Research Logistics* 46.8 (Dec. 1999), pp. 978–991.

Balakrishnan, Anantaram and Joseph Geunes. "Production Planning with Flexible Product Specifications: An Application to Specialty Steel Manufacturing." In: *Operations Research* 51.1 (2003), pp. 94–112.

Bauer, Andreas, Bernd Bullnheimer, Richard F. Hartl, and Christine Strauss. "An ant colony optimization approach for the single machine total tardiness problem." In: *Proceedings of the Congress on Evolutionary Computation (CEC99)* 2 (1999), pp. 1–13.

Bellabdaoui, A. and J. Teghem. "A mixed-integer linear programming model for the continuous casting planning." In: *International Journal of Production Economics* 104.2 (Dec. 2006), pp. 260–270.

Ben-Daya, M. and M. Al-Fawzan. "A simulated annealing approach for the one-machine mean tardiness scheduling problem." In: *European Journal of Operational Research* 93.1 (1996), pp. 61–67.

Bigras, Louis-Philippe, Michel Gamache, and Gilles Savard. "The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times." In: *Discrete Optimization* 5.4 (2008), pp. 685–699.

Bilge, Ümit, Müjde Kurtulan, and Furkan Kiraç. "A tabu search algorithm for the single machine total weighted tardiness problem." In: *European Journal of Operational Research* 176.3 (2007), pp. 1423–1435.

Biskup, Dirk and Wolfgang Piewitt. "Note on 'An efficient algorithm for the single-machine tardiness problem'." In: *International Journal of Production Economics* 66.3 (2000), pp. 287–292.

Blazewicz, J., J.K. Lenstra, and A.H.G.Rinnooy Kan. "Scheduling subject to resource constraints: classification and complexity." In: *Discrete Applied Mathematics* 5.1 (1983), pp. 11–24.

Box, Richard E. and Donald G. Jr. Herbe. "A scheduling model for LTV Steel's Cleveland Works' twin strand continuous slab caster." In: *Interfaces* 18.1 (1988), pp. 42–56.

Bozejko, Wojciech, Józef Grabowski, and Mieczyslaw Wodecki. "Block approach-tabu search algorithm for single machine total weighted tardiness problem." In: *Computers and Industrial Engineering* 50.1-2 (2006), pp. 1–14.

Briskorn, Dirk, Florian Jaehn, and Erwin Pesch. *Genetic Algorithms for Inventory Constrained Scheduling on a Single Machine.* Tech. rep. 2009.

Briskorn, Dirk, Byung Cheon Choi, Kangbok Lee, Joseph Leung, and Michael Pinedo. "Complexity of single machine scheduling subject to nonnegative inventory constraints." In: *European Journal of Operational Research* 207.2 (2010), pp. 605–619.

Briskorn, Dirk, Florian Jaehn, and Erwin Pesch. "Exact algorithms for inventory constrained scheduling on a single machine." In: *Journal of Scheduling* 16 (2013), pp. 105–115.

Carlier, Jacques and A. H. G. R. Kan. "Scheduling subject to nonrenewable-resource constraints." In: *Operations Research Letters* 1.2 (1982), pp. 52–55.

Carroll, D. C. "Heuristic sequencing of single and multiple components." PhD thesis. Massachusetts Institute of Technology, 1965.

Chang, S., Q. Lu, G. Tang, and W. Yu. "On decomposition of the total tardiness problem." In: *Operations Research Letters* 17.5 (1995), pp. 221–229.

Chang, Soo Y., Myong-Rae Chang, and Yushin Hong. "A lot grouping algorithm for a continuous slab caster in an integrated steel mill." In: *Production Planning & Control* 11.4 (Jan. 2000), pp. 363–368.

Cheng, T. C. E., C. T. Ng, and J. J. Yuan. "The single machine batching problem with family setup times to minimize maximum lateness is strongly NP-hard." In: *Journal of Scheduling* 6.5 (2003), pp. 483–490.

Cheng, T. C E, C. T. Ng, J. J. Yuan, and Z. H. Liu. "Single machine scheduling to minimize total weighted tardiness." In: *European Journal of Operational Research* 165.2 (2005), pp. 423–443.

Cheng, T. C E, a. a. Lazarev, and E. R. Gafarov. "A hybrid algorithm for the single-machine total tardiness problem." In: *Computers and Operations Research* 36.2 (2009), pp. 308–315.

Cicirello, Vincent a. and Stephen F. Smith. "Enhancing Stochastic Search Performance by Value- Biased Randomization of Heuristics." In: *Journal of Heuristics* 11 (2005), pp. 5–34.

Congram, Richard K., Chris N. Potts, Van De Velde, and L. Steef. "An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem." In: *INFORMS Journal on Computing* 14.1 (2002), pp. 52–67.

Conway, Richard Walter, William L Maxwell, and L W Miller. *Theory of scheduling*. Mineola, N.Y: Dover, 2003.

Cowling, Peter I. and Wafa Rezig. "Integration of continuous caster and hot strip mill planning for steel production." In: *Journal of Scheduling* 3.4 (July 2000), pp. 185–208.

Cowling, Peter I., Djamila Ouelhadj, and Sanja Petrovic. "A multi-agent architecture for dynamic scheduling of steel hot rolling." In: *Journal of Intelligent Manufacturing* 14 (2003), pp. 457–470.

Crauwels, H. a J, Chris N. Potts, and Luk N. Van Wassenhove. "Local search heuristics for the single machine total weighted tardiness scheduling problem." In: *INFORMS Journal on Computing* 10 (1998), pp. 341–350.

Dash, S., J. Kalagnanam, C. Reddy, and S. H. Song. "Production design for plate products in the steel industry." In: *IBM Journal of Research and Development* 51.3.4 (May 2007), pp. 345–362.

Dawande, M, J Kalagnanam, and HS Lee. "The slab-design problem in the steel industry." In: *Interfaces* 34.3 (2004), pp. 215–225.

Degner, Michael, Reinhardt Fandrich, Gerhard Endemann, Jean Theo Ghenda, Karsten Letz, Hans Bodo Lüngen, Ingo Steller, Hans-Joachim Wieland, Achim Winkhold, Ralf Bartos, and Reinhard Winkelgrund. *Steel Manual*. Verlag Stahleisen, 2008.

Della Croce, Federico, R. Tadei, P. Baracco, and Andrea Grosso. "A new decomposition approach for the single machine total tardiness scheduling problem." In: *Journal of the Operational Research Society* 49 (1998), pp. 1101–1106.

Denton, Brian, Diwakar Gupta, and Keith Jawahir. "Managing Increasing Integrated Steel Variety at Integrated Steel Mills." In: *Interfaces* 33.2 (2003), pp. 41–53.

Drótos, Márton and Tamás Kis. "Scheduling of inventory releasing jobs to minimize a regular objective function of delivery times." In: *Journal of Scheduling* 16 (2013), pp. 337–346.

Du, J. and J. Y.-T. Leung. *Minimizing Total Tardiness on One Machine is NP-Hard*. 1990.

Ecorys. *Study on the Competitiveness of the European Steel Sector*. Tech. rep. August. Ecorys2008: Ecorys, 2008, p. 149.

Emmons, Hamilton. *One-Machine Sequencing to Minimize Certain Functions of Job Tardiness*. 1969.

Ergun, Özlem ÌĹ and James B. Orlin. "Fast neighborhood search for the single machine total weighted tardiness problem." In: *Operations Research Letters* 34.1 (2006), pp. 41–45.

Ernst&Young. *Global Steel 2011 trends 2012 outlook*. Tech. rep. Ernst&Young, 2012, p. 38.

Fadlalla, Adam, James R. Evans, and Martin S. Lew. "A greedy heuristic for the mean tardiness sequencing problem." In: *Computers & Operations Research* 21.3 (1994), pp. 329–336.

Fisher, M. L. "A dual algorithm for the one-machine scheduling problem." In: *Mathematical programming* 11.1 (1976), pp. 229–251.

França, Paulo M., Alexandre Mendes, and Pablo Moscato. "A memetic algorithm for the total tardiness single machine scheduling problem." In: *European Journal of Operational Research* 132 (2001), pp. 224–242.

Frisch, Alan M., Ian Miguel, and Toby Walsh. "Modeling a Steel Mill Slab Design Problem." In: *Proceedings of the IJCAI01 workshop on modeling and solving problems with constraints*. 2001, pp. 39–45.

Frisch, Alan M., I Miguel, and T Walsh. "Symmetry and implied constraints in the steel mill slab design problem." In: *CP 2001. LNCS vol. 2239*. Ed. by T. Walsh. 2001.

Fry, T. D., K. Macleod, and S. Fernandez. "A Heuristic Solution Procedure to Minimize T on a Single Machine." In: *The Journal of the Operational Research Society* 40.3 (1989), pp. 293–297.

Gagné, C., W L Price, and M Gravel. *Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times*. 2002.

Gargani, Antoine and Philippe Refalo. "An efficient model and strategy for the steel mill slab design problem." In: *Principles and Practice of Constraint Programming* (2007), pp. 77–89.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. "Optimization and approximation in deterministic sequencing and scheduling: a survey." In: *Annals of Discrete Mathematics* (1979), pp. 287–326.

Grigoriev, Alexander, Martijn Holthuijsen, and Joris Van De Klundert. "Basic scheduling problems with raw material constraints." In: *Naval Research Logistics* 52 (2005), pp. 527–535.

Grosso, A., F. Della Croce, and R. Tadei. "An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem." In: *Operations Research Letters* 32.1 (2004), pp. 68–72.

Gupta, Jatinder N. D. and Samarn Chantaravarapan. "Single machine group scheduling with family setups to minimize total tardiness." In: *International Journal of Production Research* 46.6 (2008), pp. 1707–1722.

