Junior Management Science 10(3) (2025) 781-809

Junior Management Science

www.jums.academy
ISSN: 2942-1861

Numerical Studies for the Scheduling of Continuous Annealing Lines

Hagen Alexander Hönerloh

Leibniz University Hannover

Abstract

The continuous annealing of flat steel improves its properties for applications such as automotive manufacturing. Scheduling these processes on Parallel Heterogeneous Annealing Lines (PHALs) is complex due to diverse coil properties, incompatible process modes, and due date constraints. Introducing stringers to address incompatibilities between steel sheets raises costs, energy use, and CO2 emissions, highlighting the need for optimized scheduling. This thesis implements a mathematical model in Python using the Gurobi solver to optimize PHAL scheduling by minimizing stringer usage while meeting tardiness constraints. The model is extended to include coil-specific release dates and expanded to address trade-offs between stringer use, tardiness, and due date deviations, including earliness. A computational study evaluates the model under various scenarios, examining the effects of coil heterogeneity, urgency, process flexibility, and stringer processing times. Results show that optimized schedules reduce stringer use and delays, particularly under high process flexibility. These findings demonstrate the potential of optimization to improve efficiency and sustainability in steel production while guiding future research in dynamic scheduling approaches.

Keywords: continuous annealing lines; Gurobi solver; scheduling optimization; steel industry; stringer minimization

1. Introduction

1.1. Subject and motivation

Our modern economy thrives on digital transformation and its many facets. One key aspect is the computerization of processes using advanced digital technologies. Through computerization, companies can create and implement optimal schedules for their processes and thereby increase efficiency and productivity, as companies always sought to improve their decision-making through new scheduling and planning methods.[1] But the potential impact of computerization and digitalization could be far greater than anything before.

The steel industry is one of the many industries that stand to benefit significantly from this development. By leveraging digital technologies, manufacturers are able to minimize production time, while also maximizing the use of resources and thus their profits.[2] The impact on this highly competitive industry, which supplies us with materials needed for everyday appliances, railways, or even buildings, is astonishing.[3] One process in the steel industry, that can benefit greatly from digitalization, is the continuous annealing of flat steel, a method of processing steel to change its physical and mechanical properties.[4] In this process, coils of cold rolled flat steel are processed in furnaces with different annealing temperatures and transport speeds that make up the process mode of an individual coil. Different process modes lead to different mechanical and physical properties of the steel.[4] Hence, the mode is chosen according to the desired characteristics. Cold rolled flat steel is essential for building cars, and household appliances and has many more areas of application.[5] Scheduling the continuous annealing process on continuous annealing lines (CALs) is a difficult task due to the different properties of the flat steel, processing modes, and other aspects like due dates. If the properties or processing modes of two successive coils are too different, scrap

[1] Tang and Meng, 2021, p. 1
[2] Iannino et al., 2021, p. 620
[3] Zhao and Yang, 2016, p. 3417 and Terence, 2020
[4] Sarna, 2013
[5] Commodity-Inside, 2019

DOI: https://doi.org/10.5282/jums/v10i3pp781-809

coils, so-called stringers, must be added between the coils to bridge these differences, resulting in additional material costs and a loss of efficiency.[6]

The impact of this loss of efficiency cannot be overstated. Not only does the manufacturer lose valuable production time, but he also has to waste energy, increasing the amount of CO_2 emitted per kg of steel. With a share of seven % of the world's annual CO_2 emissions, the steel industry is already one of the most energy-intensive industries worldwide.[7] In the meanwhile, in the European Union, the steel sector is facing an ever-increasing cost of energy, as well as an increase in $CO2$ price per ton of $CO2$ emitted, which some experts predict could reach 50% by the end of the decade.[8]

Hence, manufacturers should have a serious interest in optimizing their production schedules by minimizing the introduction of stringers and thus the costs caused by material and energy wastage, loss of efficiency, and CO_2 emissions, which can be achieved through the use of digital technologies.

1.2. Research question and structure of the work

This thesis is based on a paper by Wegel et al. (2024), in which they propose a mathematical model to optimize the scheduling of the continuous annealing process. The objective of this model is to minimize the introduction of stringers in a schedule, while also considering a tardiness constraint that limits the number of delays. It is designed for short-term planning and can therefore be used at the operational level of operations management.[9] So far, this model has only been implemented in the Julia programming language and used with a proprietary algorithm. Thus, the model has not yet been implemented in the popular Python programming language. This implementation forms the basis of this thesis.

During the course of this work, a numerical study will be conducted on this model and on its expansions created during the work. It will consist of different sections, which study the impact of certain parameters and scenarios on the scheduling of CALs. The results will be thoroughly analyzed and discussed to derive emerging trends and formulate managerial insights.

First, the CALs for flat steel will be explained in terms of their design and their scheduling, which will be followed by the current state of research. Afterwards, an explanation of the underlying problem and the mathematical model itself will be given. This will be continued by further extensions of the model with the aim of mimicking the real-world process. The numerical studies conducted will be the core of this thesis. Different trends that occur with increasing instance sizes will be explored first, followed by sections in which the impact of scenarios and the alteration of certain parameters on the scheduling process will be investigated. Furthermore, the base model will be compared with the extended model regarding its performance and solutions. The results of the computational study carried out on the Python implementation and the instances used are presented and discussed, leading to a summary of the work and an outlook for future research, further extensions, and managerial insights.

2. Continuous Annealing Lines for Flat Steel

2.1. Process description

The industrial continuous annealing process is made possible by CALs, which consist of several sections. One such CAL is depicted in figure 1. CALs are not standalone, they are part of a greater complex consisting of different areas dedicated to different processing methods, like the cold rolling and galvanization of steel.[10] This study will focus on the annealing of cold rolled steel, which is usually the bottleneck of steel processing.[11] In some cases, it may also be possible and advantageous to anneal hot rolled steel.[12] In comparison to hot rolling, cold rolling happens at far lower temperatures. But these processes are complementary and not substitutional, since the hot rolling process is an upstream process of cold rolling. While hot rolling rolls the steel to the desired width, cold rolling reduces its thickness by up to 90 %, increasing its strength and hardness but also severely diminishing its ductility and increasing its brittleness.[13] To improve these mechanical and physical properties, the steel strips are annealed in CALs. Through recrystallization of microstructures and other processes, the steel regains some of its lost characteristics, especially its ductility.[14]

The process starts at the entry section with the coils of cold rolled flat steel and the so-called pay-off reel, as depicted in figure 1. These coils stem from the upstream cold rolling process. The pay-off reels fixate the coils of flat steel with a mandrel and rotate to continuously unwind them.[16] Further downstream, a welding machine automatically welds the tail end to the head end of two consecutive coils together, thus providing continuous strip feeding to the succeeding sections.[17] The welding process itself is of the utmost importance since a weld break could result in a complete line shutdown.[18] Therefore, compatibility between consecutive coils regarding their thickness and width has to be ensured. If the two consecutive coils are incompatible with each other, a stringer has to be added between them to assure a continuous annealing process, resulting in a loss of efficiency and higher material and energy costs.[19] Stringers can be reused, but only for a limited number of times.[20]

[6] Besson, 1998, p. 29
[7] Joint-Research-Centre, 2022
[8] Twidale et al., 2021 and Krukowska, 2021
[9] Karakostas et al., 2020, p. 2 and Karakostas et al., 2019, p. 2
[10] Zhao and Yang, 2016, p. 3418
[11] Li et al., 2023, p. 1
[12] Steel-Warehouse, n.d.
[13] Zhao and Yang, 2016, p. 1
[14] Sarna, 2013
[15] Takurou et al. (2016), p. 113
[16] Jiyuan-Shenzhou-Industry, n.d.
[17] United-Enterprises, n.d.
[18] Williamson, n.d., p. 3
[19] Besson, 1998, p. 29
[20] Mujawar et al., 2004, p. 1

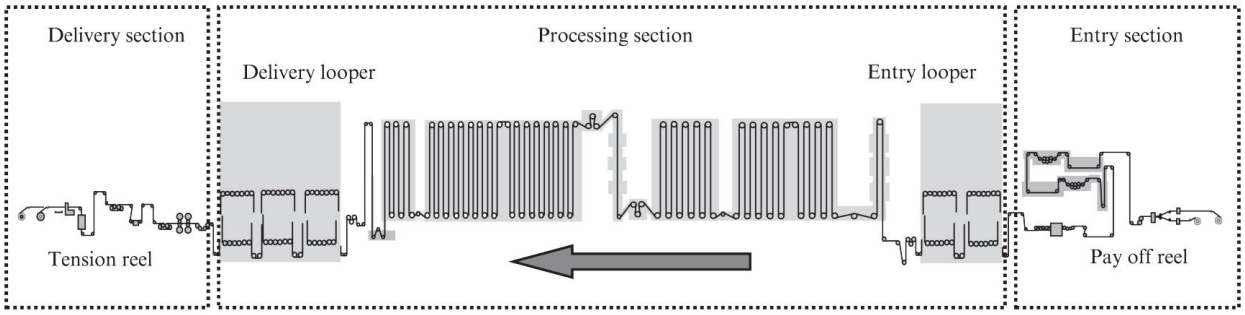

Figure 1: A Continuous Annealing Line as implemented in a Japanese factory.[15]

After degreasing by a degreasing unit, the steel strip enters the process section, where it first feeds into an entry looper.[21] This looper counteracts interruptions in the continuous annealing process, such as the welding process, and maintains a continuous strip speed throughout the annealing process. It achieves this by moving its rolls apart from each other, thus increasing the length of strip steel it can hold and the distance the steel has to traverse. The continuous strip then enters several furnaces, that can reach temperatures of up to 850 °C, while maintaining a strip speed of up to 800 m/min.[22] The steel strip cycles through the furnaces for several minutes, depending on the chosen processing mode. As previously described, a processing mode is a combination of annealing temperature and strip speed, which leads to certain steel properties.[23] Since the strip speed and furnace temperature can only be adjusted in a certain range from the preceding coil to the succeeding one, compatibility between their processing modes is necessary or else a stringer has to be introduced between them.[24] It should also be mentioned, that adjusting the furnace temperature costs a lot of energy and should therefore be minimized.[25]

After the heat treatment, the steel strip is cooled down in several steps, until it feeds into the delivery or exit looper.[26] This looper works like the entry looper and can therefore compensate for interruptions like maintenance and the cutting process, which follows downstream.[24] After the cutting, the tension reel recoils the steel strip into the previous coils.[27] Compared to the batch annealing process, in which the steel is processed as a coil, the continuous annealing process has a higher efficiency and productivity, while also delivering a more uniform product, regarding the physical and mechanical properties of the strip steel. Some of its disadvantages, however, are the large amounts of space and capital needed to construct it.[23]

[21] United-Enterprises, n.d.
[22] United-Enterprises, n.d.
[23] Sarna, 2013
[24] Besson, 1998, p. 29
[25] Zhao and Yang, 2016, p. 3417
[26] Sarna, 2013
[27] Jiyuan-Shenzhou-Industry, n.d.

2.2. Differentiation from literature

The topic of scheduling has been a research subject since the early 20th century and is nowadays one of the most researched fields in operations research, with several hundred papers published each year.[28] Scheduling in the steel industry particularly is one of the most difficult and complex problems, due to the complexity of the steel industry itself and therefore, there have been many attempts to optimize certain aspects of it.[29] This section will feature different approaches concerning CALs and continuous galvanizing lines, a process further downstream of the annealing process that has many similarities with CALs regarding its scheduling.[30]

Li et al. (2023) aimed to minimize earliness and tardiness costs, as well as setup costs, which occur through changes in annealing temperature, on one processing line.[31] The authors noted, that these objectives conflict with each other since minimizing the setup costs by constructing large batches of similar coils could increase the earliness and tardiness costs and vice versa. Stringers were not directly considered. To minimize the three objectives, they used an adaptive multi-objective differential evolutionary algorithm (MODE) based on deep reinforcement learning (DRL). Several other papers also include the use of MODEs for the optimization of CALs and for other processes as well, like for the hot rolling process.[32] The usage of MODEs can be highly effective, as can be seen in a study by Dong et al. (2021), in which they were able to find optimal schedules for the color-coating process for up to 400 coils with three different objectives. During the color coating process, protective or decorative coatings are applied to the steel.[33]

Zhao and Yang (2016) tested a discrete differential evolution algorithm (DDE), which uses discrete job permutations to find optimal solutions, to average the line capacity and minimize the changeover costs. These occur through annealing temperature changes in the furnace, and Zhao and Yang (2016) concluded that their algorithm performed well for up to 90 coils on parallel processing lines.[34] Even though the dif-

[28] Potts and Strusevich, 2009, p. 1
[29] Harjunkoski and Grossmann, 2001, p. 1649
[30] Zhao and Yang, 2016, p. 3418
[31] Li et al., 2023, p. 1-2
[32] Tang and Wang, 2010, p. 104-116 and Pan et al., 2019, p. 327-348
[33] Sarna, 2014
[34] Tasgetiren et al., 2007, p. 271-273

ferences between annealing temperatures of adjacent coils were considered, stringers and tardiness were not. Since DDEs are difficult to apply to real-life problems, the algorithms' usefulness may be in question.[35]

Evolutionary algorithms also tend to focus on local optima, which can be less optimal than a global optimum.[36] The so-called Tabu search solves this problem and was used for optimization in continuous galvanizing lines.[37] Gao et al. (2008) claimed their respective approaches to be very effective as they were able to solve instances with up to 100 coils in only some seconds. But it has to be noted that some aspects like tardiness were not considered. A study by Pan et al. (2017) aimed to minimize the total weighted completion times on a parallel CAL with up to 18 different lines and 120 coils and achieved optimal solutions in under ten minutes.[38]

One similarity of the previously discussed studies is the deterministic characteristic of the used models. Therefore, some authors used dynamic approaches like a multi-agent system (MAS) to optimize dynamic scheduling problems, which consider randomness.[39] A MAS uses artificial intelligence and multiple agents or perspectives to solve a problem to provide flexible solutions.[40] Iannino et al. (2021) used both deterministic and dynamic models for scheduling. The authors used three different approaches to optimize a day's schedule with around 2100 coils in several iterations, with the objective to improve scheduling flexibility. For short-term planning with unstable circumstances, they used a MAS. The second approach was a deterministic, mixed integer linear program (MILP), as is used in this study. The third and last approach utilized a continuous flow model (CFM) for long-term scheduling, which uses a simplified model to schedule the manufacturing process over various time periods.[41] Iannino et al. (2021) summarized that all three approaches have their advantages and disadvantages and that they should be used complementarily rather than substitutionally to utilize each advantage in the right circumstances.

The second most important study about scheduling for this thesis is by Mujawar et al. (2012). The authors proposed a MILP for the minimization of both stringers and tardiness in terms of total time overdue per coil. Due to technical limitations, they were only able to solve the model with up to 15 coils, with a solving time of close to three hours. Because of these limitations, they leveraged two different heuristics, which were able to yield feasible solutions for up to 150 coils, but could only minimize the number of stringers introduced in the schedule and disregarded the tardiness.[42]

Nonetheless, the proposed mathematical model mimicked the real-world process to a higher degree than others. Because of this, Wegel et al. (2024) decided to base their study on the model proposed by Mujawar et al. (2012) and develop it further for better performance in the solving process.

3. Optimization model for Parallel Heterogeneous Annealing Line Scheduling

3.1. Problem description

As mentioned in section 2.1, the continuous annealing process is very complex, and many parameters have to be accounted for. One such parameter, that was briefly mentioned before, is the individual due date of each coil. These are necessary, to schedule the further downstream production stages, like the galvanization.[43] These schedules only have limited flexibility and to not put them at risk, the maximum number of delays has to be bound. This limitation is implemented through a service level, which value is relative to the number of coils processed. Thus, a greater instance is granted with a higher service level than a small or medium one.

Until now, only the specific characteristics of the coils and their processing modes have been considered. But the lines they are processed on have different characteristics themselves. Some lines may only be able to process coils with certain thicknesses and widths, while others may be able to process all coils, regardless of their characteristics. When creating a schedule for a continuous annealing line, the manufacturer thus also has to consider which coil can be processed on which of the parallel heterogeneous lines, therefore decreasing the amount of planning flexibility.

This results in a schedule that has to consider the different characteristics of the coils, their desired specifications in terms of strip speed and annealing temperature, the compatibility of said characteristics and processing modes of consecutive coils, as well as their compatibility with each processing line and their due dates, while it also has to comply with the service level and aims to minimize the number of used stringers. To be able to create such a complex schedule, Wegel et al. (2024) set four assumptions, regarding the release dates, internal service level, costs, and operating conditions.

To reduce complexity, it is first assumed that every coil is available and waiting for processing at the beginning of the schedule. Therefore, each coil could be the first to be processed in the schedule since they have no release dates.

The second assumption concerns the previously described internal service level. As already mentioned, the service level bounds the absolute number of delayed coils to a certain value that is relative to the instance size. But the internal service level also bounds the maximum delay that can occur in the schedule, since all coils have to be processed during it. This maximum delay depends on the maximum amount of time needed to process all coils in one schedule.

[35] Zhao and Yang, 2016, p. 3418
[36] Mirjalili and Gandomi, 2023, p. 393
[37] Gao et al., 2008, p. 1829-1833
[38] Zhang and Yang, 2014, p. 800-802
[39] Cowling et al., 2004, p. 178-188 and Ouelhadj et al., 2004, p. 161-172
[40] Balaji and Srinivasan, 2010, p. 1-2
[41] Iannino et al., 2021, p. 620-630
[42] Mujawar et al., 2012, p. 440-444
[43] Zhao and Yang, 2016, p. 3418

Table 1: Notation of the mathematical model.

Symbol	Meaning
	Indices and index quantities
$i, j \in 1, ..., \mathcal{N}$	Coils
$k \in 1, ..., \mathcal{K}$	Processing lines
$m \in 1, ..., \mathcal{M}_{ik}$	Feasible processing modes of coil i on line k
$n \in 1, ..., \mathcal{M}_{jk}$	Feasible processing modes of coil j on line k
	Parameters
d_i	Due date of coil i
p_{ikm}	Processing time of coil i on line k in mode m
α	Service level
M	Maximum duration of the schedule
c_{ijkmn}	Costs of adding a stringer between coils i in mode m and j in mode n on line k
t_{ijkmn}	Additional processing time of a stringer between coils i in mode m and j in mode n on line k
	Decision variables
$y_{ijkmn} \in \{0, 1\}$	Binary variable with value 1, if coils i in mode m and j in mode n are processed in sequence on line k, else 0
$\delta_{ijk} \in \{0, 1\}$	Binary variable with value 1, if coils i and coil j are processed in sequence on line k, else 0
$x_{ikm} \in \{0, 1\}$	Binary variable with value 1, if coil i is processed in mode m on line k, else 0
$s_i \geq 0$	Start date of coil i
$z_i \in \{0, 1\}$	Binary variable with value 1, if coil i is delayed, else 0

The third assumption made sets the costs of all coils to a fixed value. The reasoning behind this is that only the costs caused by the introduction of stringers should be considered since we cannot avoid the costs caused by the processing of coils. Due to CALs being quite stable and completely automated, Wegel et al. (2024) also assume that no randomness occurs during the process and therefore propose a deterministic model, in which all parameters are known a priori. This is the fourth and last assumption regarding the mathematical model.

Based on these assumptions and the aspects mentioned previously, the authors build a deterministic optimization model that aims to create an optimal, cost minimizing, and tardiness-bound schedule for parallel heterogeneous annealing lines, which is based on the model by Mujawar et al. (2012)[44] In the following sections, the notation of this mathematical model and the model itself will be explained, which will be followed by further extensions to it.

3.2. Notation

This section presents the notation of the mathematical model proposed by Wegel et al. (2024). The coils of flat steel are denoted by $\mathcal{N} = \{1, ..., N\}$ and the parallel continuous annealing lines they are processed on by $\mathcal{K} = \{1, ..., K\}$. Every coil $i \in \mathcal{N}$ has a specific due date $d_i > 0$. The maximum amount of acceptable delays, the service level, is denoted by $\alpha \in \mathbb{N}$.

Each coil i also has a set amount of different processing modes $\mathcal{M}_{ik}$ on each line $k \in \mathcal{K}$. The feasible process mode m of coil i on line k includes a specific annealing temperature and strip speed. If coil i can be processed in a specific mode on line k depends on the characteristics of the coil i and of the line k, as well as on the parameters of the processing mode itself. It is therefore possible, that no mode for the processing of coil i on line k exists. If coil i can be processed on line k in mode m, the processing time p_{ikm} can be derived.

Furthermore, coil i on line k could be succeeded by coil j, with $j \in \mathcal{N}$, in the feasible processing mode n, with $n \in \mathcal{M}_{jk}$. In this case, the compatibility of coil i in mode m and coil j in mode n regarding their width, thickness, annealing temperatures, and strip speeds has to be reviewed. If the coils and modes are incompatible with each other, a stringer has to be added between them. The additional costs caused by introducing a stringer into the schedule between coil i in mode m and coil j in mode n on line k are expressed by c_{ijkmn}, while the additional processing time is contained in t_{ijkmn}. If the coils and their respective modes are compatible, the additional costs and processing time will equal zero, since no stringer has to be introduced.

The processing schedule will be defined by the following decision variables. Each coil's start time is represented by $s_i \geq 0$, while its processing line and mode in the optimized

[44] Mujawar et al., 2012, p. 440

schedule are indicated by the binary variable $x_{ikm} \in \{0,1\}$ taking the value of one if coil i is being processed on line k in mode m or zero, if not. If the completion of coil i is behind the scheduled due date d_i, the binary variable $z_i \in \{0,1\}$ will equal one or zero, if coil i's processing was finished in time. The sequence of the schedule is indicated by the binary variable $\delta_{ijk} \in \{0,1\}$, with the value of one if coil i precedes coil j on line k or zero if this is not the case.

Additional information regarding their respective modes m and n is provided by the binary variable $y_{ijkmn} \in \{0,1\}$ with the value of one if coil i is being processed in mode m and is succeeded by coil j in mode n on line k or zero, if not.

3.3. Mathematical model

$$\min_{z,s,x,\delta,y} Z = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}_{ik}}\sum_{n\in\mathcal{M}_{jk}} c_{ijkmn}\cdot y_{ijkmn}$$

s.t.

$$s_i + \sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}_{ik}} p_{ikm}\cdot x_{ikm} \le d_i + z_i\cdot M \qquad \forall i\in\mathcal{N}, \quad (1)$$

$$s_i + \sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}_{ik}}\sum_{n\in\mathcal{M}_{jk}} (p_{ikm}+t_{ijkmn})\cdot y_{ijkmn} \le s_j + M\cdot(1-\sum_{k\in\mathcal{K}}\delta_{ijk}) \qquad \forall i,j\in\mathcal{N}, \quad (2)$$

$$\delta_{ijk} = \sum_{m\in\mathcal{M}_{ik}}\sum_{n\in\mathcal{M}_{jk}} y_{ijkmn} \qquad \forall i,j\in\mathcal{N}, \forall k\in\mathcal{K}, \quad (3)$$

$$\sum_{j\in\mathcal{N}}\sum_{n\in\mathcal{M}_{jk}} y_{ijkmn} \le x_{ikm} \qquad \forall i\in\mathcal{N}, \forall k\in\mathcal{K}, \forall m\in\mathcal{M}_{ik}, \quad (4)$$

$$\sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}_{ik}} y_{ijkmn} \le x_{jkn} \qquad \forall j\in\mathcal{N}, \forall k\in\mathcal{K}, \forall n\in\mathcal{M}_{jk}, \quad (5)$$

$$\sum_{k\in\mathcal{K}}\sum_{j=1}^{\mathcal{N}+1}\delta_{ijk} = 1 \qquad \forall i\in\mathcal{N}, \quad (6)$$

$$\sum_{k\in\mathcal{K}}\sum_{i=0}^{\mathcal{N}}\delta_{ijk} = 1 \qquad \forall j\in\mathcal{N}, \quad (7)$$

$$\sum_{j=1}^{\mathcal{N}+1}\delta_{0jk} = 1 \qquad \forall k\in\mathcal{K}, \quad (8)$$

$$\sum_{i=0}^{\mathcal{N}}\delta_{i(\mathcal{N}+1)k} = 1 \qquad \forall k\in\mathcal{K}, \quad (9)$$

$$\sum_{j=0}^{\mathcal{N}}\delta_{jik} = \sum_{j=1}^{\mathcal{N}+1}\delta_{ijk} \qquad \forall i\in\mathcal{N}, \forall k\in\mathcal{K}, \quad (10)$$

$$\sum_{i\in\mathcal{N}} z_i \le \alpha, \quad (11)$$

$$s_i \ge 0 \qquad \forall i\in\mathcal{N}, \quad (12)$$

$$x_{ikm}, \delta_{ijk}, y_{ijkmn}, z_i \in \{0,1\} \qquad \forall i,j\in\mathcal{N}, \forall k\in\mathcal{K}, \forall m\in\mathcal{M}_{ik}, \forall n\in\mathcal{M}_{jk}. \quad (13)$$

The main objective of this mathematical model is the minimization of costs caused by the introduction of stringers. Therefore, the objective function minimizes the sum of products of c_{ijkmn} and y_{ijkmn}. By multiplying c_{ijkmn} and y_{ijkmn} the cost of a stringer is only regarded if coil i in mode m and coil j in mode n on line k are not compatible with each other and are processed in sequence in the optimized schedule, thus requiring a stringer. If they are not processed in sequence or if they are compatible with each other or both, the value of the product will equal zero since y_{ijkmn} or c_{ijkmn} or both will equal zero, respectively. By taking the sum over all coils, lines and modes, all introduced stringers in the schedule are considered.

To mimic the real-world process and maintain consistency, several constraints are necessary and will be discussed in the following. Constraints (1) and (2) assure time consistency, while constraints (3)-(5) ensure the consistency of decision variables that indicate the same information. The consistency of the processing sequence in the schedule is established by the constraints (6)-(10). These are followed by three additional constraints, that regulate general aspects of the mathematical model and the schedule.

The first constraint (1) takes the tardiness of the schedule into account. The time of completion of coil i is calculated by adding its processing time p_{ikm} in mode m on line k to its start date s_i. This completion date has to be smaller or equal to its due date d_i for the coil to be finished processing in time. In this case, z_i would equal zero due to the restriction of the number of delayed coils by the service level and the possibility that the delay could be used somewhere else in the schedule to further minimize the number of introduced stringers. But if the completion date is after the due date, z_i has to equal one to not violate the equation, since the time of completion of coil i will be greater than its due date d_i. Additionally, the product of z_i and M assures that coil i has to be processed sometime in the schedule since M is defined as $max_k(\sum_{i\in\mathcal{N},\mathcal{M}_{ik}\neq\emptyset} max_{j,m,n}(p_{ikm}+t_{ijkmn}))$, which is the maximum amount of time necessary to process all coils in the schedule. This constraint has to be set for every coil, as every coil has a due date and could be delayed.

Constraint (2) considers the time sequences of processing in the schedule. The left side represents the earliest time a succeeding coil j could be processed after the processing of the preceding coil i was finished. This is achieved by adding the processing time p_{ikm} of coil i in mode m on line k and the possible additional processing time of a stringer t_{ijkmn}, in case of incompatibility of coil i in mode m and coil j in mode n, to the start date of coil i. Processing of the succeeding coil j can only start after the processing of coil i or that of

the stringer is completed. Thus, s_j has to be greater or equal to that time of completion. In the case that coils i and j are processed in sequence on line k, δ_{ijk} would equal one, and thus, M would not be added to the right-hand side. But as this constraint holds for all coils, it could compare two coils with each other, that are not processed in sequence. To prevent this comparison to have an effect on the solution, M is added to s_j, since in this case, δ_{ijk} will equal zero. Thus, the right side will always be greater than or equal to the left side since $M \leq s_j + M$ and $s_i + (p_{ikm} + t_{ijkmn}) \cdot y_{ijkmn} \leq M$ in every scenario, because of the nature of M described previously.

Constraint (3) assures consistency throughout the variables δ_{ijk} and y_{ijkmn}. Both variables indicate the sequence of coils i and j, as well as the line k they are processed on, and therefore should be equal to each other with the same coils i, j and line k. Since y_{ijkmn} also indicates the processing modes of both coils and cannot represent a binary value without this information, the sum of y_{ijkmn} over all modes has to be taken for this constraint to be effective. This constraint also reassures, that the coils are only processed in one mode and not in multiple, hence in this case, two y_{ijkmn} would equal one with the same coils and line. Therefore, δ_{ijk} would have to equal two, which is impossible due to the binary nature of δ_{ijk}.

Constraint (4) contributes to the decision variable consistency as well. Both variables y_{ijkmn} and x_{ikm} indicate the processing line k and the processing mode m of coil i. Hence, they should be equal to each other for every coil, mode, and line. But in the case that coil i is the last coil that is being processed on line k, y_{ijkmn} would equal zero since y_{ijkmn} does not take the coils $i \in \{0, \mathcal{N}+1\}$ into account and could therefore not be succeeded by another coil j. To not violate this restriction in this case, y_{ijkmn} must only be less or equal to x_{ikm}. Therefore, with x_{ikm} equalling one, y_{ijkmn} could be either one or zero. Furthermore, by taking the sum of y_{ijkmn} over all coils and modes, the constraint prevents coil i from being succeeded by multiple different coils in multiple different modes and limiting this number to the value of x_{ikm}, which is either one or zero due to its binary nature.

Therefore, coil i can only be succeeded by a single coil j in a single mode n on line k, if i is being processed on line k in mode m or by none if it is not. This holds for all coils, lines, and modes.

The following constraint (5) is essentially the same as (4) but for the succeeding coil j. Hence, it prevents coil j from succeeding multiple coils in multiple modes by limiting this number to x_{jkn}. It has to be noted, that the indices of x_{jkn} in this constraint differ from the indices of x_{ikm} in the other constraints since this constraint should only restrain the succeeding coil j. Additionally, y_{ijkmn} must only be less or equal to x_{jkn}, since all y_{ijkmn}'s would be zero if j would be the first coil to be processed on line k, which would violate the constraint. This is because this constraint only accounts for coils in $\mathcal{N}$ and thus the first coil on line k cannot be preceded. This restriction applies to all succeeding coils on every line in every processing mode.

Constraint (6) restricts the number of coils that succeed coil i on any line to one. This prevents coil i from being processed and preceded by several coils on different lines. If this would be the case, the sum of δ_{ijk} over all lines and coils in $[1, \mathcal{N}+1]$ would be greater for coil i than one, thus violating the restraint. If coil i is the last coil to be processed on line k, it could not be succeeded by any coil, resulting in δ_{ijk} equalling zero. Therefore, a virtual coil $\mathcal{N}+1$ has to be introduced so that in the case mentioned above, coil i would not violate this constraint. This restriction applies to every coil.

Constraint (7) is similar to constraint (6), as this constraint restricts the number of preceding coils of coil j on all lines to one and therefore prevents coil j from being processed and preceded more than once during the schedule. The extension of $\mathcal{N}$ by zero can be explained with the constraint (8). Here, i is set to the value zero. This virtual coil zero marks the beginning of the schedule on every line k. Since this coil is not being processed, it is not included in most other constraints. But since coil zero is the first on every line k, it has to be succeeded by one coil $j \in [1, \mathcal{N}+1]$. Therefore, constraint (8) ensures that one coil j is the first one to be processed on line k. No real coils are processed on line k during the schedule if coil $\mathcal{N}+1$ succeeds coil zero. This holds for every line because every line has to have a virtual first coil that is being succeeded by another coil, virtual or real.

Constraint (9) on the other hand ensures that one coil $i \in \{0, \mathcal{N}\}$ has to be the last real coil to be processed on line k. To mark this coil as the last coil to be processed on line k, it will be succeeded by another virtual coil $\mathcal{N}+1$. As on every line k there has to be a last coil to be processed, this holds for every line.

The last sequence consistency constraint, constraint (10), assures that every real coil i has to have a predecessor and successor coil j on the line k it is being processed on. But it also accounts for the extreme case, that a processing line k processes no real coil, which is why it considers coil zero as a predecessor on the left-hand side and coil $\mathcal{N}+1$ as a successor on the right-hand side. Additionally, this considers that one real coil has to be the first and another has to be the last coil if real coils are processed on the line, as described in constraints (8) and (9). This restriction applies to every line.

Hence, the introduction of the virtual coils zero and $\mathcal{N}+1$ prevents δ_{ijk} from equalling zero in the cases that coil j is the first or coil i is the last coil to be processed on line k, which would violate the sequence consistency constraints.

The eleventh constraint (11) restricts the maximum amount of delayed coils to the service level α. As this applies to the entire schedule, the sum of z_i over all coils has to be taken. Since processing only starts at the beginning of the schedule, no start date s_i can take a value of less than zero. This is ensured by constraint (12). As every coil needs a start date, since every coil has to be processed at some point during the schedule, this holds for every coil. The last constraint (13) ensures that the binary variables x_{ikm}, δ_{ijk}, y_{ijkmn}, and z_i only equal the binary values of zero or one and also indicates which index is the element of what parameter.

3.4. Expansions of the model

3.4.1. Release dates

The first expansion tackles the assumption of no coil-specific release dates made by Wegel et al. (2024), which was explained in section 3.1. To mimic a more realistic process, coil-specific release dates were added to the model with the new parameter r_i. This parameter holds the specific release date of each coil i, with the release dates being distributed with the same distribution as the urgency of the coils, but with zero as the lower and a as the upper bound. Therefore, in the case of $Very\ high$ UC, the release dates are distributed via the uniform distribution $U(0, a)$. The UC and other factors will be explained in section 4.1 For the release dates to be considered in the model, the following constraints had to be added:

$$s_i \geq r_i \qquad \forall i \in \mathcal{N} \quad (14)$$

$$r_i \geq 0 \qquad \forall i \in \mathcal{N} \quad (15)$$

Constraint (14) restricts the start time of coil i, s_i, from being earlier than its release date r_i, while (15) prevents the release date from being earlier than the start of the schedule. These hold for all coils.

This is only a minor extension of the model. The two major extensions will be described in the following sections.