Gupta, Skylab R. and Jeffrey S. Smith. "Algorithms for single machine total tardiness scheduling with sequence dependent setups." In: *European Journal of Operational Research* 175 (2006), pp. 722–739.

Györgyi, Peter and Thomas Kis. "Approximability of scheduling problems with resource consuming jobs." In: *Mathematics of Operations Research* (2014), pp. 1–21.

Harjunkoski, Iiro and Ignacio E. Grossmann. "A decomposition approach for the scheduling of a steel plant production." In: *Computers & Chemical Engineering* 25.11-12 (Nov. 2001), pp. 1647–1660.

Hartmann, Sönke and Dirk Briskorn. "A survey of variants and extensions of the resource-constrained project scheduling problem." In: *European Journal of Operational Research* 207.1 (2010), pp. 1–14.

Hentenryck, P. Van and L. Michel. "The steel mill slab design problem revisited." In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Lecture Notes In Computer Science. Springer, 2008, pp. 377–381.

Herr, Oliver and Asvin Goel. "Comparison of Two Integer Programming Formulations for a Single Machine Family Scheduling Problem to Minimize Total Tardiness." In: *Procedia CIRP* 19.RoMaC (2014), pp. 174–179.

– "Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints." In: *European Journal of Operational Research* in press (2015).

Hirakawa, Yasuhiro. "Quick optimal algorithm for sequencing on one machine to minimize total tardiness." In: *International Journal of Production Economics* 60 (1999), pp. 549–555.

Holsenback, J.Edward and Randolph M. Russell. "A heuristic algorithm for sequencing on one machine to minimize total tardiness." In: *Journal of the Operational Research Society* 43.1 (1992), pp. 53–62.

– "Evaluation of leading heuristics for the single machine tardiness problem." In: *European Journal of Operational Research* 96.3 (1997), pp. 538–545.

Holthaus, O and C Rajendran. "A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs." In: *Journal of the Operational Research Society* 56.8 (2005), pp. 947–953.

IISI. *World Steel in Figures 2011*. Tech. rep. IISI, 2011, p. 15.

Jackson, James R. *Scheduling a production line to minimize maximum tardiness*. Tech. rep. DTIC Document, 1955.

Kanet, J. J. “New Precedence Theorems for One-Machine Weighted Tardiness.” In: *Mathematics of Operations Research* 32.3 (2007), pp. 579–588.

Kanet, John J. “One-Machine Sequencing to Minimize Total Tardiness: A Fourth Theorem for Emmon.” In: *Operations Research* 62.2 (2014), pp. 345–347.

Kao, Gio K., Edward C. Sewell, and Sheldon H. Jacobson. “A branch, bound, and remember algorithm for the 1|r_i |sum t_i scheduling problem.” In: *Journal of Scheduling* 12 (2009), pp. 163–175.

Kellegöz, Talip, Bilal Toklu, and John Wilson. “Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem.” In: *Applied Mathematics and Computation* 199.2 (2008), pp. 590–598.

Kolliopoulos, Stavros G. and George Steiner. “Approximation algorithms for minimizing the total weighted tardiness on a single machine.” In: *Theoretical Computer Science* 355.3 (2006), pp. 261–273.

Kondakci, Suna, O. Kirca, and Meral Azizoglu. “An efficient algorithm for the single machine tardiness problem.” In: *International Journal of Production Economics1* 36 (1994), pp. 213–219.

Koulamas, C.P. “The single-machine total tardiness scheduling problem: Review and extensions.” In: *European Journal of Operational Research* 202.1 (2010), pp. 1–7.

Koulamas, Christos. “The single-machine total tardiness scheduling problem: Review and extensions.” In: *Operations Research* 42.6 (1994), pp. 1025–1041.

– “A faster fully polynomial approximation scheme for the single-machine total tardiness problem.” In: *European Journal of Operational Research* 193.2 (2009), pp. 637–638.

Kovalyov, Mikhail Y. “Improving the complexities of approximation algorithms for optimization problems.” In: *Operations Research Letters* 17.2 (1995), pp. 85–87.

Lawler, E.L. “A fully polynomial approximation scheme for the total tardiness problem.” In: *Operations Research Letters* 1.6 (1982), pp. 207–208.

Lawler, Eugene L. “A "Pseudopolynomial" Algorithm for Sequencing Jobs to Minimize Total Tardiness.” In: *Annals of Discrete Mathematics* 1 (1977), pp. 331–342.

Lee, H. S., S. S. Murthy, S. W. Haider, and D. V. Morse. “Primary production scheduling at steel making industries.” In: *IBM Journal of Research & Development* 40.2 (1996), pp. 231–252.

Lee, Kangbok, Soo Y. Chang, and Yushin Hong. “Continuous slab caster scheduling and interval graphs.” In: *Production Planning & Control* 15.5 (2004), pp. 495–501.

Lee, Young Hoon, Kumar Bhaskaran, and Michael Pinedo. “A heuristic to minimize the total weighted tardiness with sequence-dependent setups.” In: *IIE Transactions* 29.March 2015 (1997), pp. 45–52.

Lenstra, J. K., a. H G Rinnooy Kan, and P. Brucker. *Complexity of Machine Scheduling Problems*. 1977.

Liao, Ching Jong and Hsiao Chien Juan. “An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups.” In: *Computers and Operations Research* 34 (2007), pp. 1899–1909.

Liao, Ching Jong, Hsin Hui Tsou, and Kuo Ling Huang. “Neighborhood search procedures for single machine tardiness scheduling with sequence-dependent setups.” In: *Theoretical Computer Science* 434 (2012), pp. 45–52.

Lin, S-W and K-C Ying. "A hybrid approach for single-machine tardiness problems with sequence-dependent setup times." In: *Journal of the Operational Research Society* 59 (2008), pp. 1109–1119.

Lin, Shih Wei and Kuo Ching Ying. "Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics." In: *International Journal of Advanced Manufacturing Technology* 34.11-12 (2007), pp. 1183–1190.

Lourenco, H. R., O. C. Martin, and T. Stützle. "Iterated Local Search." In: *Handbook of Metaheuristics*. Ed. by Fred Glover and Gary A. Kochenberger. KLUWER ACADEMIC PUBLISHERS, 2003, pp. 321–353.

Lü, Zhipeng and Jin-Kao Hao. "A Critical Element-Guided Perturbation Strategy for Iterated Local Search." In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Cotta Carlos and Peter Cowling. Springer, 2009, pp. 1–12.

Luo, Xiaochuan and Feng Chu. "A branch and bound algorithm of the single machine schedule with sequence dependent setup times for minimizing total tardiness." In: *Applied Mathematics and Computation* 183.1 (Dec. 2006), pp. 575–588.

Mathirajan, M. and a. I. Sivakumar. "A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor." In: *International Journal of Advanced Manufacturing Technology* 29 (2006), pp. 990–1001.

Missbauer, Hubert, Wolfgang Hauber, and Werner Stadler. "A scheduling system for the steelmaking-continuous casting process. A case study from the steel-making industry." In: *International Journal of Production Research* 47.15 (Aug. 2009), pp. 4147–4172.

Moore, J. M. *An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs*. 1968.

Naderi, B., M. Zandieh, A. Khaleghi Ghoshe Balagh, and V. Roshanaei. "An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness." In: *Expert Systems with Applications* 36.6 (2009), pp. 9625–9633.

Naidu, J. T., Jatinder N. D. Gupta, and B. Alidaee. "Insights into two solution procedures for the single machine tardiness problem." In: *Journal of the Operational Research Society* 53 (2002), pp. 800–806.

Nakamura, Nobuto, Teruhiko Yoshida, and Katsundo Hitomi. "Group production scheduling for minimum total tardiness Part(I)." In: *A I I E Transactions* 10.March 2015 (1978), pp. 157–162.

Neumann, Klaus and Christoph Schwindt. "Project scheduling with inventory constraints." In: *Mathematical Methods of Operations Research* 56.3 (2002), pp. 513–533.

Neumann, Klaus, Christoph Schwindt, and Norbert Trautmann. "Scheduling of continuous and discontinuous material flows with intermediate storage restrictions." In: *European Journal of Operational Research* 165.2 (2005), pp. 495–509.

Numao, M. and S.-i. Morishita. "A scheduling environment for steel-making processes." In: *[1989] Proceedings. The Fifth Conference on Artificial Intelligence Applications* (1989).

Ouelhadj, Djamila. "A multi-agent system for the integrated scheduling of steel production." PhD thesis. University of Nottingham, 2003.

Panneerselvam, R. “Simple heuristic to minimize total tardiness in a single machine scheduling problem.” In: *The International Journal of Advanced Manufacturing Technology* 30.7-8 (2006), pp. 722–726.

Panwalkar, S. S., M.L. Smith, and C.P. Koulamas. “A heuristic for the single machine tardiness problem.” In: *European Journal of Operational Research* 70.3 (Nov. 1993), pp. 304–310.

Picard, J.-C. and Maurice Queyranne. *The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling*. 1978.

Potts, C. N. and Luk N. Van Wassenhove. “Single Machine Tardiness Sequencing Heuristics.” In: *IIE Transactions* 23.March 2015 (1991), pp. 346–354.

Potts, C.N and L.N Van Wassenhove. “A decomposition algorithm for the single machine total tardiness problem.” In: *Operations Research Letters* 1.5 (1982), pp. 177–181.

Potts, Chris N. and Mikhail Y. Kovalyov. “Scheduling with batching: A review.” In: *European Journal of Operational Research* 120.2 (Jan. 2000), pp. 228–249.

Potts, Chris N. and Luk N. Van Wassenhove. “Integrating Scheduling with Batching and Lot-Sizing: a review of algorithms and complexity.” In: *The journal of the Operational Research Society* 43.5 (1992), pp. 395–406.