3.4.2. Minimization of stringer use and tardiness

Since both tardiness and costs caused by stringer introduction play a pivotal role in the scheduling of parallel heterogeneous annealing lines (PHALs), Wegel et al. (2024) proposed an expansion of the mathematical model. This expansion modifies the objective function in such a way, that the sum of introduced stringers and the absolute number of delays are minimized, as depicted below. Because of this, the service level α and constraint (11) from the mathematical model in section 3.3 are not necessary anymore and are removed. To account for different weights of tardiness and stringer costs, the two sub-objectives are multiplied with new parameters. While r_str sets the weight of the stringer costs, r_tar does the same for the tardiness, expressed by the sum of delayed coils, with $r\_str + r\_tar = 1$. To achieve this, an additional restriction (16) was introduced. This constraint ensures that the total weight of tardiness and stringer costs does not differ from 100%.

$$\min_{z,s,x,\delta,y} Z = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}_{ik}}\sum_{n\in\mathcal{M}_{jk}} c_{ijkmn}\cdot y_{ijkmn}\cdot r\_str + \sum_{i\in\mathcal{N}} z_i \cdot r\_tar$$

$$r\_str + r\_tar = 1 \quad (16)$$

3.4.3. Minimization of stringer use and due date deviation

The second major extension of the model is an enhancement of the first extension by not only taking tardiness into account but also the earliness of the schedule. If products are produced before the due date, they have to be stored in warehouses, which leads to so-called inventory holding costs. They are not only caused by the storage and handling of the inventory, but also by insurance and other factors.[45]

Thus, there is also an incentive to minimize earliness in production processes. To minimize both the use of stringers and the deviation from the due date, the objective function had to be altered accordingly.

$$\min_{z,s,x,\delta,y} Z = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}_{ik}}\sum_{n\in\mathcal{M}_{jk}} c_{ijkmn}\cdot y_{ijkmn}\cdot r\_str + \sum_{i\in\mathcal{N}}(v_i\cdot r\_v + e_i\cdot r\_e)\cdot r\_dd$$

The first part of the objective function is the same as in the first major extension. But the second part introduces two new variables and three new parameters. The two new continuous variables e_i and v_i can be explained by describing the new first constraint:

$$s_i + \sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}_{ik}} p_{ikm}\cdot x_{ikm} + e_i = d_i + v_i \quad \forall i \in \mathcal{N} \quad (17)$$

These new continuous variables, with a lower bound of zero, work in such a way, that only one of them can have a value higher than zero for coil i. If coil i in mode m on line k was finished processing before its due date d_i, $s_i + p_{ikm}\cdot x_{ikm} < d_i$. This would violate the constraint since both sides have to be equal to each other. Therefore, e_i will take the value of $d_i - (s_i + p_{ikm}\cdot x_{ikm})$, which represents the positive time left until the due date and therefore the earliness of coil i. In this case, v_i will equal zero, since both e_i and v_i are to be minimized. A v_i greater than zero would lead to $d_i + v_i - (s_i + p_{ikm}\cdot x_{ikm}) > d_i - (s_i + p_{ikm}\cdot x_{ikm})$, therefore increasing both v_i and e_i and thus the objective value.

The opposite case, in which coil i is delayed, is similar. Now, $s_i + p_{ikm}\cdot x_{ikm} > d_i$. To not increase this difference and hence the objective value, e_i will equal zero. Because of the necessary equality of both sides, v_i will equal $s_i + p_{ikm}\cdot x_{ikm} - d_i$, representing the positive time that coil i is delayed.

In the third and last case, in which the processing of coil i is finished processing exactly on time, $s_i + p_{ikm}\cdot x_{ikm} = d_i$. Since e_i and v_i are to be minimized, both variables will equal their lower bound zero. In the altered objective function, e_i and v_i are multiplied with the parameters r_v and r_e, to provide a possibility for weighing the costs of tardiness and earliness. The third and last parameter r_dd weighs the costs of due date deviation against the weight of the stringer costs

[45] Durlinger, 2015

r_str. Since weights cannot differ from 100 %, the following constraints have to hold:

$$r\_str + r\_dd = 1 \tag{18}$$

$$r\_v + r\_e = 1 \tag{19}$$

In addition, the variable z_i is removed from the model because absolute tardiness is not relevant anymore.

4. Computational study

4.1. Structure of the study and instances

After describing the industrial process and the mathematical model for creating and optimizing a schedule for CALs in the before sections, this section will outline the structure of the numerical study, which will be followed by the results and the interpretations of it.

The computational study makes use of four main factors proposed by Wegel et al. (2024), with each having five different levels ranging from $Very\ low$ to $Very\ high$, which are depicted in table 2. The heterogeneity of coils (HC) describes the possible range of each characteristic of the coils and their distribution among them. This includes the width, thickness, and length, as well as the mid-temperature and mid-speed, and the symmetric intervals around them. These will be used to assign physical properties to each coil during the computational study, as well as to define the feasible processing modes. The distributions of coil characteristics differ in the five levels. $TR(a, b, b)$ and $U(a, b)$ represent a triangular and uniform distribution, respectively.

As an example, the width in the $Very\ high$ level of HC will be distributed by the uniform distribution $U(60, 100)$, since 60 cm is the lower bound and therefore equal to a and b, the upper bound, equals 100 cm. Hence, the width of each coil will be between 60 cm and 100 cm, with all values having an equal probability. This aspect of the uniform distribution increases the heterogeneity and difficulty of finding a stringer-free solution since the characteristics will be evenly spread between a and b, thus decreasing the probability of compatibility between different coils. With a triangular distribution $TR(a, b, b)$, however, most values will be close to b, hence increasing the probability of compatibility. This is the reason the distribution changes from a triangular to a uniform distribution with increasing intensity of HC.

Each coil receives a mid-temperature and -speed through the distribution of $midTemp$ and $midSpeed$ in steps of ten °C and ten m/min respectively, which represent the centers of symmetric intervals. The widths of these intervals are determined by $\Delta temp$ and $\Delta speed$. The ranges of these intervals indicate the different feasible annealing temperatures and strip speeds, under which the coil can be processed. A combination of feasible annealing temperature and strip speed is a feasible processing mode for that specific coil. In order to limit the number of processing modes, to 48 in total, the annealing temperature and strip speed in processing modes only vary in increments of ten °C and 20 m/min, respectively. To give an example, coil i in the *Basecase* scenario has a mid-temperature of 680 °C and a mid-speed of 510 m/min. Thus, the intervals with $\Delta temp = 20$ and $\Delta speed = 20$ are [670 °C; 690 °C] and [500 m/min; 520 m/min]. The resulting feasible processing modes for coil i are [(670, 500), (670, 520), (680, 500), (680, 520), (690, 500), (690, 520)], with the first entry being the annealing temperature and the second being the strip speed. The maximum number of feasible processing modes per coil is therefore six.

This holds for every level of HC, except for the $Very\ high$ level, due to the narrower interval widths. In this case, the annealing temperature and strip speed can only vary by + ten °C and + ten m/min, respectively. The intervals in the example from above are therefore [680 °C; 690 °C] and [510 m/min; 520 m/min], resulting in only two possible processing modes [(680, 520), (690, 520)]. Hence, each coil has two to six different processing modes, depending on the level of HC. An extension to the HC is the distribution of the length of coils. This results in far more heterogeneous processing times p_{ikm} since the processing time of coil i on line k in processing mode m is defined as the length of coil i divided by the strip speed of mode m. This minor extension will be included in all models used in the computational study.

The urgency of the coils (UC) defines the size of the time horizon that is used for the distribution of the individual due dates. As with HC, lower levels distribute the due dates via a triangular distribution, while higher levels distribute them with a uniform distribution or a mixture of both, thus spreading the due dates more evenly on the time horizon. Adding to this, the uniform distribution distributes more due dates close to the start of the schedule than the triangular distribution. By additionally decreasing the value of a by increasing the UC, the earliest due date will also be decreased, since both the triangular distribution $TR(a, b, b)$ and the uniform distribution $U(a, b)$ are using a as the earliest possible due date. These factors restrict the feasible solution space and could thereby be detrimental to finding an optimal, stringer-free schedule. The latest possible due date b is a function of $\mathcal{N}$, which is defined as $f(\mathcal{N}) = avg\_proc\_time \cdot \mathcal{N}/\mathcal{K}$, with avg_proc_time being the sum of coil i's length divided by the strip speeds of feasible process modes $\mathcal{M}_{ik}$ over all coils and processing lines, which is then divided by the product of $\mathcal{N}$ and $\mathcal{K}$. That the values of a and b both depend on the number of coils $\mathcal{N}$ assures that the time horizon is adequate to the instance size and does not lead to infeasibility with a higher number of $\mathcal{N}$.

Process flexibility (PF) defines the compatibility between consecutive coils in terms of their thickness, width, annealing temperature, and strip speed. Higher PF allows for a higher difference in these characteristics, thus increasing the scheduling flexibility. To give an example, the difference in widths between two consecutive coils in the *Basecase* scenario can be up to five cm, without having to introduce a

[46] Wegel et al., 2024, p. 20

Table 2: Parameterization of the different factors and levels.[46]

Factor \ Level	Very low	Low	Basecase	High	Very high
Heterogeneity coils					
distribution	TR(a,b, (a+b)/2)	TR(a,b, (a+b)/2)	50%TR(a,b, (a+b)/2), 50%U(a,b)	U(a,b)	U(a,b)
$width(a,b)$ [cm]	70, 90	60, 100	60, 100	60, 100	60, 100
$thickness(a,b)$ [cm]	0.15, 0.35	0.1, 0.499	0.1, 0.499	0.1, 0.499	0.1, 0.499
$length(a,b)$ [km]	15, 16	14, 17	14, 17	14, 17	14, 17
$midTemp(a,b)$ [°C]	690, 720	680, 730	680, 730	680, 730	680, 730
$\Delta temp$ [°C]	20	20	20	20	10
$midSpeed(a,b)$ [m/min]	520, 580	510, 590	510, 590	510, 590	510, 590
$\Delta speed$ [m/min]	20	20	20	20	10
Urgency of coils					
distribution	TR(a,b,b)	TR(a,b,b)	TR(a,b,b)	50%TR(a,b, (a+b)/2), 50%U(a,b)	U(a,b)
a [min]	$\lvert\mathcal{N}\rvert \cdot 2$	$\lvert\mathcal{N}\rvert \cdot 1.5$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{N}\rvert$
b [min]	$f(\mathcal{N})+a$	$f(\mathcal{N})+a$	$f(\mathcal{N})+a$	$f(\mathcal{N})+a$	$f(\mathcal{N})+a$
Process flexibility					
$\Delta width$ [cm]	>5	>5	>5	>10	>15
$\Delta thickness$ [mm]	>0.05	>0.1	>0.1	>0.1	>0.2
$\Delta temp$ [°C]	>0	>0	>10	>10	>20
$\Delta speed$ [m/min]	>0	>20	>20	>20	>20
Stringer processing					
$time$ [min]	0	1.5	3	10	15

stringer between them. If the difference extended five cm, a stringer would have to be added to bridge the differences. Thus, higher PF minimizes the need for stringers and vice versa. Please note, that the length has no PF since it does not have an impact on the welding or process mode compatibility.

The stringer processing time (SPT) defines the amount of time needed to process a stringer if introduced into the schedule. A high SPT diminishes flexibility in terms of time constraints and tardiness. There were no different stringer costs added to the base model since these are the only costs that are being considered. Therefore, different stringer costs would not affect problem-solving in any way. However, the extensions include relative stringer costs through r_str, since the minimization of stringers is only one of two different sub-objectives.

Since the optimization model was built for a PHAL, three lines with different specifications were chosen for the numerical studies, if not stated otherwise. For this study, the heterogeneity of the lines will be limited to the width of the coils the line can process. The first line can process all coils, regardless of their width. Only narrow coils can be processed on the second line, while only wide ones can be processed on the third one. This follows the line specifications selected by Wegel et al. (2024), which were adopted for this study, as well as the maximum solving time of 720 seconds. This time limit is not a hard limit, hence some optimizations might slightly exceed it. The exact line specifications depend on the chosen level of HC. They are configured such they can only process coils that are five cm above or below the width that represents the center of the width range. As an example, with $High$ HC, the second line can only process coils with a width of less than or equal to 75 cm.

First, the tuning parameterization will be discussed. Tuning influences the behavior of the solver and thus could increase its performance.

The second part consists of a study on different instance sizes and the *Basecase* scenario for each factor. For each instance size, the service level will vary between low, medium and high, with low representing a service level of zero %, medium representing ten % and high 20 %. Each optimiza-

tion in the numerical study will be conducted with the same ten seeds, to ensure a certain degree of comparability between the different instant sizes and sections.

A seed determines the outcome of randomness, like the distribution of coil characteristics or due dates. Therefore, by increasing the number of coils, new coils will be added to the already existing ones. Hence, two calculations with the same parameters and the same seed yield nearly the same result, but the optimization itself could still be affected by minor randomness.[47] This assures that differences in solution characteristics between two calculations with different parameters, i.e., different service levels, are mostly caused by the different parameters and not by randomness.

This section of the study aims to uncover possible trends that occur with certain parameterizations. The solutions and performance achieved with the default solver will be compared with the tuned solver. At the end of this part, an instance size will be chosen that will be used for the rest of the study.

For the third part of the numerical study, the Best-case and Worst-case scenarios are defined by using the different factors in table 3.

Table 3: Parameterization of the Worst- and Best-case scenario.

Scenario	HC	UC	PF	SPT
Best-case	Very low	Very low	Very high	Very low
Worst-case	Very high	Very high	Very low	Very high

The results of these extreme cases will be compared with the *Basecase* scenario to determine the impact of the four factors. Additionally, the factors PF and HC will be altered in the Best- and Worst-case scenarios, respectively, to further investigate their influence. The levels used for the factors are *Very low*, *Basecase* and *Very high*.

The fourth section will explore the impact of changing factor and model specifications on the solution. First, the number of processing lines in the *Basecase* scenario will be altered, as depicted in table 4.

Table 4: Parameterization of processing line altering cases.

Case	Alteration
Addition	Addition of processing line, that can process all coils
Removal I	Removal of processing line, that can only process wider coils
Removal II	Removal of processing line, that can only process narrower coils

[47] Miltenberger, 2023b

The results will be compared to the *Basecase* scenario with the default number of processing lines, with emphasis on the number of introduced stringers and tardiness. Secondly, the a and b parameters used in the UC will be reduced to study the impact of earlier due dates on the feasibility of the scheduling. For each of the cases, the *Basecase* scenario with a *Very high* UC will be used.

The last section compares some aspects of the solutions of the original model with the solutions of the expanded models. The first expanded model to be studied will be the model which aims to minimize both the number of stringers and absolute tardiness. In addition to varying weights of stringer and tardiness costs, the UC will be altered to study its influence on the solution. This results in the cases in table 6 that will be explored with a *Very low*, *Basecase* and *Very high* UC.

These cases are chosen to accommodate for different weighing preferences and since it is assumed, that under no circumstances one of the costs could be neglected. Thus, no case only considers stringer or tardiness costs.

The second expanded model, which aims to minimize stringer, earliness and tardiness costs, will also be studied using the different cases in table 7. Different stringer and due date deviation weights will be combined with different earliness and tardiness weights, which are the components of the due date deviation.

The entire computational study consists of 1180 individual runs, with an approximate total computation time of 140 hours and was conducted on a computer with a Ryzen 7 3700X CPU, 16 GB of 3200MHz memory, and Windows 10. The CPU frequency could have differed due to changing room temperature and longer computation sessions, thus slightly affecting the performance of some optimizations.

4.2. Results and interpretations

4.2.1. Tuning of the solver

In the following, the tuning for each instance size will be briefly explained, which was defined through trial and error. These runs and optimization times were not included in the above-mentioned numbers of 1180 runs and 140h. It has to be noted, that the tuning parameters of a lower instance size are also applied to higher instance sizes, if not stated otherwise.

The two smallest instance sizes ten and 20 coils received only minor tuning. The so-called *PreSolve* was set to two, which is an aggressive setting. *PreSolve* tightens the model before solving it by, as an example, removing redundant and inactive constraints, thus increasing the probability of finding a feasible solution and the performance of the solver in general. The duration of *PreSolve* was cut short with the *PrePasses* parameter since smaller instances already have smaller models and thus find feasible solutions fast, as will be explained later in the second section. Another parameter, *Cuts*, was set to conservative, which behaves like *PreSolve* by tightening the model. The difference between *PreSolve*

Table 5: Parameterization of due date altering cases.

Case	Alteration	a [min]	b [min]
100 %	Very high UC	40	1306
75 %	Very high UC and reduction of a and b by 25%	30	979.5
50 %	Very high UC and reduction of a and b by 50%	20	653
25 %	Very high UC and reduction of a and b by 75%	10	326.5

Table 6: Parameterization of cases with different stringer and tardiness weights.

Case	Stringer weight [%]	Tardiness weight [%]
1	80	20
2	65	35
3	50	50
4	35	65
5	20	80

Table 7: Parameterization of cases with different stringer, due date deviation, earliness and tardiness weights.

Case	Stringer:Due date deviation weight [%]	Earliness weight [%]	Tardiness weight [%]
	65:35		
1		80	20
2		65	35
3		50	50
4		35	65
5		20	80
	50:50		
6		80	20
7		65	35
8		50	50
9		35	65
10		20	80
	35:65		
11		80	20
12		65	35
13		50	50
14		35	65
15		20	80

and Cut is that $Cuts$ tightens the model during the solving process and not before it, as $PreSolve$ does.[48]

The $PrePass$ parameter will be set to automatic or negative one when solving instances with more than 20 coils, allowing for a longer $PreSolve$ and thus more tightening. Additionally, Gurobi-intern heuristics are maximized through the $heuristics$ parameter. Gurobi has many types of complex heuristics that can improve the ability to find a feasible solution significantly. By increasing the $heuristics$ parameter, the solver will spend more time on heuristics. The reason for this aggressive parameterization will be discussed in the next section regarding figure 2. Another tuning parameter, the $MIPFocus$, was modified with instances of 50 coils and higher. It influences, how much time will be spent on finding feasible solutions and on the optimality of the solutions. The $MIPFocus$ was parameterized to one to focus on finding feasible solutions rather than proving their optimality with the MIPGap. The MIPGap is a qualifier of a feasible solution, that only exists in Gurobi with MIPs. It is a relative value that represents the minimal optimality of the solution and is defined as $|ObjBound - ObjVal|/|ObjVal|$. To explain the MIPGap, the solving process of Gurobi has to be briefly explained first.[49]

For MILPs, Gurobi uses the so-called branch and bound approach. The solving process starts by solving the model without its integrality restrictions. If the resulting solution does not violate any restrictions, the solving stops since the solution is optimal. But in most cases, some variables of the solution violate constraints. In this case, one of these variables will be chosen as a branching variable and the infeasible solution becomes a node, from which branches extend. These branches are MIPs, that are more restricted in the possible values of the branching variable. After solving these MIPs, they can either generate a feasible solution or become a new node. This process will then be repeated until either a feasible solution was found or it was proven that there are no feasible solutions in this branch. This definitive end of a branch is called a leaf, but all yet unexplored branches are also leaves. If the first or a new best solution was found, it becomes the incumbent solution. The objective value of this incumbent solution is the $ObjVal$ mentioned above in the formula for the MIPGap. The $ObjBound$, also called the best bound, is the lowest or highest value that the solver approximates can be achieved by taking the lowest or highest

[48] Gurobi, n.d.-c and Gurobi, n.d.-b
[49] Miltenberger, 2023a and Gurobi, n.d.-b

Table 8: Tuning parameterization for each instance size.

Parameter \ # coils	10	20	30	40	50	60	70
Cuts	1	1	1	1	1	2	2
PrePasses	1	1	-1	-1	-1	-1	-1
PreSolve	2	2	2	2	2	2	1
Heuristics	-	-	1	1	1	1	1
MIPFocus	-	-	-	-	1	1	1
NodefileStart	-	-	-	-	-	-	0.5
Threads	-	-	-	-	-	-	8

value of all the different leaves, depending on the direction of the optimization. For a minimization model, which is used in this study, it will take the lowest value. This best bound can change throughout the solving process through new information about the feasibility of a solution with the best bound as an objective value. As an example of a minimization problem, if the solver notices that the best bound cannot be achieved in a feasible solution, it could raise the best bound to the second-lowest value.[50]

The MIPGap, therefore, is the absolute difference between the objective value of the best already found solution and the objective value of the best solution the solver approximates could be possible, relative to the former. Hence, the MIPGap represents the maximum relative improvement the solver approximates could be possible over the current incumbent solution.[51] Another function of MIPGap is, that the solver will terminate the solving process if the MIPGap of one solution is equal to or lower than the MIPGap that was set as a threshold, with the default threshold being close to zero %, or if the difference between the lower and upper bound is less than the MIPGap multiplied with the upper bound.[52]

For optimizations with 60 coils, the *Cuts* parameter was set to two, aggressive, to improve the tightening of the model. The number of out-of-memory errors increased dramatically with optimizations of instance sizes with 70 coils, which will be discussed later. To avoid this error, the number of CPU threads was limited to eight by using the parameter $Threads$ and the RAM usage was reduced by increasing $NodefileStart$ to 0.5. Higher thread counts generally lead to higher performance, but also demand more memory, which is why the amount of available threads for Gurobi was limited. Additionally, the $NodefileStart$ parameters relieves the RAM by storing data on the disk. Both of these parameters thus decrease RAM usage and out-of-memory errors, while also decreasing performance, which is why they have to be fine-tuned to not be detrimental to the optimization process.[53]

4.2.2. Impact of increasing instance size

Before inspecting all the data from the first section of the numerical studies, figure 2 has to be considered first. The figure depicts the number of optimizations per instance size and per service limit that did not result in a feasible solution.

It can be observed, that this number rises with the increasing instance size and that this trend is weaker in the tuned model and with higher service levels. As previously described, α determines the maximum amount of delayed coils relative to $\mathcal{N}$. Therefore, an optimization with a higher α is less restrained and has higher scheduling flexibility than one with a lower service level. This data suggests, that due to the higher scheduling flexibility, optimizations with higher service levels can solve for a feasible solution more easily. But it is important to note that this data includes three different reasons for the failed optimization.

The first reason is, that the Gurobi solver could not find a feasible solution during the 720 seconds time limit, thus aborting the solving process. As previously described, this is affected by the service level. Most unsuccessful optimizations occurred because of this reason. The second reason was an out-of-memory error. It occurs if the Gurobi solver demands more RAM than is physically available. This terminates the solving process, without finding a feasible solution prior in all cases. This happened only with greater instance sizes, especially with 70 coils. But the two failed optimizations at an instance size of only 20 coils happened due to a different reason. These optimizations failed due to the infeasibility of the model, in both the default and the tuned model. Since this only happened with a low service level, it suggests that these two seeds cannot be optimized for an instance size of 20 coils without delay. The most probable cause is a combination of low compatibility due to high coil characteristic heterogeneity, early due dates d_i, and high coil processing times p_{ikm}.

Regardless of the reason, this results in a decreasing number of data points for greater instance sizes. Hence, the validity of trends observed in this data could be affected by a

[50] de Harder, n.d. and Gurobi, n.d.-c
[51] de Harder, n.d.
[52] Miltenberger, 2023a and Gurobi, n.d.-b
[53] Gurobi, n.d.-b and Gurobi, n.d.-a, p. 821, 807

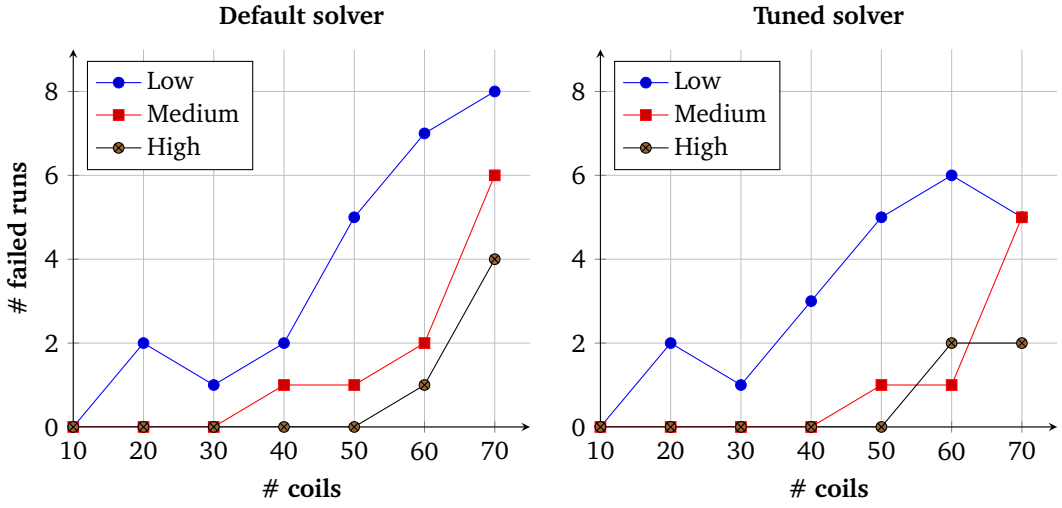

Figure 2: Number of optimizations per $\mathcal{N}$ and service level that yielded no feasible solution with the default and tuned solver.

selectivity effect, in which only optimizations of such seeds yield feasible solutions, that are easier to solve. This could be due to a more homogeneous distribution of coil characteristics, later due dates, and many other factors. Additionally, these seeds do not have to be easier to solve than others for every instance size since additional coils can, as an example, have earlier due dates and thus decrease the scheduling flexibility of the optimization. This notion has to be considered throughout this section of this study.

As depicted in figure 2, optimizations with the tuned solver yielded a feasible solution more often than with the default solver, with 40 failed optimizations with the default and only 33 unsuccessful optimizations with the tuned solver, with a total amount of 210 optimizations per solver. The impact of the tuning seems to rise with the increasing number of coils. This is due to the fact, that with higher instance sizes, more tuning was applied since these had a higher ratio of failed optimizations. After discussing the problem of data density, the characteristics of the solutions will be investigated in the following.

Figure 3 displays that the average number of introduced stringers varies between different instance sizes and service levels but generally seems to decline with the increasing number of coils. The tuning of the solver seems to have no significant effect on the quality of the solution since the objective values yielded by both solvers are mostly similar and only vary by a small amount. This was to be expected because the tuning mainly focussed on improving the solver's performance and not the solutions yielded by it.

But it can be observed, that allowing delays and thus increasing the scheduling flexibility typically results in more optimal solutions regarding stringer introduction. Optimizations with a service level of 20 % and hence the highest scheduling flexibility generally yielded the best solutions concerning stringers but differed only slightly from solutions obtained with a lower service level of ten %. At 70 coils, the optimization of a particular seed yielded an exceptionally high solution with both the default and the tuned solver.

Table 9: Results of seed 100 in optimizations with the tuned and the default solver, 70 coils and a high service level.

	# stringers	# solutions	MIPGap	Tardiness
Tuned solver	67	1	0.970	5
Default solver	65	1	0.985	14

As shown in table 9, these extraordinarily high objective values were the only feasible solution the solvers were able to calculate in time. The high MIPGap of both solutions suggests, that the optimality of these solutions either could not be proved in time or that they are not optimal, with the latter being the more plausible explanation when regarding the high objective values. The most significant difference between these solutions is the tardiness, which is much higher with the default solver.

As previously described, the tuning should have little impact on the solutions themselves. This, combined with the fact, that it was the only feasible solution found, suggests that this more optimal solution, regarding tardiness, is a result of randomness. Even though these solutions could be characterized as outliers, they were still included in figure 3 because they represent feasible solutions and could technically be implemented, despite being probably insensible businesswise.

More exceptions to the downward trend are the solutions yielded at an instance size of 20 coils across all service levels. This is the same instance size that resulted in infeasibility with two seeds, as described at the beginning of the section. The objective values of nearly every seed across all service levels increased when transitioning from ten to 20 coils, and

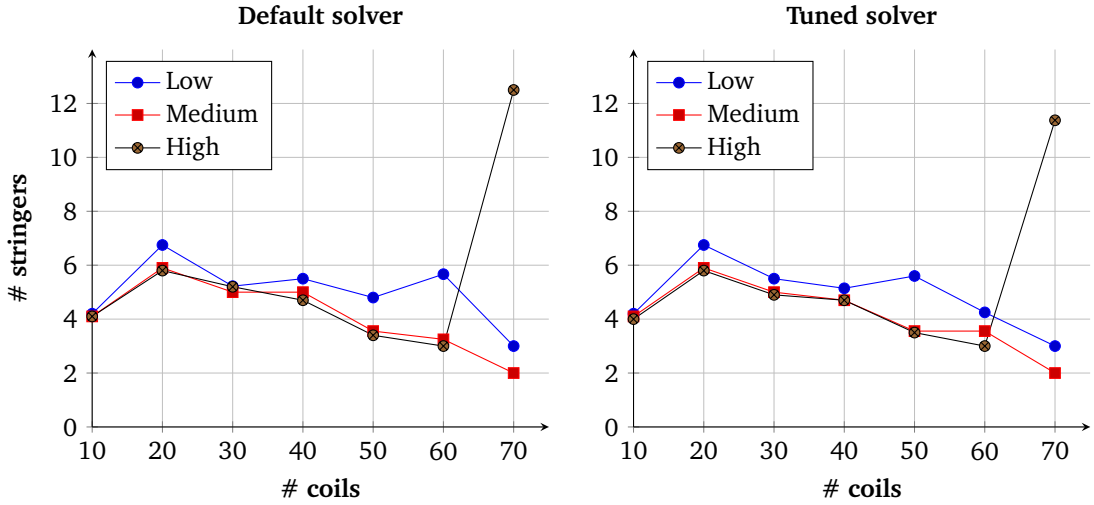

Figure 3: Average number of introduced stringers per $\mathcal{N}$ and service level in optimizations with the default and tuned solver.

therefore it is not a result of an outlier as it was in the case of 70 coils. Additionally, this suggests that this is not a random occurrence but a trend. Hence, it might be beneficial to observe the relative average number of stringers introduced, which are depicted in figure 4.

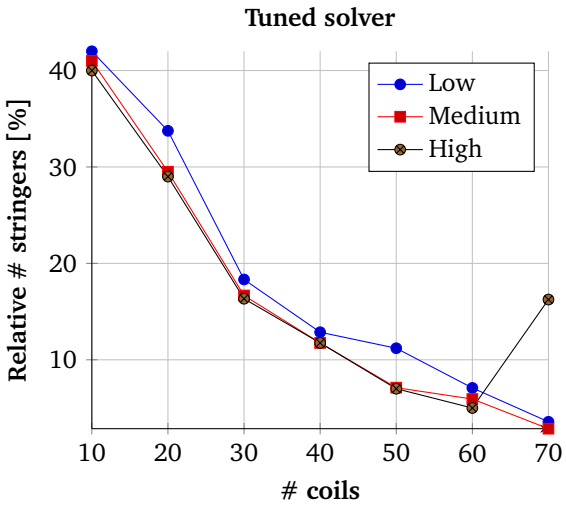

Figure 4: Average number of stringers relative to the instance size of optimizations with the tuned solver in %.

Figure 4 portrays the average number of stringers relative to the instance size of the optimizations. To increase the visualization, only data from the tuned solver is shown. The figure depicts, that the absolute numbers of coils are deceiving in this regard and that in fact, the results from the optimization with 20 coils are in line with the observed downward trend. Now that the existence of a constant downward trend, except for the case with seed 100 mentioned above, is established, a possible explanation can be discussed.

To recapitulate, a stringer has to be introduced between adjacent coils if the difference between the physical or mode-specific or both characteristics of these coils is above the limit specified by the process flexibility PF. The introduction of a stringer can be avoided if the schedule introduces a third coil between these incompatible coils, but this is only possible if the third coil is compatible with the other two coils. An explanation for the high relative number of introduced stringers in optimizations with small instance sizes could therefore be the absence or an insufficient number of these third coils that can bridge the gap between two incompatible coils. By increasing the number of coils, these gaps could then be closed or narrowed by the new coils. The likelihood of this increases with the increasing number of coils, which results in fewer stringers that have to be introduced due to differences in characteristics.

To give a simplified example, in which only the annealing temperature of the coils is considered as compatibility restriction: Three coils have to be scheduled in sequence, with coil one having an annealing temperature of 670 °C, coil two having an annealing temperature of 740 °C and coil three with an annealing temperature of 720 °C. The PF is 40 °C in a symmetric interval and the annealing temperature is only distributed in increments of ten °C, as explained in section 4.1. Thus, coil two and three are compatible with each other, but not with coil one. Since there is no other coil to bridge this gap, a stringer has to be added between them. Now, a fourth coil has been added to the schedule with an annealing temperature of 700 °C. It is therefore compatible with coil three, but not with coils one and two. Since no coil is compatible with coil one, a stringer still has to be added, but the relative number of stringers decreased from 33,3 % to 25 %. If another coil with an annealing temperature of 680 °C or 690 °C would be added, it would close the gap and lead to a stringer-free schedule. If it has a different annealing temperature though, it would still be compatible with one of the remaining coils and therefore, still only one stringer would

be necessary, decreasing the relative number of stringers even further. By adding more and more coils to this schedule, the probability that one of the coils eventually closes this gap increases.

The reason for the increase in the number of absolute stringers from ten to 20 coils could therefore be that most of the existing gaps could not be closed by the new coils and that some coils may have characteristics which were incompatible with both ends of the gap, thus increasing the number of stringers that have to be introduced. To summarize, the probability of a single coil being compatible with the other coils in the schedule is higher for a higher number of coils. Thus, these schedules with a higher number of coils need fewer stringers, since there are fewer gaps and more possibilities to sequence these coils to avoid gaps. In the actual optimizations performed in this study, there are more compatibility restraints as well as due dates to consider, which is why in most cases there are still stringers introduced in the schedule, even with higher numbers of coils. The impact of the restriction by due dates can be observed by comparing the average of the optimizations with a high and low service level because, in fact, most optimizations do not utilize the entire service level. This is portrayed in figure 5.

Since the service level is a parameter to limit the total amount of delayed coils relative to its instance size, the threshold for relative tardiness for a high service level is 20 % and so forth. As mentioned above, it can be observed that on average and with a high service level, at no instance size the full service level of 20 % is leveraged, with the highest average being 19,25 %. The data suggests, that for most cases a service level of 16 % to 17 % would suffice. This is supported by the fact, that optimizations with a service level of ten % always utilize the entire service level or almost all of it. This does not imply that if the service level would be increased to over 20 %, the solver would not leverage it. But it states, that to receive the solutions displayed in figure 3, a service level under 20 % would suffice, at least in most cases. As expected with a low service level, the solver does not allow a delay of coils.

The data also suggests a slight decrease in delays in optimizations with a higher number of coils through the tuning of the solver. But, as previously described, this could be affected by the selectivity effect. When comparing the results of figure 3 and figure 5, it can be observed that even though solutions found with a service level of 20 % always had a higher number of delays, they did not necessarily yield more optimal solutions in terms of stringers than optimizations with a service level of only ten %. In fact, the most significant difference in average stringers introduced between the two parameterizations was only 0,556 stringers, which occurred in the case of an instance size of 60 coils with the default solver and thus could be influenced by the selectivity effect.

The question that could arise is, if not profoundly positively affecting the solutions in terms of introduced stringers, what is the benefit of using a higher service level? This question could be answered by taking a look at figure 6.

The data of higher instance sizes, especially 60 and 70 coils, seems to be strongly affected by the aforementioned selectivity effect and will therefore be disregarded in the following discussion concerning the total optimization time. The top graphs portray, that a higher service level usually leads to a lower average total optimization time, which could save costs caused by the operation of the computer, but that in general, the total optimization time increases profoundly. This trend can be explained by the bottom graphs, which display the extraordinary increase in variables and constraints, and thus the size of the MILP. The stark increase in variables is mostly driven by binary variables, like y_{ijkmn} and x_{ikm}. Since the number of variables and constraints are not influenced by the parameterization of the service level or tuning, they are only displayed by one graph each. When comparing the top with the bottom graphs, a positive correlation can clearly be observed. Therefore, the probable explanation for the increase in total optimization time is the growth of the model the solver is trying to solve.