Ragatz, Gary L. “A branch-and-bound method for minimum tardiness sequencing on a single processor with sequence dependent setup times.” In: *Proceedings: twenty-fourth annual meeting of the Decision Sciences Institute*. 1993, pp. 1375–1377.

Rubin, Paul A. and Gary L. Ragatz. “Scheduling in a sequence dependent setup environment with genetic search.” In: *Computers & Operations Research* 22.1 (1995), pp. 85–99.

Ruiz, Rubén and Thomas Stützle. “An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives.” In: *European Journal of Operational Research* 187.3 (2008), pp. 1143–1159.

Sabuncuoglu, Ihsan and Burckaan Gurgun. “A neural network model for scheduling problems.” In: *European Journal of Operational Research* 93.2 (1996), pp. 288–299.

Schaller, Jeffrey E. “Scheduling on a single machine with family setups to minimize total tardiness.” In: *International Journal of Production Economics* 105.2 (Feb. 2007), pp. 329–344.

– “Minimizing total tardiness for scheduling identical parallel machines with family setups.” In: *Computers & Industrial Engineering* 72 (2014), pp. 274–281.

Schaus, Pierre, Pascal Hentenryck, Jean-Noël Monette, Carleton Coffrin, Laurent Michel, and Yves Deville. “Solving Steel Mill Slab Problems with constraint-based techniques: CP, LNS, and CBLS.” In: *Constraints* 16.2 (Aug. 2010), pp. 125–147.

Schwindt, Christoph and Norbert Trautmann. “Batch scheduling in process industries: an application of resource-constrained project scheduling.” In: *OR Spectrum* 22.4 (2000), pp. 501–524.

Selim Akturk, M. and M. Bayram Yildirim. “A new lower bounding scheme for the total weighted tardiness problem.” In: *Computers & Operations Research* 25.4 (1998), pp. 265–278.

Sen, Tapan T., Larry M. Austin, and Parviz Ghandforoush. "An Algorithm for the Single-Machine Sequencing Problem to Minimize Total Tardiness." In: *IIE Transactions* 15.4 (1983), pp. 363–366.

Sen, Tapan and Bolindra N. Borah. "On the Single-Machine Scheduling Problem with Tardiness Penalties." In: *The journal of the Operational Research Society* 42.8 (1991), pp. 695–702.

Sen, Tapan, Joanne M. Sulek, and Parthasarati Dileepan. "Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey." In: *International Journal of Production Economics* 83 (2003), pp. 1–12.

Sewell, Edward C., Jason J. Sauppe, David R. Morrison, Sheldon H. Jacobson, and Gio K. Kao. "A BB&R algorithm for minimizing total tardiness on a single machine with sequence dependent setup times." In: *Journal of Global Optimization* 54 (2012), pp. 791–812.

Shwimer, J. *On the N-Job One-Machine, Sequence-Independent Scheduling Problem with Tardiness Penalties: A Branch-Bound Solution*. 1972.

Sioud, Aymen, Marc Gravel, and Caroline Gagné. "A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times." In: *Computers & Operations Research* 39.10 (2012), pp. 2415–2424.

Smith, Wayne E. "Various optimizers for single-stage production." In: *Naval Research Logistics Quarterly* 3 (1956), pp. 59–66.

Souissi, A and I Kacem. "Minimizing Total Tardiness on a Single Machine with Sequence-Dependent Setup Times." In: *IEEE International Conference on Systems, Man and Cybernectics*. 2004, pp. 1481–1485.

Subramanian, Anand, Maria Battarra, and Chris N. Potts. "An Iterated Local Search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times." In: *International Journal of Production Research* 52.9 (2014), pp. 2729–2742.

Szwarc, W. "Single machine total tardiness problem revisited." In: *Creative and Innovative Approaches to the Science of Management* (1993), pp. 407–419.

– "Decomposition in single-machine scheduling." In: *Annals of Operations Research* 83 (1998), pp. 271–287.

Szwarc, W. and John J Liu. *Weighted Tardiness Single Machine Scheduling with Proportional Weights*. 1993.

Szwarc, W., Federico Della Croce, and Andrea Grosso. "Solution of the single-machine total tardiness problem." In: *Journal of Scheduling* 2.2 (1999), pp. 55–71.

Szwarc, W., Andrea Grosso, and Federico Della Croce. "Algorithmic paradoxes of the single- machine total tardiness problem." In: *Journal of Scheduling* 4.2 (2001), pp. 93–104.

Szwarc, Wlodzimierz. "Some remarks on the decomposition properties of the single machine total tardiness problem." In: *European Journal of Operational Research* 177.1 (2007), pp. 623–625.

Szwarc, Wlodzimierz and Samar K Mukhopadhyay. "Decomposition of the single machine total tardiness problem." In: *Operations Research Letters* 19 (1996), pp. 243–250.

Tan, K.C. and R. Narasimhan. "Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach." In: *Omega* 25.6 (1997), pp. 619–634.

Tan, Keah-Choon, Ram Narasimhan, Paul a Rubin, and Gary L. Ragatz. "A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times." In: *Omega* 28.5 (Oct. 2000), p. 609.

Tanaka, Shunji and Mituhiko Araki. "An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times." In: *Computers & Operations Research* 40.1 (2013), pp. 344–352.

Tanaka, Shunji and Shuji Fujikuma. "A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time." In: *Journal of Scheduling* 15.3 (2012), pp. 347–361.

Tang, Lixin and Shujun Jiang. "The charge batching planning problem in steelmaking process using Lagrangian relaxation algorithm." In: *Industrial & Engineering Chemistry Research* (2009), pp. 7780–7787.

Tang, Lixin and Gongshu Wang. "Decision support system for the batching problems of steelmaking and continuous-casting production." In: *Omega* 36.6 (Dec. 2008), pp. 976–991.

Tang, Lixin, Jiyin Liu, Aiying Rong, and Zihou Yang. "A mathematical programming model for scheduling steelmaking-continuous casting production." In: *European Journal of Operational Research* 120.2 (Jan. 2000), pp. 423–435.

– "A review of planning and scheduling systems and methods for integrated steel production." In: *European Journal of Operational Research* 133.1 (Aug. 2001), pp. 1–20.

Tang, Lixin, Gongshu Wang, and Jingyi Liu. "A combination of Lagrangian relaxation and column generation for order batching in steelmaking and continuous-casting production." In: *Naval Research Logistics (NRL)* 58.4 (2011), pp. 370–388.

Tang, Lixin, Gongshu Wang, and Zhi-Long Chen. "Integrated charge batching and casting width selection at Baosteel." In: *Operations Research* (2014).

Tansel, B. C. and Ihsan Sabuncuoglu. "New Insights on the Single Machine Total Tardiness Problem." In: *1Journal of the Operational Research Society* 48 (1997), pp. 82–89.

Tansel, B. C., B. Y. Kara, and Ihsan Sabuncuoglu. "An efficient algorithm for the single machine total tardiness problem." In: *IIE Transactions* 33.8 (2001), pp. 661–674.

Tasgetiren, M. Fatih, Yun Chia Liang, M. Sevkli, and G. Gencyilmaz. "Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem." In: *International Journal of Production Research* 44.22 (2006), pp. 4737–4754.

Tasgetiren, M. Fatih, Quan Ke Pan, and Yun Chia Liang. "A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times." In: *Computers and Operations Research* 36 (2009), pp. 1900–1915.

Valente, J. M S and R. a F S Alves. "Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups." In: *Computers and Operations Research* 35.August (2008), pp. 2388–2405.

Vanhoucke, Mario and D. Debels. "A finite-capacity production scheduling procedure for a Belgian steel company." In: *International Journal of Production Research* 47.3 (Feb. 2009), pp. 561–584.

Volgenant, A. and E. Teerhuis. "Improved heuristics for the n-job single-machine weighted tardiness problem." In: *Computers and Operations Research* 26.1 (1999), pp. 35–44.

Voß, Stefan and Andreas Witt. "Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application." In: *International Journal of Production Economics* 105.2 (Feb. 2007), pp. 445–458.

Wang, Gongshu and Lixin Tang. "A Lagrangian relaxation for flexible order batching problem in iron and steel industry." In: *IEEE Transactions on industrial technology* (2008), pp. 1–5.

Wang, Xianpeng and Lixin Tang. "A population-based variable neighborhood search for the single machine total weighted tardiness problem." In: *Computers and Operations Research* 36.6 (2009), pp. 2105–2110.

Webster, Scott and Kenneth R. Baker. "Scheduling groups of jobs on a single machine." In: *Operations Research* 43.4 (1995), pp. 692–703.

Wichmann, Matthias Gerhard, Thomas Volling, and Thomas Stefan Spengler. "A GRASP heuristic for slab scheduling at continuous casters." In: *OR Spectrum* 36 (2014), pp. 693–722.

Wilkerson, L. J. and J. D. Irwin. "An Improved Method for Scheduling Independent Tasks." In: *A I I E Transactions* 3.3 (1971), pp. 239–245.

Wodecki, Mieczyslaw. "A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem." In: *International Journal of Advanced Manufacturing Technology* 37.9-10 (2008), pp. 996–1004.

Ying, Kuo Ching, Shih W. Lin, and Chien Y. Huang. "Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic." In: *Expert Systems with Applications* 36 (2009), pp. 7087–7092.

Yu, Wenci. "Augmentations of consistent partial orders for the one-machine total tardiness problem." In: *Discrete Applied Mathematics* 68.1-2 (1996), pp. 189–202.

Zhou, Shuyu and Zhaohui Liu. "A theoretical development for the total tardiness problem and its application in branch and bound algorithms." In: *Computers and Operations Research* 40.1 (2013), pp. 248–252.