The size of the model has two different impacts on the total optimization time. The first and most obvious is, that it has to consider a growing number of constraints and variables when trying to find a feasible solution, which increases the time it takes to prove the feasibility of it. But before the model can be solved, it first has to be built, which is the second impact of the model size. The time it took the solver to build the model rose from only 4,5 seconds with ten coils to over 90 seconds with 50 coils, which does not seem to be affected by the service level or by tuning. Therefore, the model should be kept as small as possible to avoid wasting optimization time, which is especially important when certain circumstances severely limit it. What seems to be affected by the service level and by tuning is the solving time, which accounts for the majority of the total optimization time. Tuning can in fact increase the solving time through different parameterizations of $PrePasses$ and $Presolve$, as an example.[54] In the case of this study, the tuning thus seems to increase the success rate of optimizations, especially with greater instance sizes, at the cost of solving time. A higher service level, on the other hand, seems to decrease the solving time, which suggests that the solver can find optimal solutions faster if it is less restricted. The MIPGap also seems to benefit from an increased service level, as depicted in figure 7.

A positive trend between the MIPGap and the instance size can be observed, as well as the trend that optimizations with more scheduling flexibility tend to have a higher grade of proven optimality, which can be explained by the lower solving times. The lower the service level, the more optimizations reach the solving time limit of 720 seconds and thus have to abort the solving process. Hence, the solver has no or less time to prove the optimality of the solution, resulting in a higher MIPGap. This does not necessarily result in a worse solution, as the solutions, in terms of stringers, between a service level of ten % and 20 % only differed slightly, as portrayed in figure 3. Thus, the general upwards trend of

[54] Gurobi, n.d.-b

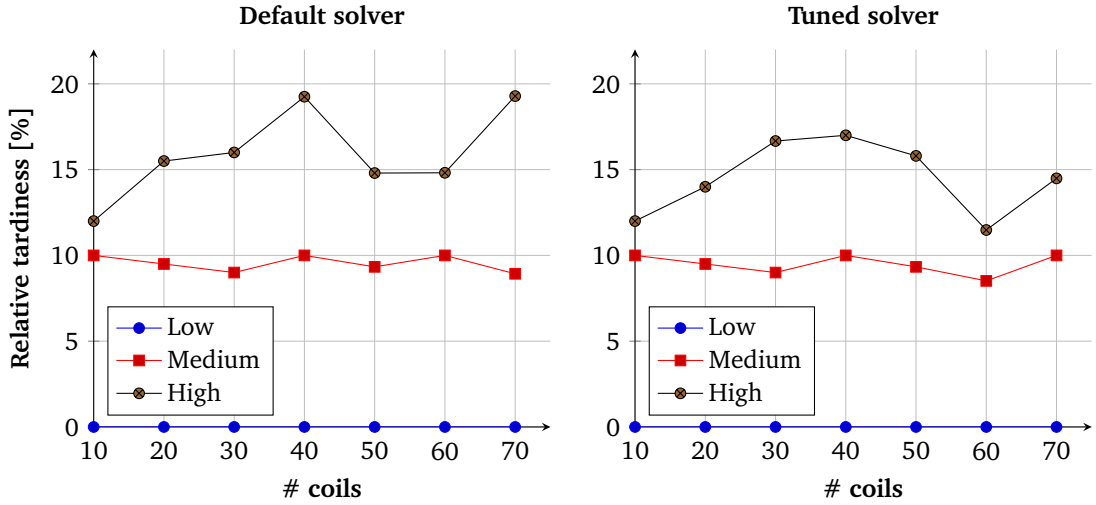

Figure 5: Average tardiness relative to the different instance sizes per service level in optimizations with the default and tuned solver.

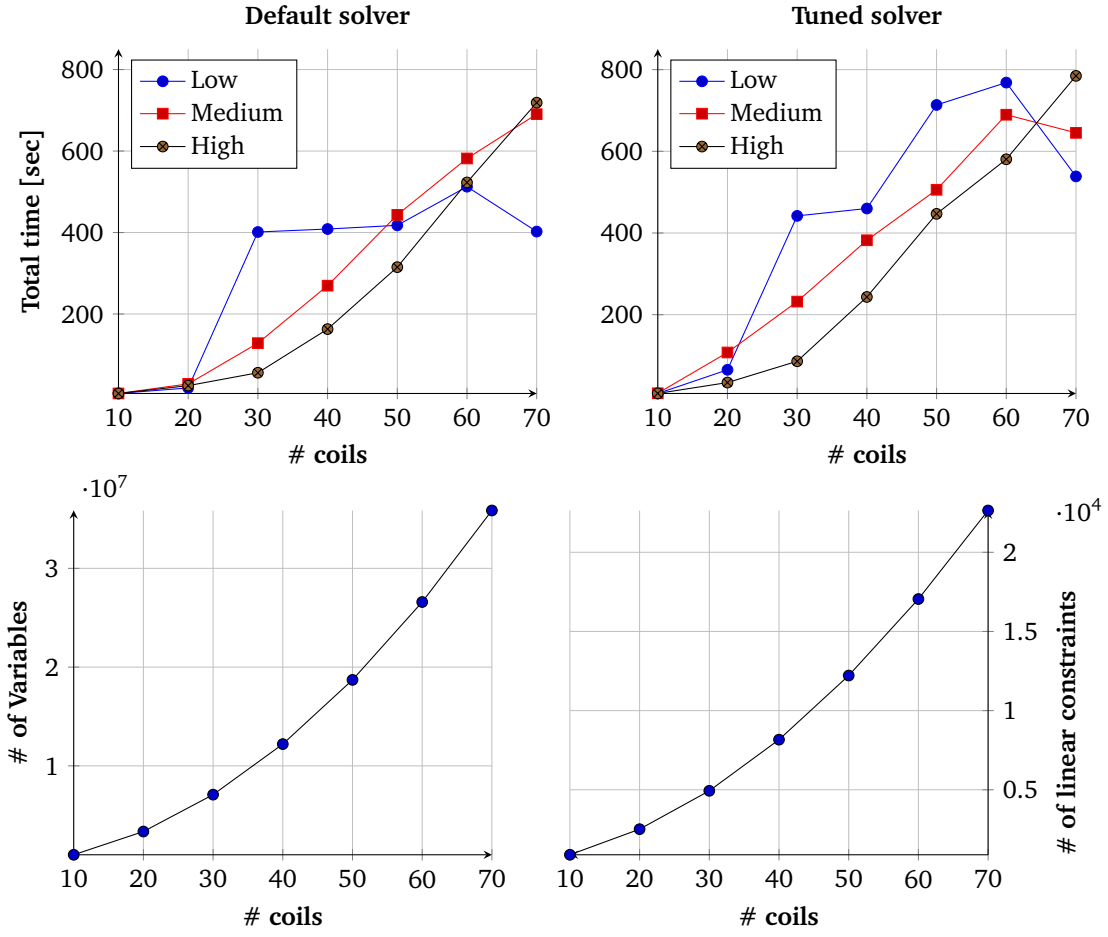

Figure 6: Average total optimization time (top) and number of variables (bottom-left), as well as average number of linear constraints (bottom-right) per service level and instant size in optimizations with the default and tuned solver.

the MIPGap could be a consequence of the increasing solving times of each seed. Hence, it could be advantageous to choose a higher service level if the available optimization time is severely limited or if a higher proven grade of opti-

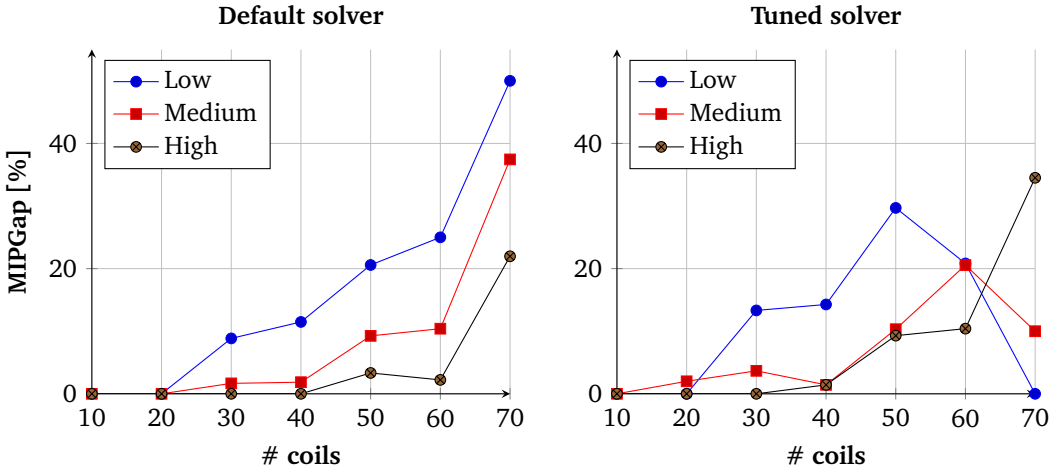

Figure 7: Average MIPGap per instance size and service level with the default and tuned solver in %.

mality of the feasible solution is preferred. The data from the tuned solver seems to be affected by the selectivity effect, but the MIPGaps of smaller instances seem to be similar to that of the default solver. The mentioned increase in solving times is displayed in figure 8 with data from optimizations solved by the default solver due to its better visualization of this trend. All boxplots were created with the median and the highest and lowest value from the ten optimizations, as well as the 25 % and 75 % quartile.

Figure 6 already depicted the increasing average total time. But this data could suggest that this applies to every seed, when in fact it does not. The data in figure 8 portrays that, starting from an instance size of 30, the differences in solving times increase drastically if they are solved with the two lower service levels. This increase can be observed from 40 coils onward, with the highest service level. The higher number of optimizations reaching the time limit combined with the higher average MIPGaps in optimization bound by the lowest service level indicates, that the solver has no time to prove the optimality of the feasible solution after finding it. With a medium service level, fewer optimizations have to be aborted due to the time limit and thus have time to prove the optimality of the feasible solution, which results in a lower average MIPGap. This is also true for the highest service level that displays the fewest number of time limit terminations.

This concludes the first part of the numerical studies. Based on the presented data, an instance size of 40 coils was chosen for the following sections since it is the highest number of coils with a tolerable amount of failed optimizations. In addition, the optimizations were conducted with the tuned solver to increase the number of successful optimizations.

4.2.3. Best- and Worst-case scenario

For the third part, a Best- and Worst-case scenario was derived from the different factors defined by Wegel et al. (2024), which were described in section 4.1. The results will be compared with the results achieved with the tuned solver from the previous section. In addition, the PF and HC were altered because it was suspected that these factors are the most influential among them. A more detailed analysis of the influence of each factor and level on the scheduling is provided by Wegel et al. (2024).

Figure 9 compares the number of stringers introduced between the Best-case scenario with a *Very low* PF and the *Basecase* scenario. The data from the *Very high* and *Basecase* PF is not portrayed, as no stringers are introduced at any point in these scenarios. As previously observed, the number of stringers decreases with higher service levels in the Best-case as well due to the higher scheduling flexibility. Compared with the results from the *Basecase* scenario, the Best-case yields better solutions, in terms of stringers, for every level of PF and for every service level. Even with *Very low* PF, a stringer reduction of over 50 % can be observed. Thus, a lower PF leads to an increase in stringer introduction, but in this case, this effect is probably reduced by the *Very low* HC. Since these effects could therefore counteract each other, the stringer reduction may be a result of the lower UC and SPT. If true, this could also result in fewer delayed coils, which are depicted in figure 10.

By comparing the average tardiness in the Best-case scenario - *Very low* PF with the tardiness from the *Basecase* scenario, only a slight decrease can be observed. This difference increases with the B. PF, but does not with the highest PF. Thus, the *Very low* UC and SPT do not lead to severely less tardiness when HC and PF are parameterized as *Very low*. But the increased scheduling flexibility through more relaxed time constraints could still benefit the optimization, thus decreasing the number of introduced stringers.

Figure 10 also displays the distribution solving times. The data suggests, that the level of PF, in the context of the Best-case scenario, has an immense impact on the solving time. The optimizations with the lowest PF have the highest solving times, even higher than in the *Basecase* scenario. But by increasing the PF to the *Basecase* level, the median solving

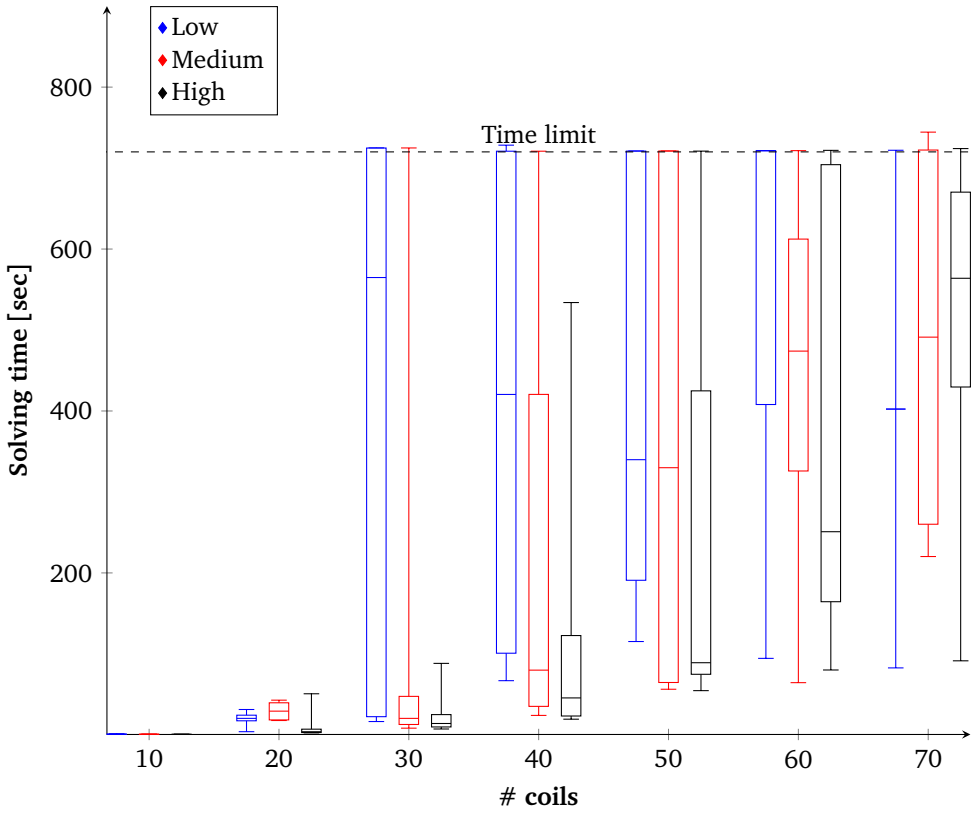

Figure 8: Distribution of solving times in seconds per instance size and service level with the tuned solver.

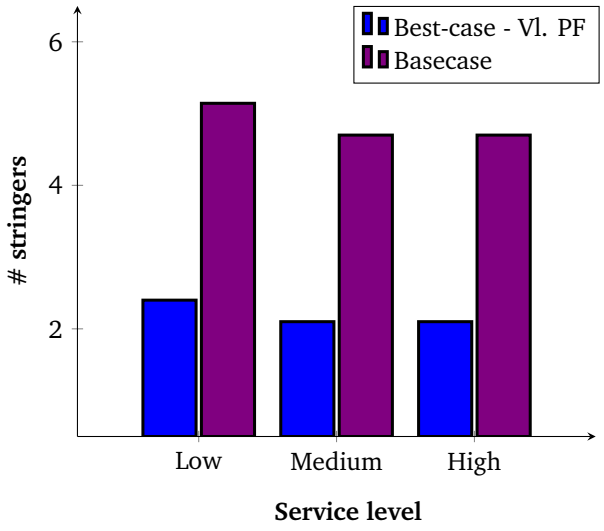

Figure 9: Average number of introduced stringers per service level in the Best-case scenario with *Very low* PF and *Basecase* scenario.

times decrease to a fourth. Further increasing the PF to *Very high*, however, does decrease the solving times only slightly. That the solving times for the two higher investigated PF levels are remarkably lower than in the *Basecase* scenario can be explained by the parameterization of the HC, UC and SPT. As previously described, the *Very low* UC and SPT increase the scheduling flexibility by relaxing the time constraints. The *Very low* HC, on the other hand, increases the general compatibility between different coils, which leads to more feasible, stinger-free processing sequences. All of these factors could result in more solutions being feasible, thus decreasing the time it takes the solver to find an optimal, feasible solution. This could be applied to the lowest PF as well but in reverse. It could suggest that the *Very low* PF drastically limits the number of feasible, stringer-free processing sequences of adjacent coils, which increases the solving time since the use of stringers has to be minimized. In this regard, the other very beneficially parameterized factors cannot offset the *Very low* PF, which could indicate that the PF is the most impactful factor of the four. This statement could be confirmed by the data from the Worst-case scenario.

Figure 11 portrays the impact of the HC on the Worst-case scenario in terms of introduced stringers and failed runs. As previously described, *Very low* HC is beneficial for stringer-free sequences. In this case, the HC seems to impact the average number of stringers more than the other factors, displayed by the fact that the lowest HC in combination with the Worst-case yields better solutions than the *Basecase* scenario. This would contradict the statement, that the PF is the

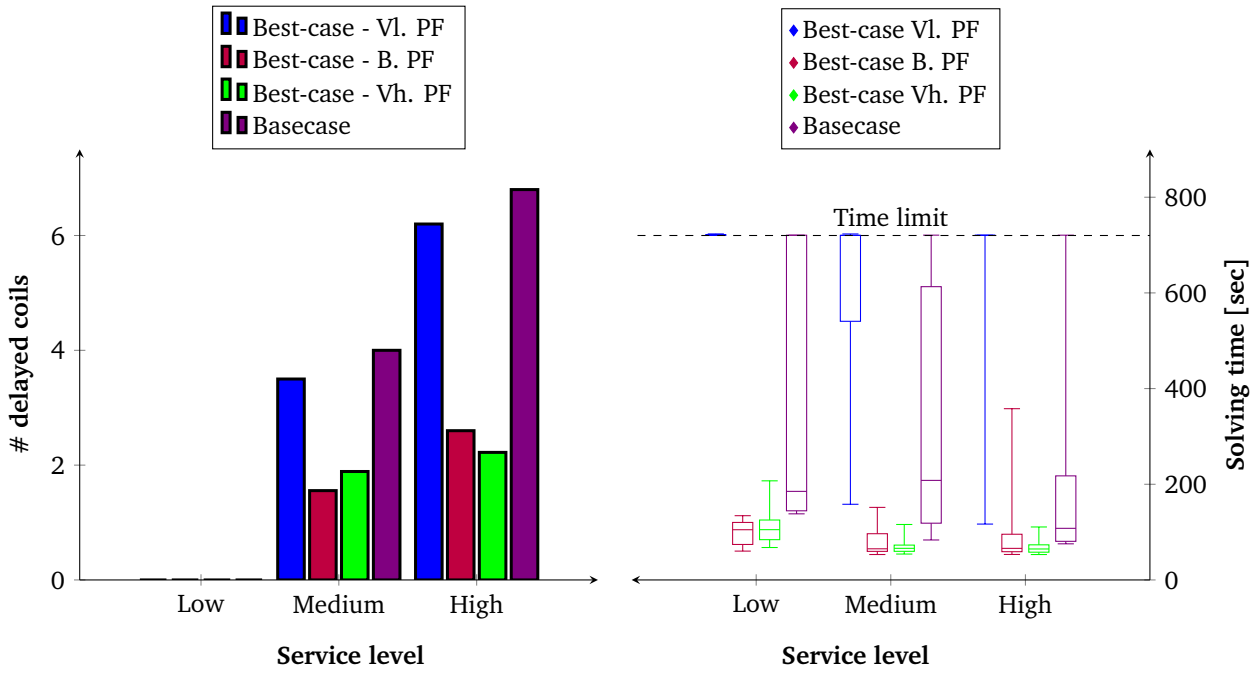

Figure 10: Average number of delayed coils and distribution of solving times in seconds per service level in the Best-case instances and *Basecase* scenario.

most influential factor. But the data from the two other HC parameterizations support this claim. While the optimizations with Worst-case and *Basecase* HC did still yield some but very high solutions, no optimization was able to calculate a feasible solution with the highest HC.

In general, the Worst-case scenario seems to negatively impact the number of successful optimizations. As previously described, the reasons are twofold. While all runs with the lowest service level fail due to infeasibility, optimizations with a higher service level fail because of the solving time limit, with two exceptions with a medium service level and *Very high* HC. This demonstrates the great impact of the Worst-case and especially of the HC on the feasibility of the model. Before finalizing the second part of this numerical study, one last test was conducted to analyze which of the two factors, HC and PF, impact the solution the most. To accomplish this the case in table 10 was defined.

Table 10: Case parameterization to identify the most impactful factor.

Case	HC	UC	PF	SPT
Impact	Very low	Basecase	Very low	Basecase

The figure 12 displays the differences in stringers and delayed coils between the *Basecase* scenario and the Impact case, by subtracting the average number of stringers and delayed coils in the Impact case from the *Basecase* scenario.

It can be concluded, that the PF has a greater impact on these aspects of the solution than the HC. This observation was confirmed by a second Impact case, which used the *Basecase* scenario and *Very high* PF and HC. But it has to be noted that this only applies to the definitions of the different levels of HC and PF by Wegel et al. (2024). Both factors indicate a strong influence on the characteristics of the solution and should therefore be optimized to achieve optimal solutions in real-world applications. After studying the drastic effects of the Best- and Worst-case, different parameterizations of the model will be explored in the following section.

4.2.4. Alteration of processing lines and due dates

First, the number of processing lines in the model was altered. To recapitulate, in the first case, an additional line was added to the model, which can process every coil. In the second case, Removal I, the processing line that can only process wider lines was removed, while the processing line that can only process narrower coils was removed in case three. The line that can process every coil was not removed because it would lead to infeasibility since some coils could not be processed.

Foremost, none of the three cases displayed in figure 13 seems to have a great impact on the number of introduced stringers. As expected, the number of stringers decreased in the addition case and increased in both removal cases, but all these effects were only minor. The reason the addition of one processing line generally reduces the number of stringers is that an incompatible coil or a sequence of coils, that is incompatible with another sequence of coils, can be processed on the additional line, thus reducing the number of stringers by one.

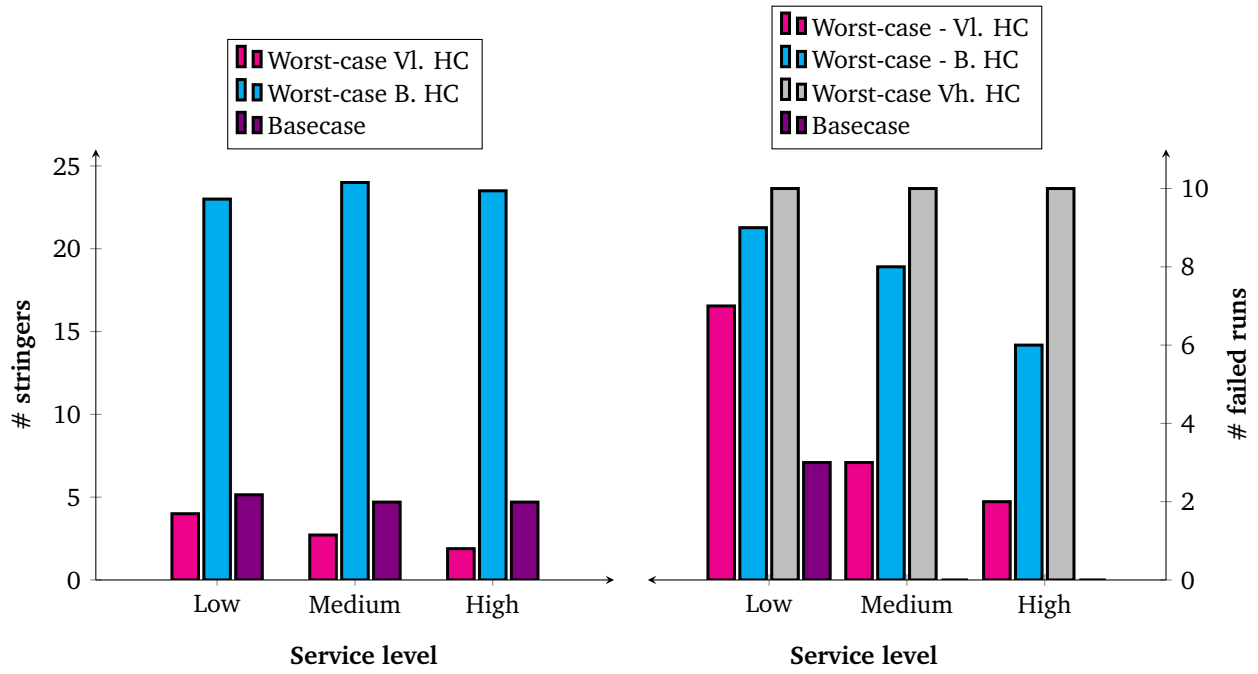

Figure 11: Average number of introduced stringers and failed runs per service level in the Worst-case instances and *Basecase*.

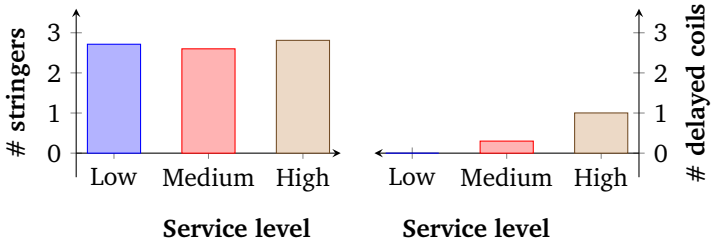

Figure 12: Differences in stringers and delayed coils between the Impact case and *Basecase* per service level.

To give an example, four sequences of coils have to be scheduled, with each coil in a sequence being compatible with every other coil in the same sequence. However, the sequences are incompatible with each other. If these four sequences are processed on one processing line, three stringers would have to be introduced to ensure compatibility. If a new processing line would be added, one of the sequences could be processed on the new line, reducing the number of stringers to two, but nothing more could be done to reduce the number of stringers further.

Therefore, the maximum benefit of adding a processing line, in terms of stringer use, is the reduction of introduced stringers by one. This benefit could increase if the schedule introduced stringers to not violate the service level, but this was not the case for these optimizations.

The reason the data does not show a reduction of the average introduced stringers by one is that two optimizations, that were unsuccessful in the *Basecase* scenario, were successfully solved in the addition case and have a stringer count of six, thus increasing the average number of stringers. In fact, all three cases had only one failed optimization each with a low service level. The explanation for this reduction could be the solving itself since the solver behaves differently even if only one parameter changed.[55] Because the Addition case and the Removal cases have opposite effects, the addition of a processing line increases the number of feasible solutions and the removal decreases them, it is the most likely explanation.

The removal of a processing line influences the number of stringers in the same way as an addition of a line does but in reverse. Thus, if no stringers are introduced to not violate the service level, the maximum increase in stringers is one. This is true for every optimization with a MIPGap of zero %. However, some optimizations with a MIPGap of zero % did not suffer any increase in stringer use, suggesting that they did not utilize the line in the schedule. But some optimizations with a MIPGap of over zero % had an increase of up to three stringers. Hence, the optimizations either had

[55] Miltenberger, 2023b

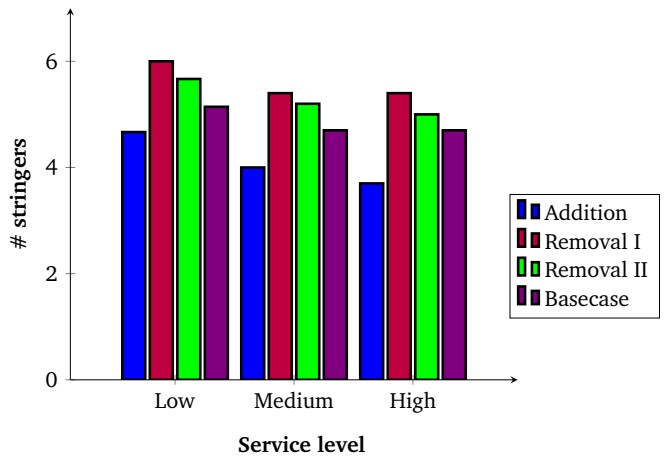

Figure 13: Average number of stringers in the three line-altering cases and *Basecase* scenario per service level.

to introduce stringers to comply with the service level and the solver was unable to prove the optimality of the solution in time or the solver simply could not find the optimal feasible solution until it had to terminate the solving process. This termination happened more often with the addition of one line and is presented in figure 14.

As portrayed, the majority of optimizations with an additional processing line and a low service level had to be terminated due to the time limit. This results in a higher MIP-Gap, as mentioned before. Higher solving times and MIP-Gaps than in the *Basecase* scenario are also observed with higher service levels but are not as severe with the highest one. This overall trend probably results from the increasing model complexity, which could also be observed in figure 6 from the first part of this numerical study. In this case, however, the increase results from the additional line and not from additional coils. The adverse effect can be examined in figure 15.

The data indicates a decreasing solving time through complexity reduction when applying a medium or high service level, but not with a low one. A possible explanation could be, that the restrictions caused by the removal of a processing line in combination with the low service level result in many former solutions being infeasible and thus a longer search for a feasible solution. That Removal II displays lower solving times than Removal I could suggest that the line which can only process wider coils and was removed in Removal I, was utilized to a higher degree than the line that was removed in Removal II. The reason for this is the partially triangular distribution of the coil width due to the *Basecase* level of the HC, which results in more coils being wider than narrower. Hence, the effect of the removal of one line depends on the HC setting.

The last parameter to investigate is tardiness, which indicates to not be severely affected by the addition or removal of a processing line. That the number of delays only complies with the service level and does not have to be minimized could explain why tardiness did not decrease with an additional line. Additionally, no increase in tardiness suggests that most optimizations in Removal I and Removal II had enough scheduling flexibility to find feasible solutions with similar tardiness, even with only two lines.

In contrast, tardiness was the most important factor in the next part of the numerical study, in which the time horizon for the due dates was moved closer to the beginning of the schedule. These earlier due dates are expected to have a certain effect on the number of stringers and delayed coils. With earlier due dates and an imposed service level, the schedule has to implement more stringers to not exceed the allowed tardiness. This further reduces the scheduling flexibility due to the SPT and thus increases the difficulty of finishing the processing of other coils before their due date. Therefore, an increased number of failed optimizations and a higher stringer and tardiness count in successful runs were expected.

The data displayed in figure 16 shows an increasing number of failed optimizations, which aligns with the expectations. In case 100 %, most optimizations are terminated and do not fail because of infeasibility, suggesting that they could be solved with a higher time limit. But this changes with case 50 %, in which no optimization with a low service level is feasible, while most unsuccessful optimizations with a higher service level are terminated. By reducing the values of a and b by 75 %, almost all optimizations with a low and medium service level are infeasible, but only one optimization with a high service level. Therefore, provided with a high service level and a higher time limit, most optimizations could still yield feasible solutions. However, the necessary solving time and the quality of the resulting solution cannot be determined without further studies.

The second expectation was a rise in the number of introduced stringers. Due to the very low number of solutions, the following interpretations could be heavily influenced by the selectivity effect. Because of this, only the most important information will be presented in table 11 and briefly discussed. If a case is not included in a service level, then no optimizations for that case and service level yield a feasible solution.

As indicated by the data, the number of introduced stringers increases through the shift of the due dates. In addition, almost all optimizations completely utilize the service level. Both of these observations represent an increase in comparison to the *Basecase* scenario and align with the expected results, thus concluding the third part of the numerical study.

4.2.5. Numerical studies on the extended models

The last part of the numerical study consists of several tests conducted on the extended models described in sections 3.4.2 and 3.4.3. Both additionally incorporate the release dates of coils from section 3.4.1. The first extended model minimizes stringers and absolute tardiness. First, the average number of stringers and delayed coils in the

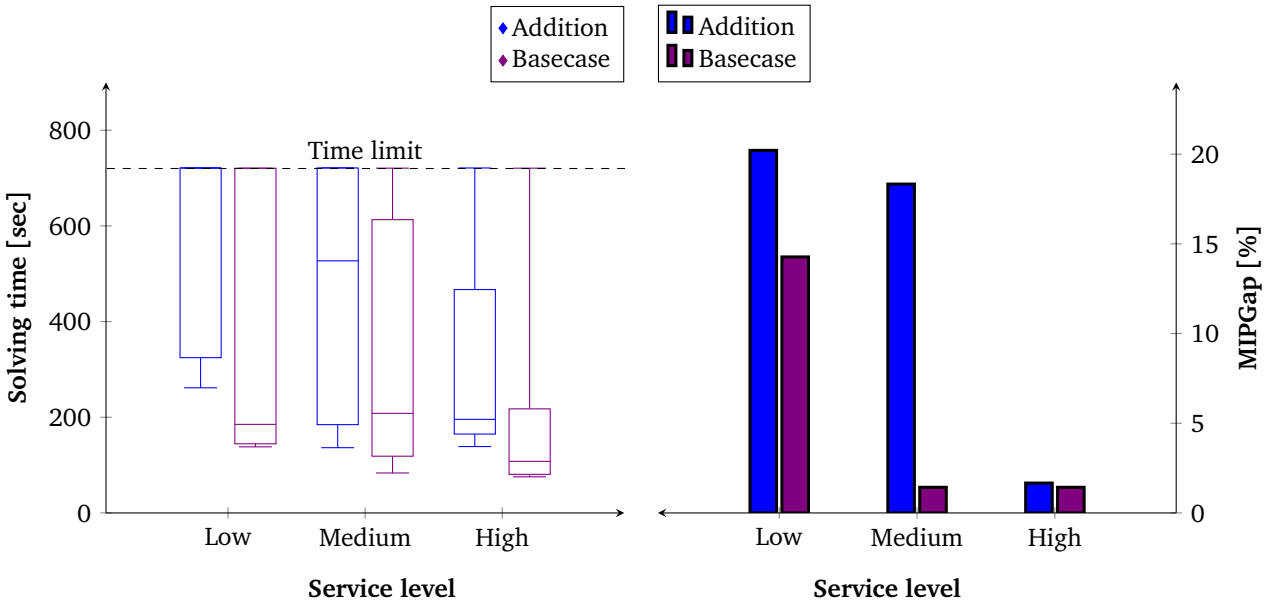

Figure 14: Distribution of solving times in seconds and average MIPGaps in % per service level in the Addition case and *Basecase* scenario.

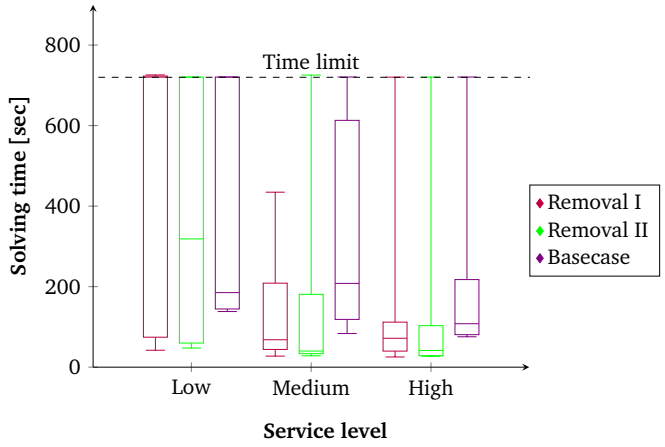

Figure 15: Distribution of solving times in seconds per service level in the Removal I and II cases, as well as in the *Basecase* scenario.