Appendix

Detailed Experiment Results

In this appendix detailed results for all instances are reported. Tables 7.1 - 7.8 give the results for Model 1, Tables 7.9 - 7.16 give the results for Model 2, Tables 7.17 - 7.24 give the results for Model 3, Tables 7.25 - 7.32 give the results for Model 4. In the tables the first column provides the name of the instance. The names of the instances are formatted as $|F|$X$|J|_k$ where $k \in \{1, 2, 3, 4, 5\}$ is a counter to distinguish different instances with the same number of jobs $|J|$ and families $|F|$. The second column in the tables give the total tardiness of the initial solution. The next columns present the results for the MIP and the heuristic. The tables shows the calculation time (CPU), lower bound obtained by CPLEX (LB), the upper bound (UP), and the degree to which the gap between initial solution and lower bound is closed (GAP). The last column indicates the ratio between the GAP value for the heuristic divided by the GAP value for the MIP.

Model 1

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X8_1	55704	8.73	37825	37825	100.00	0.01	37825	100.00	1.00
2X8_2	16395	3.59	7450	7450	100.00	0.01	7450	100.00	1.00
2X8_3	27717	1.60	20409	20409	100.00	0.01	20409	100.00	1.00
2X8_4	36554	11.85	30489	30489	100.00	0.01	30489	100.00	1.00
2X8_5	42766	7.45	28104	28104	100.00	0.01	28104	100.00	1.00
3X8_1	4150	0.01	2444	2444	100.00	0.01	2444	100.00	1.00
3X8_2	67692	17.34	54453	54453	100.00	0.02	54453	100.00	1.00
3X8_3	37320	6.16	29895	29895	100.00	0.02	29895	100.00	1.00
3X8_4	8988	2.30	4694	4694	100.00	0.02	4694	100.00	1.00
3X8_5	3479	0.29	0	0	100.00	0.01	0	100.00	1.00
4X8_1	51740	4.04	24191	24191	100.00	0.02	24191	100.00	1.00
4X8_2	28070	2.89	27624	27624	100.00	0.02	27624	100.00	1.00
4X8_3	48078	6.11	45934	45934	100.00	0.02	45934	100.00	1.00
4X8_4	19234	0.91	12432	12432	100.00	0.02	12432	100.00	1.00
4X8_5	92953	10.60	44948	44948	100.00	0.02	44948	100.00	1.00

Table 7.1: Results for Model1 and instances with 8 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X10_1	84884	988.24	53070	53070	100.00	0.03	53070	100.00	1.00
2X10_2	42667	199.78	33714	33714	100.00	0.02	33714	100.00	1.00
2X10_3	28516	128.74	26679	26679	100.00	0.02	26679	100.00	1.00
2X10_4	98254	1555.89	66866	66866	100.00	0.02	66866	100.00	1.00
2X10_5	31682	32.20	24466	24466	100.00	0.02	24466	100.00	1.00
3X10_1	49587	220.41	47780	47780	100.00	0.03	47780	100.00	1.00
3X10_2	64449	237.09	43408	43408	100.00	0.03	43408	100.00	1.00
3X10_3	36900	74.03	29142	29142	100.00	0.02	29142	100.00	1.00
3X10_4	78409	598.68	57941	57941	100.00	0.02	57941	100.00	1.00
3X10_5	60166	54.83	17315	17315	100.00	0.03	17315	100.00	1.00
4X10_1	6667	0.01	6667	6667	100.00	0.02	6667	100	1.00
4X10_2	37546	62.16	29211	29211	100.00	0.02	29211	100.00	1.00
4X10_3	8157	7.83	7926	7926	100.00	0.03	7926	100.00	1.00
4X10_4	108993	460.56	59626	59626	100.00	0.03	59626	100.00	1.00
4X10_5	102591	980.37	63809	63809	100.00	0.03	63809	100.00	1.00

Table 7.2: Results for Model1 and instances with 10 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X12_1	19042	1837.79	18015	18015	100.00	0.02	18015	100.00	1.00
2X12_2	53813	3600.00	21010	28202	78.08	0.03	28202	78.08	1.00
2X12_3	136353	3600.00	26149	100606	32.44	0.02	100606	32.44	1.00
2X12_4	121291	3600.00	20993	42905	78.15	0.02	42905	78.15	1.00
2X12_5	12266	68.94	12156	12156	100.00	0.03	12156	100.00	1.00
3X12_1	23750	353.79	16206	16206	100.00	0.05	16206	100.00	1.00
3X12_2	37493	589.72	15415	15415	100.00	0.04	15415	100.00	1.00
3X12_3	97812	3600.00	16329	43479	66.68	0.05	43479	66.68	1.00
3X12_4	26492	2170.65	18114	18114	100.00	0.04	18114	100.00	1.00
3X12_5	84666	3600.00	22060	39967	71.40	0.05	39967	71.40	1.00
4X12_1	21125	986.06	2453	2453	100.00	0.04	3276	95.59	0.96
4X12_2	45495	243.90	4482	4482	100.00	0.04	4482	100.00	1.00
4X12_3	119403	3600.00	27705	63532	60.93	0.04	68689	55.31	0.91
4X12_4	173096	3600.00	28459	129725	29.99	0.04	129725	29.99	1.00
4X12_5	2615	0.01	2615	2615	100.00	0.04	2615	100	1.00

Table 7.3: Results for Model1 and instances with 12 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X15_1	127971	3600.00	5583	46763	66.35	0.06	48305	65.09	0.98
2X15_2	63924	3600.00	11522	43956	38.11	0.05	43956	38.11	1.00
2X15_3	29305	3600.00	14544	26031	22.18	0.05	26031	22.18	1.00
2X15_4	74120	3600.00	17112	58199	27.93	0.06	58199	27.93	1.00
2X15_5	18787	3600.00	16102	18133	24.36	0.04	18133	24.36	1.00
3X15_1	20798	6.77	0	0	100.00	0.08	0	100.00	1.00
3X15_2	97023	3600.00	9448	60175	42.08	0.08	60279	41.96	1.00
3X15_3	57338	3600.00	2427	6544	92.50	0.06	6544	92.50	1.00
3X15_4	81896	3600.00	16303	38215	66.59	0.06	36408	69.35	1.04
3X15_5	78642	3600.00	9463	22692	80.88	0.09	22692	80.88	1.00
4X15_1	72424	3600.00	19912	48370	45.81	0.07	48370	45.81	1.00
4X15_2	113754	3600.00	22429	61453	57.27	0.09	61453	57.27	1.00
4X15_3	184398	3600.00	19134	106872	46.91	0.10	106659	47.04	1.00
4X15_4	200300	3600.00	12122	101062	52.74	0.09	101062	52.74	1.00
4X15_5	140666	3600.00	0	33961	75.86	0.10	36222	74.25	0.98

Table 7.4: Results for Model1 and instances with 15 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X20_1	265123	3600.00	21343	133403	54.03	0.17	142891	50.14	0.93
4X20_2	207909	3600.00	20262	115389	49.31	0.14	115389	49.31	1.00
4X20_3	167173	3600.00	11174	107780	38.07	0.17	107714	38.11	1.00
4X20_4	317106	3600.00	23204	213753	35.17	0.15	230557	29.45	0.84
4X20_5	177897	3600.00	11414	76847	60.70	0.18	72000	63.61	1.05
5X20_1	969	1.34	0	0	100.00	0.17	0	100.00	1.00
5X20_2	230635	3600.00	14855	87189	66.48	0.23	87189	66.48	1.00
5X20_3	384441	3600.00	10951	191148	51.75	0.20	190261	51.99	1.00
5X20_4	77030	3600.00	11941	69100	12.18	0.17	62981	21.58	1.77
5X20_5	145405	3600.00	11921	64469	60.63	0.17	64331	60.74	1.00
6X20_1	281750	3600.00	2768	63305	78.30	0.20	63095	78.38	1.00
6X20_2	325432	3600.00	13781	180297	46.57	0.25	183133	45.66	0.98
6X20_3	161737	3600.00	11253	87698	49.20	0.26	85416	50.72	1.03
6X20_4	115942	3600.00	7108	75510	37.15	0.20	75195	37.44	1.01
6X20_5	212566	3600.00	16419	106287	54.18	0.18	117404	48.52	0.90

Table 7.5: Results for Model1 and instances with 20 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X30_1	386564	3600.00	19564	167430	59.71	0.90	159149	61.97	1.04
4X30_2	466068	3600.00	18412	192521	61.11	0.92	198440	59.78	0.98
4X30_3	144750	3600.00	14659	100486	34.03	0.51	101701	33.09	0.97
4X30_4	411299	3600.00	13116	203483	52.19	0.70	196553	53.93	1.03
4X30_5	211918	3600.00	15416	98461	57.74	0.52	94777	59.61	1.03
5X30_1	627441	3600.00	14148	274419	57.56	0.66	282216	56.29	0.98
5X30_2	446013	3600.00	8717	131598	71.90	0.46	128107	72.70	1.01
5X30_3	346319	3600.00	12478	152240	58.14	0.59	149660	58.91	1.01
5X30_4	87998	3600.00	13769	67019	28.26	0.48	66859	28.48	1.01
5X30_5	347529	3600.00	12502	91712	76.36	0.72	81627	79.37	1.04
6X30_1	538234	3600.00	14102	168247	70.59	0.79	161687	71.84	1.02
6X30_2	812431	3600.00	12546	378033	54.31	1.00	374328	54.77	1.01
6X30_3	516467	3600.00	18856	305321	42.43	0.95	273848	48.76	1.15
6X30_4	303037	3600.00	12658	116236	64.33	0.70	111969	65.80	1.02
6X30_5	568231	3600.00	16298	267068	54.57	0.86	245326	58.50	1.07