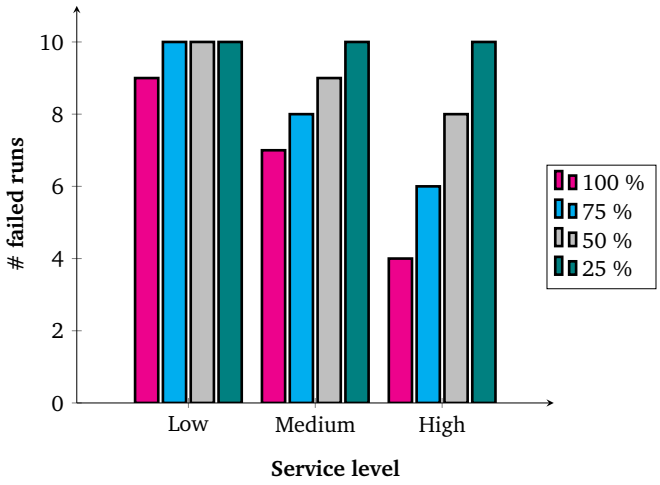

Figure 16: Number of failed runs per service level in the four different due date altering cases.

Basecase scenario with *Very low* and *Very high* UC will be examined.

Figure 17 depicts an expected trend. The number of delayed coils is the highest if it has only a weight of 20 % of the objective value, which has to be minimized. Additionally, the number of stringers is the lowest at this point, since it has a weight of 80 %. Thus, the optimal solutions seem to delay some coils to minimize the number of stringers, since it has a much higher weight. This trend changes, with tardiness being reduced at the cost of stringers with higher weights of tardiness. As portrayed with *Very low* UC, the number of delayed coils is, even at its highest point, very low. Thus, the trade-off between stringers and tardiness cannot be observed as well as with the *Very high* UC on the right side of figure 17.

In this scenario, an almost one-to-one trade-off between the number of delayed coils and stringers can be observed with the increasing weight of the former. The mechanism behind the trade-off is similar to the one discussed in the third part of the study, regarding the addition and removal of processing lines. If the tardiness has almost negligible weight, the coils are sequenced in a way that the number of stringers is minimized with no regard to the number of delayed coils. But if the tardiness weight increases, the objective value may benefit from introducing stringers to finish the processing of

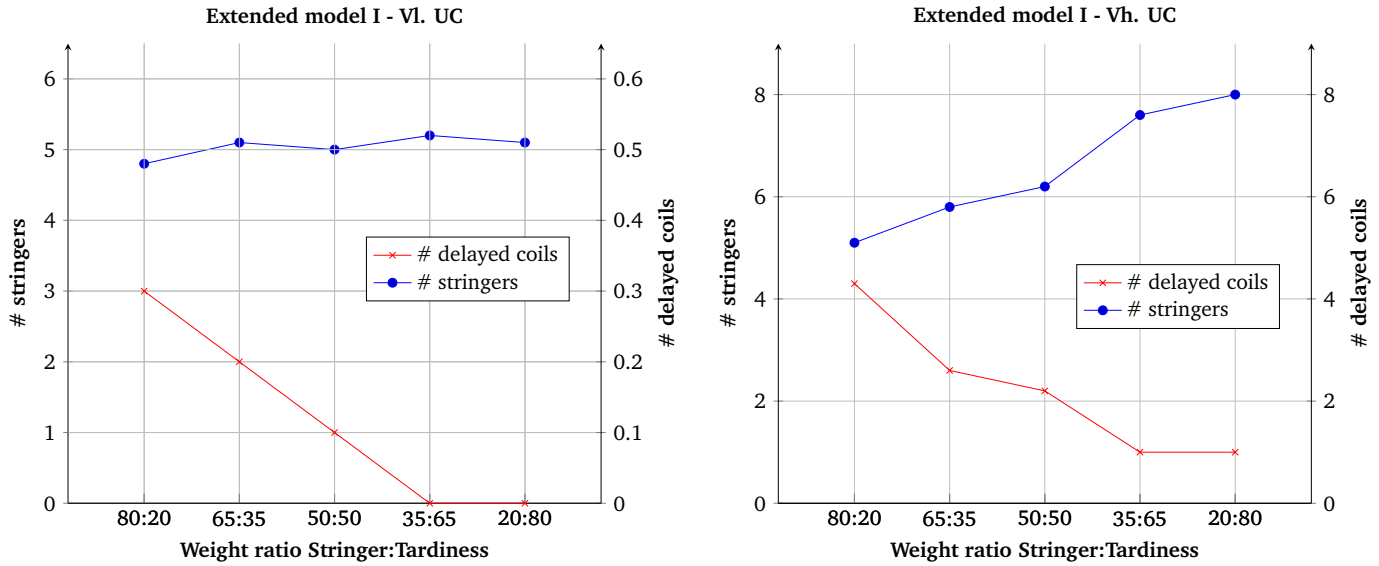

Figure 17: Average numbers of delayed coils and stringers with different UCs and weights in the first extended model.

Table 11: Stringer use and tardiness in the cases 100 %, 75 % and 50 %.

Service level	Case	# stringers	# delayed coils
Low			
	100 %	6	0
Medium			
	100 %	6.667	4
	75 %	7	4
	50 %	7	4
High			
	100 %	5	7.83
	75 %	5.75	8
	50 %	7	8

coils before their due date, thus decreasing the number of delayed coils at the cost of stringers. By further increasing the tardiness weight, more stringers are introduced to comply with due dates and therefore to minimize the objective value. With earlier due dates, this trend is better to observe since it is harder to comply with them, which increases the number of stringers that have to be introduced to decrease the tardiness by one unit and vice versa.

Due to the different optimization objectives, the comparability between the extended model and the base model is limited. But the results of the extended model with a tardiness weight of 20 % are similar to the results yielded with a medium service level in the base model, presented in figure 3. The difference, however, is the UC, which is higher in the extended model and thus highlights its performance to minimize stringer use and tardiness at the same time. But this high performance comes with the cost of increased solving time, portrayed in figure 18.

Generally, there is a wide disparity in solving time between different optimizations in the extended model, regardless of the weight ratio, with most median solving times being higher than that of the *Basecase* scenario with a low service level.

The highest median solving times and terminations in the extended model can be observed with the two higher stringer weights of 80 % and 65 %. The median solving times of the other instances are significantly lower, which could suggest that the minimization of stringers is more complex than that of delayed coils. But this cannot explain, why the median of the solving time increased by increasing the tardiness weight further to 65 %. This effect mostly stems from one optimization, which suffered an increase of 510 seconds in solving time from 210 seconds to 720 seconds. Without this run, the median would be 283 seconds, which is similar to the instances with 50 % and 80 % tardiness weight and could indicate that this optimization is an outlier.

The solving times of the extended model with a *Very high* UC are not depicted, since all of them had to be terminated due to the time limit. Nonetheless, every optimization yielded a feasible solution, which is a drastic increase over the number of failed runs in case 100 % of the third part of the numerical study, displayed in figure 16. Since every result with the extended model and a *Very high* UC also had a low relative tardiness and could therefore comply with the medium and high service levels implemented in the base model at almost every weight ratio, it suggests that it could be more performant to minimize both stringers and tardiness than to implement a service limit and only minimize

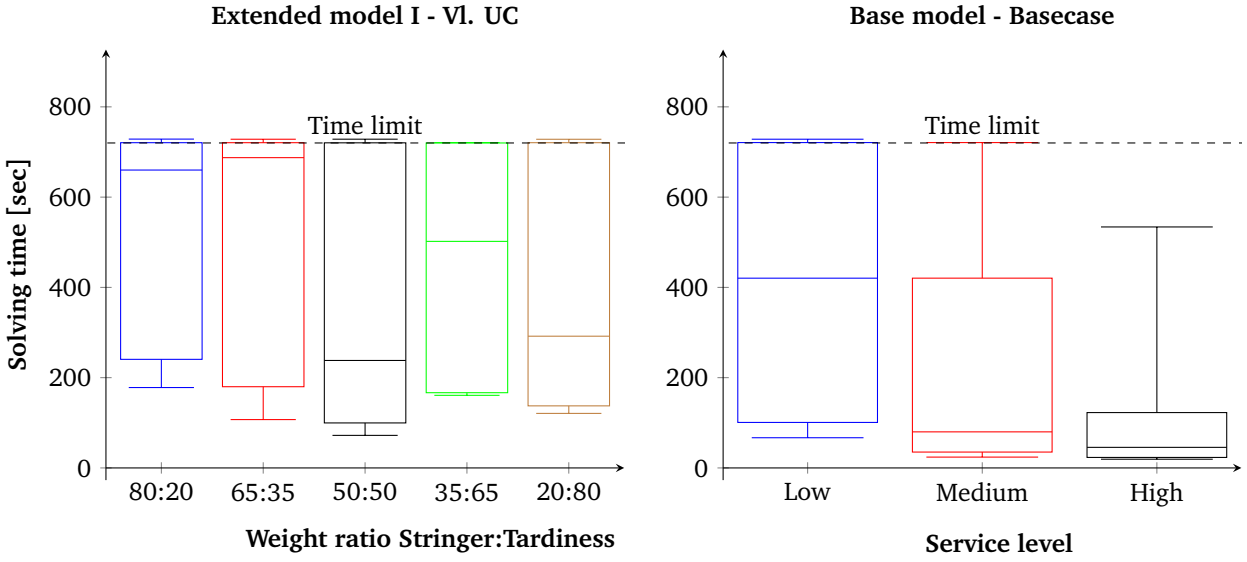

Figure 18: Distribution of solving times in seconds per service level in the base model and per weight ratio in the first extended model.

stringers.

Figure 19 displays a similar trend as in the first extended model, but in this case, the trade-off happens between earliness, tardiness, and stringers. The number of the latter is far greater than in the base and the other extended model. This could be explained by the high MIPGaps resulting from the fact that every optimization had to be terminated due to the time limit.

Therefore, it is possible that the displayed stringer counts are far above the optimal number of stringers. But it has to be noted that every optimization was successful, which is an improvement from the base model. Since the first extended model also had fewer failed runs, the service level might be too restricting, especially in its lower specification. The service level is the most likely cause due to it being the only part that was removed in both extended models.

Due to the lack of reference points, the exact values of earliness and tardiness cannot be compared with other studies. It can be observed, that by increasing the tardiness weight the tardiness can be minimized drastically, but by changing the earliness weight, the earliness only differs slightly. This could result from the different mechanisms of how to minimize tardiness and earliness.

Tardiness can be reduced by finishing the processing of each coil before its due date, which might influence the coil sequencing and thus the stringer use, as previously described. In contrast, to minimize earliness the schedule has to be prolonged, which can be achieved with two methods. The first method would be to introduce stringers in the sequence, even if they are not necessary since they also have a SPT. The second method is to choose longer-lasting processing modes for the coils, which could increase the stringer use as well since the compatibility of these modes would not be as important as it was before. Since stringers still have to be minimized as well, it is more likely that the second method is used to prolong the schedule and that the additional introduction of stringers is therefore only a result of the increasing incompatibility between the processing modes.

Regardless of the reason, an increase in stringer use can be observed with increasing earliness weights, as well as a higher overall stringer use with higher due date deviation and thus earliness weights, which suggests that one of the proposed methods is being utilized. It has to be noted that the change in stringer use is a direct result of the changing earliness and tardiness weights since the stringer and due date deviation weights are fixed throughout the instance. To further prove the earliness minimization mechanism, the optimizations of the instance with a due date deviation weight of 65 % were repeated, but with a $Very\ high$ UC.

Figure 20 portrays a similar stringer use but a significantly lower earliness and therefore supports the proposed mechanism. This is because, as observed in table 11 and figure 17, earlier due dates result in higher stringer use due to the previously described mechanism. But the stringer use did not increase by increasing the UC, which could be a result of less utilization of the proposed earliness minimization mechanism, as this would result in fewer stringers. Additionally, the reduction of stringer use through the reduction of the earliness weight did decrease slightly, but not as much as expected. Therefore, more tests need to be conducted, including alterations of the SPT or the data has to be analyzed regarding the used processing modes and introduced stringers to prove the utilization of the mechanism.

Another observation is the reduction of stringers, earliness and tardiness with certain weight ratios. Since this only occurred with more equal earliness and tardiness weights of 35 % and 50 %, it could indicate that the minimization of multiple objectives benefits from more equal weights between them. But since it did not happen in the instance with a stringer weight of 35 % with *Basecase* UC in figure 19, it

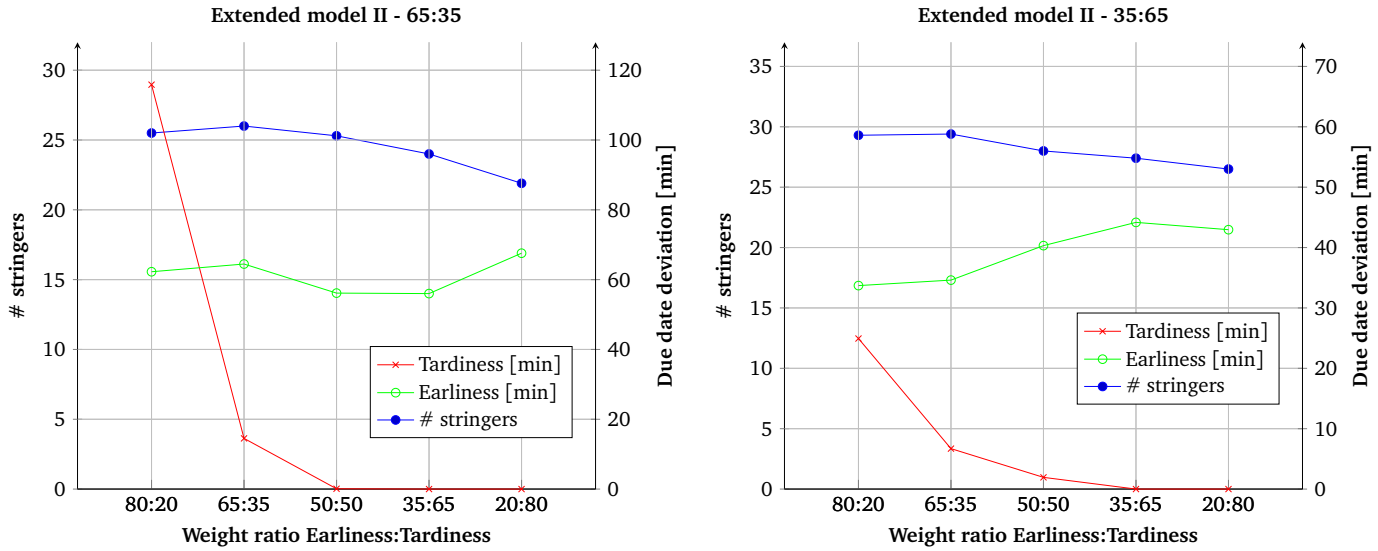

Figure 19: Trade-off between earliness and tardiness in minutes and its effect on the stringer use with different stringer and due date deviation weights in the second extended model.

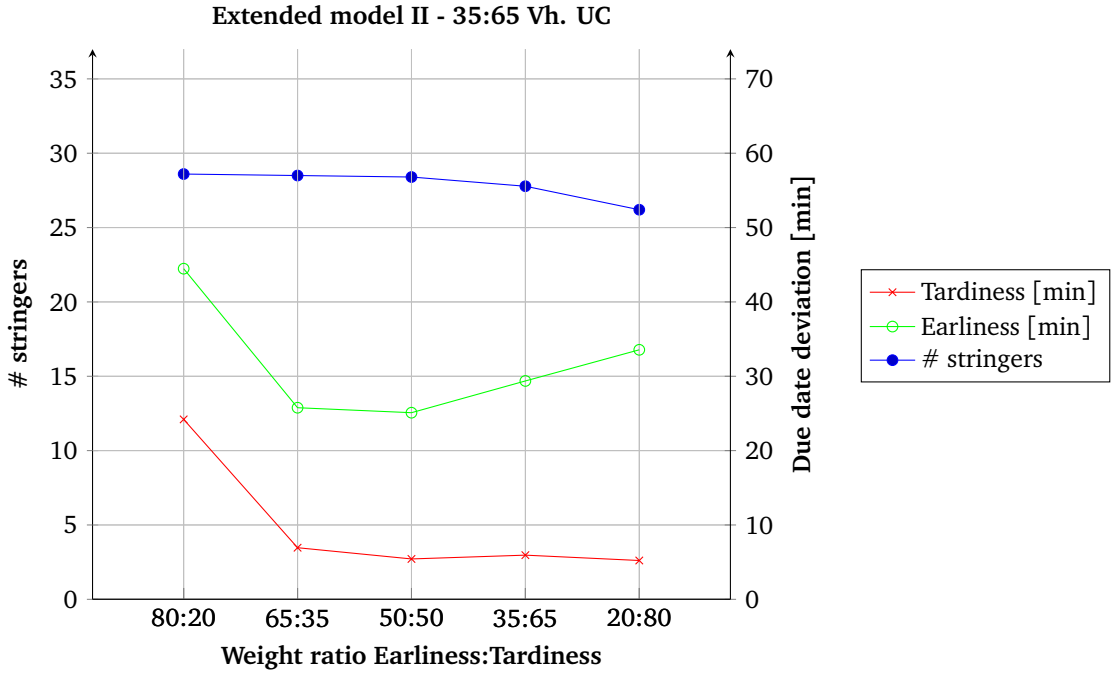

Figure 20: Trade-off between earliness and tardiness in minutes and its effect on the stringer use with a stringer and due date deviation weight of 35 % and 65 %, respectively, in the second extended model with a *Very high* UC.

could also be a random occurrence. This concludes the numerical studies, which will be followed by a brief summary of the results and their interpretations, as well as an outlook for future research and managerial insights.

5. Concluding remarks

5.1. Summary

Section 4.2 presented and discussed different aspects of solutions yielded by the Gurobi solver, as well as parameters that affected the scheduling of parallel heterogeneous CALs. This section will summarize the most important findings of this numerical study.

As observed in figure 2, the data density decreased with increasing instance sizes due to a surge of optimizations that had to be terminated without a feasible solution because of the implemented time limit. This diminished the ability to determine trends that involve higher instance sizes due to the mentioned selectivity effect, which affected the validity of the data. The tuning of the solver decreased the number of unsuccessful optimizations by almost 20 % while having no effect on the number of introduced stringers or delayed coils. It was, however, observed in figure 4, that the number of stringers per coil negatively correlates with the number of coils, which was explained by the higher probability of compatibility of coils with a greater instance size. Additionally, higher scheduling flexibility through higher service levels enabled solutions with fewer stringers, while also decreasing their total optimization time and MIPGap. Figure 6 displayed the surge of the total optimization time due to the increasing size and complexity of the model, as well as a slight increase through the tuning of the solver. A positive correlation between the MIPGap and the instance size was portrayed in 7, which was explained by the increasing number of terminations due to the time limit. It was also argued that a higher MIPGap does not necessarily indicate a less optimal solution, but primarily a deficiency in solving time. This deficiency was depicted in figure 8, as well as the substantial differences between different seeds, in terms of solving time. For the rest of the numerical study, the instance size of 40 coils in combination with the tuned solver was chosen due to their optimal trade-off between data density and the number of coils.

The results from the Best- and Worst-cases utilized in the second part of the numerical study, displayed in figures 9 to 12, were as expected. It was observed, that the parameterizations had a profound impact on the number of successful runs, stringer use and tardiness, as well as on the solving time. While optimizations with the Best-case and $Very\ high$ PF were able to schedule all coils without any stringers, none of the optimizations with the Worst-case and $Very\ high$ HC yielded any feasible solution. By altering the PF and HC respectively, the impacts could be mitigated, and it was found that both factors greatly influence the results of optimizations. An additional test proved that the impact of the HC on the solution is greater than that of the PF, at least for the used definitions of these factors.

In contrast, the impact of altering the number of processing lines on the stringer use was minimal. It was argued, that if no stringers have to be added to comply with the service level, the maximum absolute change in stringer use is one. In fact, the number of introduced stringers did not increase or decrease further for most optimizations through the removal or addition of a processing line, respectively, as depicted in figure 13. But it was observed in figure 14 that it did severely impact the solving times due to the reduction or extension of the model, respectively.

Another severe influence was observed in figure 16 by drastically increasing the UC. A negative correlation between the earliness of due dates and the number of introduced stringers and delayed coils was found, but most optimizations were not successful. The main reason was the drastically increasing share of infeasible models. But the data also portrayed the fact, that most models could still yield feasible solutions even with extremely early due dates, if they are provided with a higher service level and time limit.

By removing the service level and integrating the tardiness in the objective function, the first extended model was able to yield results similar to the base model with a medium service level. However, the former model had a higher UC, as portrayed in figure 17. Additionally, an almost one-to-one trade-off between the number of delayed coils and stringers was observed with $Very\ high$ UC. Figure 18 also indicated that the minimization of stringer use might be more complex than the minimization of tardiness through higher solving times with higher stringer weights.

With the introduction of earliness into the minimization objective, the trade-off between stringer use, tardiness, and earliness was displayed in figure 19 with the second extended model. The observations suggested that the minimization of earliness is more difficult than the minimization of tardiness and that it also correlates with a higher stringer use.

Furthermore, both extended models had significantly higher solving times than the base model, suggesting a higher complexity of the optimization process. In contrast, however, they also had a lower failure rate of zero %, indicating a negative impact of the service level on the optimizations conducted with the base model.

5.2. Outlook, further research and managerial insights

Each of the 1180 optimizations was conducted on a regular household computer. With more sophisticated hardware, much greater instance sizes could be solved while it would also increase the performance of each optimization already studied in this work. This could further be improved by the implementation of heuristics, as described in section 2.2 and allow for a higher number of runs per parameterization, thus improving the validity of the data. Additionally, a test could be conducted which combines the addition of processing lines with extremely early due dates. This should decrease the processing load per line and should therefore increase the feasibility of the optimizations and the optimality of the observed solutions. This test could also be conducted with the second extended model since it should benefit the earliness minimization and thus the stringer use. Moreover, the necessary solving time and quality of the resulting solution of optimizations with case 25 % from the third part of the numerical study could be determined.

But even without better hardware, the solver could be more fine-tuned to increase performance further. Moreover, the studies conducted on the base model during this numerical study could be applied to the extended models to investigate, if the results differ from each other and if so, why. Further extensions of the model are also possible. As an example, the release dates or due dates could be replaced with the entire upstream or downstream process, respectively. This would be one step further to scheduling the whole steel processing complex. Another study could try to investigate the

cause of the high differences between different seeds. Possible explanations could be either the different distributions of coil characteristics, due dates etc. or the randomness of the Gurobi solver if provided with different parameterizations.[56] Furthermore, the processing load of each processing line in different scenarios could be studied, which could have some implications for real-world applications. But some managerial insights can already be derived from the observations made in this study.

The impacts of the Worst- and Best-case scenarios should be considered when scheduling CALs. It is difficult to establish the Best-case in real-world scenarios because most factors are determined by the customer order and the manufacturer may not be able to influence them. But the Best-case should be established to the highest degree possible to minimize the incurring costs. Due to the decreasing relative number of stringers, it could also be advantageous to include as many coils in a schedule as possible, but it is not clear how this trend behaves with higher coil numbers. In contrast, the model building time increases disproportionately with the addition of more coils, which could pose a problem if optimization time is limited. Thus, it could be beneficial to organize all coils in smaller, homogeneous batches and put them in an order that minimizes the differences between each batch. Afterwards, each batch could be optimized individually, with the last coil on each line of the preceding batch posing as the first coil on that particular line for the succeeding batch. This could increase stringer use, but would also decrease the total optimization time. Thus, the costs of the additional optimization time would have to outweigh the possible costs caused by the introduction of additional stringers, which could be the case in situations with severely limited available computation time or low stringer cost, or both.

Furthermore, the data indicates that by minimizing both tardiness and stringer use, more optimal schedules could be found, which leads to the final managerial insights. The real-world applicability of the base model may be limited due to the consideration of only the number of delayed coils. While it is important how many coils are delayed, it is also important how long they are delayed. Since the longevity of the delay is not to be minimized in the base model, some coils could be delayed by an, for the customer, unacceptable amount of time. Hence, the second extended model may be more suitable for the scheduling of CALs. But it is also only a deterministic model. A deterministic model must be used complementarily with a dynamic model to account for random events since these naturally occur during operations. In addition, a CFM should be used to improve planning and efficiency over several periods, thus minimizing operational costs over the long term.[57]

Hence, the model and its different variants hold many possibilities for further research, which could result in more managerial insights as well as contribute to the development of a more refined model that could be implemented by a manufacturer to optimize the annealing process on continuous annealing lines.

[56] Miltenberger, 2023b

[57] Iannino et al., 2021, p. 620-630

References

Balaji, P. G., & Srinivasan, D. (2010). An introduction to multi-agent systems. *Studies in Computational Intelligence*, *310*. https://doi.org/10.1007/978-3-642-14435-6

Besson, T. (1998). *Simulation modeling as a tool for assessing the impact of inventory control and scheduling policies in the manufacturing of specialty steel*. Retrieved May 19, 2023, from http://hdl.handle.net/1721.1/50405

Commodity-Inside. (2019). *Flat Steel Market*. Retrieved June 3, 2023, from https://commodityinside.com/flat-steel-market/#:~:text=Cold%5C%20rolled%5C%20steel%5C%20is%5C%20used,which%5C%20require%5C%20good%5C%20surface%5C%20finishing.

Cowling, P., Ouelhadj, D., & Petrovic, S. (2004). Dynamic Scheduling of Steel Casting and Milling using Multi-agents. *Production Planning and Control*, *15*. https://doi.org/10.1080/09537280410001662466

de Harder, H. (n.d.). *Taking Your Optimization Skills to the Next Level*. Retrieved June 8, 2023, from https://towardsdatascience.com/taking-your-optimization-skills-to-the-next-level-de47a9c51167

Dong, Z., Wang, X., & Tang, L. (2021). Color-Coating Scheduling With a Multiobjective Evolutionary Algorithm Based on Decomposition and Dynamic Local Search. *IEEE Transactions on Automation Science and Engineering*, *18*(4), 1590–1601. https://doi.org/10.1109/TASE.2020.3011428

Durlinger, P. (2015). *Inventory and holding costs*. https://doi.org/10.13140/RG.2.1.3478.7684

Gao, C., Tang, L., & Wang, Y. (2008). Model and scheduling of a continuous galvanizing line. *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, *2*, 1829–1834. https://doi.org/10.1109/SOLI.2008.4682827

Gurobi. (n.d.-a). *GUROBI OPTIMIZER REFERENCE MANUAL*. Retrieved June 6, 2023, from https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf

Gurobi. (n.d.-b). *MIP Models*. Retrieved June 6, 2023, from https://www.gurobi.com/documentation/9.5/refman/mip_models.html

Gurobi. (n.d.-c). *Mixed Integer Programming Basics*. Retrieved June 6, 2023, from https://www.gurobi.com/resources/mixed-integer-programming-mip-a-primer-on-the-basics/

Harjunkoski, I., & Grossmann, I. (2001). A decomposition approach for the scheduling of steel plant production. *Computers & Chemical Engineering*, *25*, 1647–1660. https://doi.org/10.1016/S0098-1354(01)00729-3

Iannino, V., Colla, V., Maddaloni, A., Brandenburger, J., Rajabi, A., Wolff, A., Ordieres, J., Gutierrez, M., Sirovnik, E., Mueller, D., & Schirm, C. (2021). Improving the Flexibility of Production Scheduling in Flat Steel Production Through Standard and AI-Based Approaches: Challenges and Perspectives. In I. Maglogiannis, J. Macintyre, & L. Iliadis (Eds.), *Artificial Intelligence Applications and Innovations* (pp. 619–632). Springer International Publishing.

Jiyuan-Shenzhou-Industry. (n.d.). *Payoff reel*. Retrieved May 21, 2023, from http://www.jyszmachinery.com/product/Payoff-reel_1.html

Joint-Research-Centre. (2022). *EU climate targets: how to decarbonise the steel industry*. Retrieved May 20, 2023, from https://joint-research-centre.ec.europa.eu/jrc-news-and-updates/eu-climate-targets-how-decarbonise-steel-industry-2022-06-15_en

Karakostas, P., Sifaleras, A., & Georgiadis, M. (2019). A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing. *Computers & Chemical Engineering*, *126*, 263–279. https://doi.org/10.1016/j.compchemeng.2019.04.015

Karakostas, P., Sifaleras, A., & Georgiadis, M. (2020). Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. *Expert Systems with Applications*, *153*, 113444. https://doi.org/10.1016/j.eswa.2020.113444

Krukowska, E. (2021). *Europe CO2 Prices May Rise More Than 50% by 2030, EU Draft Shows*. Retrieved June 4, 2023, from https://www.bloomberg.com/news/articles/2021-06-29/europe-co2-prices-may-rise-more-than-50-by-2030-eu-draft-shows#xj4y7vzkg

Li, T., Meng, Y., & Tang, L. (2023). Scheduling of Continuous Annealing With a Multi-Objective Differential Evolution Algorithm Based on Deep Reinforcement Learning. *IEEE Transactions on Automation Science and Engineering*, 1–14. https://doi.org/10.1109/TASE.2023.3244331

Miltenberger, M. (2023a). *What is the MIPGap?* Retrieved June 5, 2023, from https://support.gurobi.com/hc/en-us/articles/8265539575953-What-is-the-MIPGap-

Miltenberger, M. (2023b). *Why does Gurobi perform differently on different machines?* Retrieved June 4, 2023, from https://support.gurobi.com/hc/en-us/articles/360045849232-Why-does-Gurobi-perform-differently-on-different-machines-

Mirjalili, S., & Gandomi, A. H. (2023). Chapter 21 - Employment of bio-inspired algorithms in the field of antenna array optimization: A review. In *Comprehensive Metaheuristics* (pp. 393–406). Academic Press. https://doi.org/10.1016/B978-0-323-91781-0.00021-1

Mujawar, S., Huang, S., & Nagi, R. (2012). Scheduling to minimize stringer utilization for continuous annealing operations. *Omega*, *40*(4), 437–444. https://doi.org/10.1016/j.omega.2011.08.007

Mujawar, S., Huang, S., Patel, D., & Nagi, R. (2004). Scheduling to Minimize Stringer Utilization in Metal Industries. *Institute of Industrial and Systems Engineers (IISE)*, *126*, 1–7.

Ouelhadj, D., Petrovic, S., Cowling, P., & Meisels, A. (2004). Inter-agent cooperation and communication for agent-based robust dynamic scheduling in steel production. *Advanced Engineering Informatics*, *18*, 161–172. https://doi.org/10.1016/j.aei.2004.10.003

Pan, Q.-K., Chen, Q.-d., Meng, T., Wang, B., & Gao, L. (2017). A mathematical model and two-stage heuristic for hot rolling scheduling in compact strip production. *Applied Mathematical Modelling*, *48*, 516–533. https://doi.org/10.1016/j.apm.2017.03.067

Pan, Q.-K., Gao, L., & Wang, L. (2019). A multi-objective hot-rolling scheduling problem in the compact strip production. *Applied Mathematical Modelling*, *73*, 327–348. https://doi.org/10.1016/j.apm.2019.04.006

Potts, C., & Strusevich, V. (2009). Fifty years of scheduling: A survey of milestones. *Journal of the Operational Research Society*, *60*, S41–S68. https://doi.org/10.1057/jors.2009.2

Sarna, S. K. (2013). *Annealing of Cold Rolled Steel*. Retrieved June 3, 2023, from https://www.ispatguru.com/annealing-of-cold-rolled-steel/

Sarna, S. K. (2014). *Colour Coating of Steels*. Retrieved June 3, 2023, from https://www.ispatguru.com/colour-coating-of-steels/

Steel-Warehouse. (n.d.). *Annealing Cold Rolled Steel*. Retrieved June 2, 2023, from https://www.steelwarehouse.com/annealing/#:~:text=Certain%5C%20hot%5C%20roll%5C%20grades%5C%20can,a%5C%20residual%5C%20stress%5C%2Dfree%5C%20product.

Takurou, M., Tatsuo, T., & Masayasu, O. (2016). Plant Optimal Control System for No. 5 Continuous Annealing Line (CAL) at West Japan Works (Fukuyama), JFE Steel. *JFE Technical Report*. Retrieved May 20, 2023, from https://www.jfe-steel.co.jp/en/research/report/021/pdf/021-22.pdf

Tang, L., & Meng, Y. (2021). Data analytics and optimization for smart industry. *Frontiers of Engineering Management*, *8*, 157–171. https://doi.org/10.1007/s42524-020-0126-0

Tang, L., & Wang, X. (2010). A Two-Phase Heuristic for the Production Scheduling of Heavy Plates in Steel Industry. *IEEE Transactions on Control Systems Technology*, *18*(1), 104–117. https://doi.org/10.1109/TCST.2009.2014960

Tasgetiren, M. F., Pan, Q.-K., Liang, Y.-C., & Suganthan, P. N. (2007). A Discrete Differential Evolution Algorithm for the Total Earliness and Tardiness Penalties with a Common Due Date on a Single-Machine. *2007 IEEE Symposium on Computational Intelligence in Scheduling*, 271–278. https://doi.org/10.1109/SCIS.2007.367701

Terence, B. (2020). *The Major Applications of Steel*. Retrieved May 23, 2023, from https://www.thoughtco.com/steel-applications-2340171

Twidale, S., Abnett, K., & Chestney, N. (2021). *EU carbon hits 100 euros taking cost of polluting to record high*. Retrieved May 23, 2023, from https://www.reuters.com/markets/carbon/europes-carbon-price-hits-record-high-100-euros-2023-02-21/

United-Enterprises. (n.d.). *Continuous Annealing Line (CAL)*. Retrieved May 21, 2023, from https://www.unient.co.in/all-products/Turnkey-Projects/continuous-annealing-line

Wegel, S., Ivanov, A., Lenz, R., & Volling, T. (2024). Scheduling of parallel continuous annealing lines with alternative processing modes to optimize efficiency under tardiness constraints. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2023.12.032

Williamson. (n.d.). *Annealing Line Application Overview*. Retrieved May 23, 2023, from https://www.williamsonir.com/wp-content/uploads/2018/06/Annealing_Line_Application_Note.pdf

Zhang, B., & Yang, Y. (2014). Solving production scheduling with chain constraints in parallel production lines by column generation. *Proceeding of the 11th World Congress on Intelligent Control and Automation*, 800–803. https://doi.org/10.1109/WCICA.2014.7052818

Zhao, S., & Yang, Y. (2016). Differential evolution algorithm for solving coil scheduling problem in parallel continuous annealing lines. *2016 Chinese Control and Decision Conference (CCDC)*, 3417–3422. https://doi.org/10.1109/CCDC.2016.7531573