Table 7.6: Results for Model1 and instances with 30 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X40_1	293184	3600.00	25683	257352	13.40	1.25	252664	15.15	1.13
4X40_2	627610	3600.00	25417	504425	20.46	1.42	424931	33.66	1.65
4X40_3	589173	3600.00	20559	183193	71.40	2.11	143547	78.37	1.10
4X40_4	696412	3600.00	26485	400961	44.10	1.53	309855	57.70	1.31
4X40_5	511118	3600.00	19557	191182	65.09	1.06	166866	70.03	1.08
5X40_1	578993	3600.00	13734	366871	37.53	1.82	232224	61.35	1.63
5X40_2	769975	3600.00	21809	487727	37.73	1.44	295300	63.45	1.68
5X40_3	798876	3600.00	20365	432047	47.12	2.14	300267	64.05	1.36
5X40_4	632271	3600.00	12554	245750	62.37	1.78	117655	83.04	1.33
5X40_5	528118	3600.00	28064	348491	35.92	1.91	293866	46.85	1.30
6X40_1	457376	3600.00	10503	124364	74.52	1.77	60362	88.84	1.19
6X40_2	641393	3600.00	11706	267167	59.43	1.74	208497	68.75	1.16
6X40_3	874697	3600.00	25598	579821	34.73	1.62	554819	37.67	1.08
6X40_4	841873	3600.00	11508	503746	40.72	2.86	336077	60.91	1.50
6X40_5	794153	3600.00	25737	420355	48.65	1.84	319011	61.83	1.27

Table 7.7: Results for Model1 and instances with 40 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X50_1	425795	3600.00	12400	115859	74.97	2.74	103198	78.04	1.04
4X50_2	1417590	3600.00	21490	895528	37.39	3.63	431985	70.60	1.89
4X50_3	842473	3600.00	16317	256681	70.91	3.06	110445	88.61	1.25
4X50_4	1660006	3600.00	23791	1001345	40.26	3.50	547299	68.00	1.69
4X50_5	741559	3600.00	30961	419225	45.36	3.00	364669	53.04	1.17
5X50_1	994567	3600.00	18971	706975	29.48	4.09	260422	75.25	2.55
5X50_2	1643400	3600.00	26840	1070550	35.44	6.65	603631	64.32	1.82
5X50_3	364084	3600.00	7758	169476	54.62	3.48	146443	61.08	1.12
5X50_4	1042037	3600.00	13907	428146	59.71	3.90	180526	83.79	1.40
5X50_5	1073657	3600.00	18289	810885	24.90	4.21	461768	57.98	2.33
6X50_1	1002086	3600.00	25420	596912	41.49	5.43	356252	66.13	1.59
6X50_2	529915	3600.00	21654	374260	30.63	3.76	234926	58.04	1.90
6X50_3	1335091	3600.00	34281	927520	31.33	4.60	708052	48.20	1.54
6X50_4	1209774	3600.00	22680	882116	27.60	2.83	433832	65.36	2.37
6X50_5	651200	3600.00	23932	450340	32.02	5.35	272518	60.37	1.89

Table 7.8: Results for Model1 and instances with 50 jobs

Model 2

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X8_1	55704	20.27	39413	39413	100.00	0.02	39413	100.00	1.00
2X8_2	16395	6.02	8671	8671	100.00	0.02	8671	100.00	1.00
2X8_3	27717	1.91	20409	20409	100.00	0.01	20409	100.00	1.00
2X8_4	36554	16.76	30489	30489	100.00	0.02	30489	100.00	1.00
2X8_5	42766	12.84	28701	28701	100.00	0.02	28701	100.00	1.00
3X8_1	4150	0.13	2444	2444	100.00	0.02	2444	100.00	1.00
3X8_2	67692	23.34	55756	55756	100.00	0.02	55777	99.82	1.00
3X8_3	37320	8.58	29895	29895	100.00	0.02	29895	100.00	1.00
3X8_4	8988	2.95	4694	4694	100.00	0.02	4694	100.00	1.00
3X8_5	3805	1.24	918	918	100.00	0.02	918	100.00	1.00
4X8_1	51740	6.61	25699	25699	100.00	0.03	25699	100.00	1.00
4X8_2	29559	5.56	29113	29113	100.00	0.02	29113	100.00	1.00
4X8_3	48078	8.48	45934	45934	100.00	0.02	45934	100.00	1.00
4X8_4	19294	0.47	13189	13189	100.00	0.02	13189	100.00	1.00
4X8_5	92953	15.67	51678	51678	100.00	0.02	51777	99.76	1.00

Table 7.9: Results for Model2 and instances with 8 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X10_1	84884	1698.34	54690	54690	100.00	0.03	54690	100.00	1.00
2X10_2	42667	265.97	33714	33714	100.00	0.02	33714	100.00	1.00
2X10_3	28516	168.90	26679	26679	100.00	0.03	26679	100.00	1.00
2X10_4	98254	2508.31	67671	67671	100.00	0.02	67671	100.00	1.00
2X10_5	31682	76.02	24466	24466	100.00	0.03	24466	100.00	1.00
3X10_1	49587	420.33	47780	47780	100.00	0.04	47780	100.00	1.00
3X10_2	64449	344.40	43408	43408	100.00	0.03	43409	100.00	1.00
3X10_3	36900	120.21	29142	29142	100.00	0.03	29142	100.00	1.00
3X10_4	78409	823.88	57941	57941	100.00	0.03	57941	100.00	1.00
3X10_5	60166	74.09	17315	17315	100.00	0.04	17315	100.00	1.00
4X10_1	6667	0.01	6667	6667	100.00	0.03	6667	100	1.00
4X10_2	37546	110.80	29211	29211	100.00	0.03	29211	100.00	1.00
4X10_3	8157	7.12	7926	7926	100.00	0.05	7926	100.00	1.00
4X10_4	108993	721.89	60815	60815	100.00	0.04	60815	100.00	1.00
4X10_5	102591	2030.22	68178	68178	100.00	0.05	68178	100.00	1.00

Table 7.10: Results for Model2 and instances with 10 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X12_1	19042	3443.00	18015	18015	100.00	0.04	18015	100.00	1.00
2X12_2	53813	3600.00	18759	29561	69.18	0.05	29561	69.18	1.00
2X12_3	136353	3600.00	24043	103491	29.26	0.05	103491	29.26	1.00
2X12_4	121291	3600.00	18815	43492	75.92	0.04	43685	75.73	1.00
2X12_5	12266	279.81	12156	12156	100.00	0.04	12156	100.00	1.00
3X12_1	23750	553.78	16206	16206	100.00	0.06	16206	100.00	1.00
3X12_2	37493	891.21	15415	15415	100.00	0.07	15415	100.00	1.00
3X12_3	97812	3600.00	14249	43479	65.02	0.08	43479	65.02	1.00
3X12_4	26492	3600.00	10317	18114	51.79	0.06	18114	51.79	1.00
3X12_5	84666	3600.00	20977	39967	70.18	0.06	39967	70.18	1.00
4X12_1	21125	1647.46	2579	2579	100.00	0.05	3428	95.42	0.95
4X12_2	45495	192.94	4482	4482	100.00	0.07	4482	100.00	1.00
4X12_3	119403	3600.00	24880	63532	59.11	0.05	63532	59.11	1.00
4X12_4	173096	3600.00	26714	136091	25.28	0.07	136091	25.28	1.00
4X12_5	2615	0.01	2615	2615	100.00	0.06	2615	100	1.00

Table 7.11: Results for Model2 and instances with 12 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X15_1	127971	3600.00	5365	46763	66.24	0.13	48305	64.98	0.98
2X15_2	63924	3600.00	10420	43956	37.32	0.07	43956	37.32	1.00
2X15_3	29305	3600.00	14126	26031	21.57	0.07	26031	21.57	1.00
2X15_4	74120	3600.00	16434	58199	27.60	0.09	58199	27.60	1.00
2X15_5	18787	3600.00	16102	18133	24.36	0.06	18133	24.36	1.00
3X15_1	20798	65.97	0	0	100.00	0.10	0	100.00	1.00
3X15_2	97023	3600.00	9261	60175	41.99	0.12	60279	41.87	1.00
3X15_3	57338	3600.00	2427	6544	92.50	0.13	6544	92.50	1.00
3X15_4	81896	3600.00	16366	38215	66.66	0.10	38215	66.66	1.00
3X15_5	78642	3600.00	9201	22692	80.57	0.14	22692	80.57	1.00
4X15_1	72424	3600.00	19382	48370	45.35	0.11	48370	45.35	1.00
4X15_2	113754	3600.00	20085	61453	55.84	0.14	61453	55.84	1.00
4X15_3	184398	3600.00	18424	106963	46.66	0.15	110647	44.44	0.95
4X15_4	200300	3600.00	12747	101062	52.91	0.15	101062	52.91	1.00
4X15_5	140666	3600.00	0	34958	75.15	0.10	30502	78.32	1.04

Table 7.12: Results for Model2 and instances with 15 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X20_1	265123	3600.00	21642	133150	54.20	0.28	132733	54.37	1.00
4X20_2	207909	3600.00	22709	115389	49.96	0.22	115389	49.96	1.00
4X20_3	167173	3600.00	11242	107714	38.13	0.27	107714	38.13	1.00
4X20_4	317106	3600.00	23305	214142	35.05	0.24	230557	29.46	0.84
4X20_5	177897	3600.00	10672	79192	59.03	0.36	73221	62.60	1.06
5X20_1	969	20.27	0	0	100.00	0.24	0	100.00	1.00
5X20_2	230635	3600.00	15255	91104	64.78	0.36	87189	66.60	1.03
5X20_3	384441	3600.00	10037	190563	51.78	0.39	190261	51.86	1.00
5X20_4	77030	3600.00	12423	70498	10.11	0.30	62981	21.75	2.15
5X20_5	145405	3600.00	11768	64331	60.67	0.25	64331	60.67	1.00
6X20_1	281750	3600.00	2768	69791	75.98	0.33	66761	77.06	1.01
6X20_2	325432	3600.00	13679	177012	47.61	0.43	172187	49.16	1.03
6X20_3	161737	3600.00	10941	92394	45.98	0.43	85416	50.61	1.10
6X20_4	115942	3600.00	9565	77790	35.86	0.30	75236	38.27	1.07
6X20_5	212566	3600.00	16288	117404	48.48	0.32	117404	48.48	1.00

Table 7.13: Results for Model2 and instances with 20 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X30_1	386564	3600.00	19579	191775	53.08	1.45	159149	61.97	1.17
4X30_2	466068	3600.00	18175	257730	46.52	0.94	190882	61.44	1.32
4X30_3	144750	3600.00	15503	102682	32.55	0.82	101701	33.31	1.02
4X30_4	411299	3600.00	13169	213723	49.63	1.00	196553	53.94	1.09
4X30_5	211918	3600.00	16189	100906	56.72	1.00	94777	59.85	1.06
5X30_1	627441	3600.00	15930	360661	43.63	1.38	258312	60.36	1.38
5X30_2	446013	3600.00	8757	167941	63.59	1.01	128006	72.73	1.14
5X30_3	346319	3600.00	13112	163951	54.73	0.91	149660	59.02	1.08
5X30_4	87998	3600.00	13590	67079	28.11	0.79	66859	28.41	1.01
5X30_5	347529	3600.00	12503	172063	52.37	1.61	81627	79.37	1.52
6X30_1	538234	3600.00	16714	282238	49.09	1.14	158595	72.79	1.48
6X30_2	812431	3600.00	11452	389472	52.81	1.57	358200	56.71	1.07
6X30_3	516467	3600.00	21304	332262	37.20	1.36	277794	48.20	1.30
6X30_4	303037	3600.00	13165	126631	60.86	1.38	111969	65.91	1.08
6X30_5	568231	3600.00	17765	316617	45.71	1.41	245603	58.61	1.28

Table 7.14: Results for Model2 and instances with 30 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X40_1	293184	3600.00	24290	261192	11.90	1.94	252664	15.07	1.27
4X40_2	627610	3600.00	25417	526430	16.80	2.66	424931	33.66	2.00
4X40_3	589173	3600.00	20559	349195	42.20	3.93	146161	77.91	1.85
4X40_4	696412	3600.00	26537	566610	19.38	2.95	309750	57.72	2.98
4X40_5	511118	3600.00	17136	291183	44.52	1.75	169769	69.10	1.55
5X40_1	578993	3600.00	13768	437589	25.02	2.59	213008	64.75	2.59
5X40_2	769975	3600.00	23337	495289	36.79	2.12	269797	66.99	1.82
5X40_3	798876	3600.00	20365	604130	25.02	2.00	304394	63.52	2.54
5X40_4	632271	3600.00	13617	283182	56.43	2.69	119019	82.96	1.47
5X40_5	528118	3600.00	30566	378142	30.14	3.46	293866	47.08	1.56
6X40_1	457376	3600.00	11888	173227	63.78	3.33	60362	89.12	1.40
6X40_2	641393	3600.00	13449	367095	43.68	3.25	208497	68.94	1.58
6X40_3	874697	3600.00	23688	643714	27.14	3.09	554819	37.59	1.38
6X40_4	841873	3600.00	11295	567459	33.04	4.15	338766	60.57	1.83
6X40_5	794153	3600.00	24457	413240	49.49	4.36	317720	61.90	1.25

Table 7.15: Results for Model2 and instances with 40 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X50_1	425795	3600.00	13277	169496	62.13	6.02	103198	78.20	1.26
4X50_2	1417590	3600.00	21490	913488	36.11	9.71	431369	70.64	1.96
4X50_3	842473	3600.00	16317	544471	36.07	7.46	104468	89.33	2.48
4X50_4	1660006	3600.00	23791	1371699	17.62	7.05	546758	68.04	3.86
4X50_5	741559	3600.00	29803	494861	34.66	6.28	364669	52.95	1.53
5X50_1	994567	3600.00	21174	514103	49.36	6.01	260422	75.42	1.53
5X50_2	1643400	3600.00	26840	1100611	33.58	6.09	608532	64.02	1.91
5X50_3	364084	3600.00	7430	304413	16.73	7.43	152537	59.31	3.55
5X50_4	1042037	3600.00	16163	494152	53.41	6.62	180526	83.98	1.57
5X50_5	1073657	3600.00	18289	725481	32.99	6.29	457844	58.35	1.77
6X50_1	1002086	3600.00	25420	864384	14.10	6.21	357472	66.00	4.68
6X50_2	529915	3600.00	21588	420953	21.44	6.83	217381	61.48	2.87
6X50_3	1335091	3600.00	34281	943004	30.14	7.60	582573	57.85	1.92
6X50_4	1209774	3600.00	22680	894068	26.59	5.40	433832	65.36	2.46
6X50_5	651200	3600.00	21410	501977	23.69	6.93	285516	58.06	2.45

Table 7.16: Results for Model2 and instances with 50 jobs

Model 3

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X8_1	55704	20.73	40874	40874	100.00	0.29	40874	100.00	1.00
2X8_2	16395	7.67	8671	8671	100.00	0.29	8671	100.00	1.00
2X8_3	27717	1.59	20409	20409	100.00	0.22	20409	100.00	1.00
2X8_4	36554	20.33	30489	30489	100.00	0.25	30489	100.00	1.00
2X8_5	42766	19.13	29904	29904	100.00	0.21	29904	100.00	1.00
3X8_1	4150	0.08	2444	2444	100.00	0.26	2444	100.00	1.00
3X8_2	67692	31.74	55795	55795	100.00	0.41	55795	100.00	1.00
3X8_3	37320	10.58	29895	29895	100.00	0.21	29895	100.00	1.00
3X8_4	8988	3.55	4694	4694	100.00	0.18	4694	100.00	1.00
3X8_5	3805	1.49	918	918	100.00	0.25	918	100.00	1.00
4X8_1	51740	11.99	26695	26695	100.00	0.24	26695	100.00	1.00
4X8_2	30303	6.14	29857	29857	100.00	0.20	29857	100.00	1.00
4X8_3	48078	11.06	45934	45934	100.00	0.23	45934	100.00	1.00
4X8_4	19309	1.21	13655	13655	100.00	0.28	13655	100.00	1.00
4X8_5	92953	24.60	52120	52120	100.00	0.22	52120	100.00	1.00

Table 7.17: Results for Model3 and instances with 8 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X10_1	84884	2370.24	55881	55881	100.00	0.49	55881	100.00	1.00
2X10_2	42667	379.95	33714	33714	100.00	0.50	33714	100.00	1.00
2X10_3	28516	300.66	26679	26679	100.00	0.37	26679	100.00	1.00
2X10_4	98254	3146.18	68666	68666	100.00	0.44	68666	100.00	1.00
2X10_5	31682	68.43	24466	24466	100.00	0.37	24466	100.00	1.00
3X10_1	49587	439.61	47780	47780	100.00	0.42	47780	100.00	1.00
3X10_2	64449	412.76	43408	43408	100.00	0.38	43408	100.00	1.00
3X10_3	36900	139.01	29142	29142	100.00	0.42	29142	100.00	1.00
3X10_4	78409	789.04	57941	57941	100.00	0.47	57941	100.00	1.00
3X10_5	60166	93.63	17315	17315	100.00	0.60	17315	100.00	1.00
4X10_1	6667	0.02	6667	6667	100.00	0.42	6667	100	1.00
4X10_2	37546	141.14	29211	29211	100.00	0.47	29211	100.00	1.00
4X10_3	8157	22.44	7926	7926	100.00	0.57	7926	100.00	1.00
4X10_4	108993	852.70	61535	61535	100.00	0.61	61535	100.00	1.00
4X10_5	102591	2346.90	69151	69151	100.00	0.46	69151	100.00	1.00

Table 7.18: Results for Model3 and instances with 10 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X12_1	19042	3600.00	11986	18015	14.55	0.66	18015	14.55	1.00
2X12_2	53813	3600.00	15693	30453	61.28	1.28	30453	61.28	1.00
2X12_3	136353	3600.00	23703	107190	25.89	1.38	107190	25.89	1.00
2X12_4	121291	3600.00	18463	43492	75.66	1.33	43492	75.66	1.00
2X12_5	12703	398.83	12593	12593	100.00	1.40	12593	100.00	1.00
3X12_1	23750	1354.80	16206	16206	100.00	0.74	16206	100.00	1.00
3X12_2	37493	802.63	15415	15415	100.00	0.73	15415	100.00	1.00
3X12_3	97812	3600.00	13170	43479	64.19	1.18	43479	64.19	1.00
3X12_4	26492	3600.00	11485	18114	55.83	0.89	18114	55.83	1.00
3X12_5	84666	3600.00	19580	39967	68.68	0.84	39967	68.68	1.00
4X12_1	21125	2902.54	2705	2705	100.00	0.99	2705	100.00	1.00
4X12_2	45495	229.48	4482	4482	100.00	0.82	4482	100.00	1.00
4X12_3	119403	3600.00	26376	63532	60.06	0.80	63532	60.06	1.00
4X12_4	173096	3600.00	26168	136758	24.73	0.88	136758	24.73	1.00
4X12_5	2615	0.01	2615	2615	100.00	0.76	2615	100	1.00

Table 7.19: Results for Model3 and instances with 12 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X15_1	127971	3600.00	5219	46763	66.16	1.99	46763	66.16	1.00
2X15_2	63924	3600.00	9964	43956	37.01	1.70	43956	37.01	1.00
2X15_3	29305	3600.00	14201	26031	21.68	1.53	26031	21.68	1.00
2X15_4	74120	3600.00	15990	58199	27.39	1.61	58199	27.39	1.00
2X15_5	18787	3600.00	15421	18133	19.43	1.45	18133	19.43	1.00
3X15_1	20798	3254.33	0	0	100.00	1.91	0	100.00	1.00
3X15_2	97023	3600.00	8563	60175	41.65	1.93	60021	41.83	1.00
3X15_3	57338	3600.00	2427	7496	90.77	1.91	6544	92.50	1.02
3X15_4	81896	3600.00	15592	40569	62.33	1.72	36408	68.61	1.10
3X15_5	78642	3600.00	9199	22692	80.57	1.90	22692	80.57	1.00
4X15_1	72424	3600.00	16412	48918	41.97	1.62	48370	42.94	1.02
4X15_2	113754	3600.00	21762	61453	56.85	1.81	61453	56.85	1.00
4X15_3	184398	3600.00	18112	107084	46.49	2.07	107084	46.49	1.00
4X15_4	200300	3600.00	11592	101062	52.59	2.26	101062	52.59	1.00
4X15_5	140666	3600.00	0	31067	77.91	1.93	31067	77.91	1.00

Table 7.20: Results for Model3 and instances with 15 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X20_1	265123	3600.00	21506	132853	54.29	6.02	132733	54.34	1.00
4X20_2	207909	3600.00	21121	115389	49.53	5.24	115389	49.53	1.00
4X20_3	167173	3600.00	12253	108737	37.72	5.37	107714	38.38	1.02
4X20_4	317106	3600.00	23633	213753	35.22	7.03	213753	35.22	1.00
4X20_5	177897	3600.00	11114	72446	63.23	7.81	72000	63.49	1.00
5X20_1	969	26.38	0	0	100.00	4.14	0	100.00	1.00
5X20_2	230635	3600.00	14890	89445	65.44	6.57	87189	66.49	1.02
5X20_3	384441	3600.00	11050	200634	49.23	7.56	190261	52.00	1.06
5X20_4	77030	3600.00	8286	68276	12.73	4.34	62981	20.44	1.60
5X20_5	145405	3600.00	12519	74397	53.44	6.53	64331	61.01	1.14
6X20_1	281750	3600.00	2768	72481	75.01	5.05	66318	77.22	1.03
6X20_2	325432	3600.00	14497	178470	47.26	8.00	172187	49.29	1.04
6X20_3	161737	3600.00	11130	85416	50.68	4.75	85416	50.68	1.00
6X20_4	115942	3600.00	8539	75277	37.86	5.59	75277	37.86	1.00
6X20_5	212566	3600.00	16186	108014	53.24	7.29	107298	53.60	1.01

Table 7.21: Results for Model3 and instances with 20 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X30_1	386564	3600.00	19829	181165	56.01	19.41	158092	62.30	1.11
4X30_2	466068	3600.00	20611	235036	51.86	35.97	190882	61.78	1.19
4X30_3	144750	3600.00	6551	102377	30.66	17.19	99664	32.62	1.06
4X30_4	411299	3600.00	13471	218770	48.40	19.93	194038	54.61	1.13
4X30_5	211918	3600.00	16131	105347	54.43	16.65	94777	59.83	1.10
5X30_1	627441	3600.00	16041	325794	49.34	26.31	258022	60.42	1.22
5X30_2	446013	3600.00	10009	152285	67.37	21.04	128006	72.94	1.08
5X30_3	346319	3600.00	12101	168277	53.27	14.95	149660	58.84	1.10
5X30_4	87998	3600.00	4599	67761	24.27	15.80	66859	25.35	1.04
5X30_5	347529	3600.00	12524	114924	69.43	36.22	79428	80.03	1.15
6X30_1	538234	3600.00	16739	192892	66.22	24.65	158595	72.80	1.10
6X30_2	812431	3600.00	12489	430278	47.77	33.85	358412	56.76	1.19
6X30_3	516467	3600.00	19029	319878	39.52	31.55	273848	48.77	1.23
6X30_4	303037	3600.00	13899	123783	62.00	17.80	111969	66.08	1.07
6X30_5	568231	3600.00	17731	308181	47.24	26.98	238915	59.82	1.27

Table 7.22: Results for Model3 and instances with 30 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X40_1	293184	3600.00	12720	270621	8.04	48.09	252664	14.45	1.80
4X40_2	627610	3600.00	37741	464718	27.61	77.85	424931	34.36	1.24
4X40_3	589173	3600.00	31271	285883	54.36	94.49	143547	79.88	1.47
4X40_4	696412	3600.00	25674	447987	37.04	77.90	309750	57.65	1.56
4X40_5	511118	3600.00	17136	321693	38.35	46.39	166866	69.69	1.82
5X40_1	578993	3600.00	14222	366116	37.69	64.25	213008	64.80	1.72
5X40_2	769975	3600.00	25463	458556	41.83	104.23	269797	67.18	1.61
5X40_3	798876	3600.00	22917	441287	46.08	85.56	293756	65.10	1.41
5X40_4	632271	3600.00	12554	380088	40.69	64.03	117655	83.04	2.04
6X40_1	457376	3600.00	12114	188539	60.38	61.58	57585	89.79	1.49
6X40_2	641393	3600.00	11803	306900	53.13	74.68	208497	68.76	1.29
6X40_3	874697	3600.00	36584	598956	32.90	75.49	554819	38.17	1.16
6X40_4	841873	3600.00	11295	640612	24.23	71.92	319527	62.89	2.60
6X40_5	794153	3600.00	27991	601620	25.13	71.77	309896	63.21	2.52

Table 7.23: Results for Model3 and instances with 40 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X50_1	425795	3600.00	15012	225781	48.69	119.44	103198	78.53	1.61
4X50_2	1417590	3600.00	21490	931298	34.83	233.69	430751	70.69	2.03
4X50_3	842473	3600.00	16388	618477	27.12	150.54	105119	89.26	3.29
4X50_4	1660006	3600.00	18676	1142522	31.53	163.36	537155	68.41	2.17
4X50_5	741559	3600.00	31310	548576	27.17	175.20	364669	53.06	1.95
5X50_1	994567	3600.00	19871	614189	39.03	165.56	254427	75.94	1.95
5X50_2	1643400	3600.00	30152	1258894	23.83	222.79	603631	64.45	2.70
5X50_3	364084	3600.00	7430	296383	18.98	156.94	146443	61.02	3.21
5X50_4	1042037	3600.00	11017	609230	41.98	155.17	181297	83.48	1.99
5X50_5	1073657	3600.00	19189	626315	42.42	162.52	450608	59.09	1.39
6X50_1	1002086	3600.00	14348	682046	32.40	258.23	350495	65.97	2.04
6X50_2	529915	3600.00	19384	390174	27.37	168.46	217381	61.22	2.24
6X50_3	1335091	3600.00	34281	1074879	20.00	217.20	582573	57.85	2.89
6X50_5	651200	3600.00	24853	559159	14.69	191.75	272080	60.53	4.12

Table 7.24: Results for Model3 and instances with 50 jobs

Model 4

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X8_1	49867	16.43	40874	40874	100.00	0.27	40874	100.00	1.00
2X8_2	23983	11.18	8671	8671	100.00	0.26	8671	100.00	1.00
2X8_3	61521	2.51	20409	20409	100.00	0.20	20409	100.00	1.00
2X8_4	51949	13.31	30489	30489	100.00	0.26	30489	100.00	1.00
2X8_5	52136	20.50	30192	30192	100.00	0.21	30192	100.00	1.00
3X8_1	20501	0.68	2444	2444	100.00	0.23	2444	100.00	1.00
3X8_2	59675	20.94	55795	55795	100.00	0.24	55795	100.00	1.00
3X8_3	49171	9.90	29895	29895	100.00	0.19	29895	100.00	1.00
3X8_4	13122	3.75	4694	4694	100.00	0.23	4694	100.00	1.00
3X8_5	10374	1.19	918	918	100.00	0.24	918	100.00	1.00
4X8_1	48266	13.03	26695	26695	100.00	0.26	26695	100.00	1.00
4X8_2	38030	2.41	29857	29857	100.00	0.08	29857	100.00	1.00
4X8_3	67450	9.23	45934	45934	100.00	0.23	45934	100.00	1.00
4X8_4	36360	1.29	13655	13655	100.00	0.34	13655	100.00	1.00
4X8_5	68640	24.17	52120	52120	100.00	0.24	52120	100.00	1.00

Table 7.25: Results for Model4 and instances with 8 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X10_1	82335	2997.65	55881	55881	100.00	0.60	55881	100.00	1.00
2X10_2	74701	520.19	35337	35337	100.00	0.60	35337	100.00	1.00
2X10_3	74668	337.15	55279	55279	100.00	0.52	55279	100.00	1.00
2X10_4	92222	3579.99	68666	68666	100.00	0.45	68666	100.00	1.00
2X10_5	63596	117.01	24466	24466	100.00	0.63	24466	100.00	1.00
3X10_1	81567	542.56	47780	47780	100.00	0.43	47780	100.00	1.00
3X10_2	84009	558.15	43408	43408	100.00	0.47	43408	100.00	1.00
3X10_3	71232	256.77	29142	29142	100.00	0.47	29142	100.00	1.00
3X10_4	103900	1245.31	57941	57941	100.00	0.50	57941	100.00	1.00
3X10_5	62464	110.03	17315	17315	100.00	0.50	18099	98.26	0.98
4X10_1	60091	1.71	6667	6667	100.00	0.42	6667	100.00	1.00
4X10_2	56115	148.60	29211	29211	100.00	0.41	29211	100.00	1.00
4X10_3	54464	42.28	9303	9303	100.00	0.51	9303	100.00	1.00
4X10_4	109544	1319.90	61535	61535	100.00	0.68	61535	100.00	1.00
4X10_5	109457	3191.21	69151	69151	100.00	0.55	69151	100.00	1.00

Table 7.26: Results for Model4 and instances with 10 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X12_1	65138	3044.70	18015	18015	100.00	1.04	18015	100.00	1.00
2X12_2	113771	3600.00	15029	30453	84.38	1.49	30453	84.38	1.00
2X12_3	128325	3600.00	22175	108899	18.30	1.62	108899	18.30	1.00
2X12_4	112482	3600.00	14698	43492	70.55	1.70	43492	70.55	1.00
2X12_5	61206	652.86	13137	13137	100.00	1.40	13137	100.00	1.00
3X12_1	84910	1625.90	16206	16206	100.00	0.77	16206	100.00	1.00
3X12_2	63933	1628.17	15415	15415	100.00	1.04	15415	100.00	1.00
3X12_3	109140	3600.00	11746	43479	67.42	1.36	43479	67.42	1.00
3X12_4	65853	3600.00	9558	18114	84.80	1.18	18114	84.80	1.00
3X12_5	101413	3600.00	16558	39967	72.41	0.91	39967	72.41	1.00
4X12_1	50209	2351.09	2705	2705	100.00	0.92	3276	98.80	0.99
4X12_2	58171	294.72	4482	4482	100.00	0.94	4482	100.00	1.00
4X12_3	136471	3600.00	24306	63532	65.03	0.87	63532	65.03	1.00
4X12_4	185712	3600.00	22758	136672	30.09	1.08	136672	30.09	1.00
4X12_5	45938	3.31	2615	2615	100.00	0.82	2615	100.00	1.00

Table 7.27: Results for Model4 and instances with 12 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
2X15_1	129442	3600.00	2149	46763	64.95	1.85	46763	64.95	1.00
2X15_2	135144	3600.00	7375	43956	71.37	2.03	43956	71.37	1.00
2X15_3	117760	3600.00	10466	33185	78.83	1.84	33185	78.83	1.00
2X15_4	138149	3600.00	10410	58199	62.59	1.86	58199	62.59	1.00
2X15_5	123634	3600.00	12018	22975	90.18	1.66	22975	90.18	1.00
3X15_1	64896	2532.71	0	0	100.00	1.57	0	100.00	1.00
3X15_2	170043	3600.00	5630	60274	66.76	1.87	60175	66.82	1.00
3X15_3	97450	3600.00	2427	6544	95.67	2.53	6544	95.67	1.00
3X15_4	145991	3600.00	14204	40569	79.99	2.03	36408	83.15	1.04
3X15_5	115500	3600.00	8989	22692	87.13	1.96	22692	87.13	1.00
4X15_1	143307	3600.00	15363	48370	74.20	2.11	48370	74.20	1.00
4X15_2	176081	3600.00	17716	61453	72.38	1.94	61453	72.38	1.00
4X15_3	171239	3600.00	16602	107084	41.49	2.06	106659	41.76	1.01
4X15_4	151941	3600.00	10934	101062	36.08	2.20	101062	36.08	1.00
4X15_5	83645	3600.00	0	34808	58.39	2.25	30502	63.53	1.09

Table 7.28: Results for Model4 and instances with 15 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X20_1	255894	3600.00	11256	148176	44.03	5.69	142891	46.19	1.05
4X20_2	349652	3600.00	14320	119603	68.60	6.78	115389	69.86	1.02
4X20_3	234221	3600.00	2871	109091	54.09	5.67	107714	54.68	1.01
4X20_4	345891	3600.00	11553	214178	39.40	6.22	213753	39.52	1.00
4X20_5	219619	3600.00	7718	76443	67.57	6.72	72000	69.66	1.03
5X20_1	51672	35.44	0	0	100.00	3.62	0	100.00	1.00
5X20_2	198919	3600.00	10513	87189	59.30	6.73	87438	59.17	1.00
5X20_3	354969	3600.00	8305	191553	47.14	7.33	190261	47.51	1.01
5X20_4	241928	3600.00	7943	68276	74.21	5.98	62981	76.48	1.03
5X20_5	244604	3600.00	6084	65170	75.23	7.32	64331	75.58	1.00
6X20_1	168777	3600.00	2768	66761	61.45	6.37	66318	61.72	1.00
6X20_2	264475	3600.00	8808	180725	32.76	6.81	172187	36.10	1.10
6X20_3	268969	3600.00	6822	87991	69.04	3.13	85416	70.02	1.01
6X20_4	175382	3600.00	5497	77520	57.60	6.06	75277	58.93	1.02
6X20_5	240067	3600.00	11023	110762	56.45	11.69	107298	57.97	1.03

Table 7.29: Results for Model4 and instances with 20 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X30_1	582668	3600.00	3926	181722	69.28	23.75	157131	73.53	1.06
4X30_2	521741	3600.00	11365	215698	59.96	30.52	190882	64.83	1.08
4X30_3	443084	3600.00	286	103934	76.59	16.31	99664	77.56	1.01
4X30_4	574239	3600.00	6412	221204	62.17	24.01	191700	67.37	1.08
4X30_5	365957	3600.00	8454	116510	69.77	20.11	94777	75.85	1.09
5X30_1	665628	3600.00	10302	313875	53.68	48.18	258022	62.20	1.16
5X30_2	435302	3600.00	4693	169708	61.68	42.56	128006	71.36	1.16
5X30_3	634293	3600.00	846	179090	71.86	24.19	149660	76.51	1.06
5X30_4	406038	3600.00	7371	75412	82.93	26.35	67827	84.84	1.02
5X30_5	362700	3600.00	10086	115534	70.10	31.48	79428	80.33	1.15
6X30_1	531750	3600.00	10088	218233	60.10	18.95	158595	71.53	1.19
6X30_2	729692	3600.00	1798	388882	46.82	30.61	358412	51.01	1.09
6X30_3	678466	3600.00	12	309876	54.33	51.50	273848	59.64	1.10
6X30_4	430915	3600.00	10472	136116	70.12	37.40	111969	75.86	1.08
6X30_5	602690	3600.00	8734	270995	55.85	31.79	238915	61.25	1.10

Table 7.30: Results for Model4 and instances with 30 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X40_1	845835	3600.00	2130	357549	57.87	82.12	252664	70.31	1.21
4X40_2	964199	3600.00	6	530368	44.99	71.39	424931	55.93	1.24
4X40_3	847041	3600.00	0	278367	67.14	112.68	144812	82.90	1.23
4X40_4	957772	3600.00	268	399078	58.35	79.55	309750	67.68	1.16
4X40_5	735978	3600.00	24	303539	58.76	91.84	166866	77.33	1.32
5X40_1	945374	3600.00	0	422518	55.31	51.61	213008	77.47	1.40
5X40_2	822105	3600.00	3567	361697	56.25	128.41	269837	67.47	1.20
5X40_3	923663	3600.00	0	396055	57.12	19.81	297915	67.75	1.19
5X40_4	628945	3600.00	0	225810	64.10	24.78	117655	81.29	1.27
6X40_1	771828	3600.00	0	215101	72.13	36.95	51007	93.39	1.29
6X40_2	831109	3600.00	1957	324799	61.06	89.81	208497	75.09	1.23
6X40_3	1087414	3600.00	1176	690153	36.57	142.88	554819	49.03	1.34
6X40_4	980084	3600.00	0	492527	49.75	44.33	348180	64.47	1.30
6X40_5	948465	3600.00	60	522378	44.93	145.10	319011	66.37	1.48

Table 7.31: Results for Model4 and instances with 40 jobs

		MIP				Heuristic			
Instance	**Start**	**CPU**	**LB**	**UB**	**GAP**	**CPU**	**UB**	**GAP**	**Ratio**
4X50_1	1138794	3600.00	0	282775	75.17	226.98	103198	90.94	1.21
4X50_2	1550732	3600.00	0	858106	44.66	201.73	430751	72.22	1.62
4X50_3	1117723	3600.00	0	466983	58.22	246.63	104468	90.65	1.56
4X50_4	1690356	3600.00	0	980546	41.99	314.95	546758	67.65	1.61
4X50_5	1270069	3600.00	0	678833	46.55	215.53	365413	71.23	1.53
5X50_1	1566497	3600.00	2779	517158	67.11	148.09	254427	83.91	1.25
5X50_2	1455756	3600.00	0	908699	37.58	300.42	602364	58.62	1.56
5X50_3	1026309	3600.00	0	413481	59.71	83.95	160277	84.38	1.41
5X50_4	1080624	3600.00	0	578369	46.48	183.54	180526	83.29	1.79
5X50_5	1562113	3600.00	0	719140	53.96	567.47	450608	71.15	1.32
6X50_1	1212442	3600.00	32	722661	40.40	362.46	350495	71.09	1.76
6X50_2	1128870	3600.00	0	567702	49.71	80.20	243557	78.42	1.58
6X50_3	1651105	3600.00	0	938007	43.19	147.45	582573	64.72	1.50
6X50_5	1459566	3600.00	52	587733	59.73	248.87	270256	81.49	1.36

Table 7.32: Results for Model4 and instances with 50 jobs

Declaration

I, Oliver Herr hereby declare that I have written this PhD thesis independently, unless where clearly stated otherwise. I have used only the sources, the data and the support that I have clearly mentioned. This PhD thesis has not been submitted for conferral of degree elsewhere.

I confirm that no rights of third parties will be infringed by the publication of this thesis.

Bremen, July 28, 2015

Oliver Herr