h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Hochschule Darmstadt

Fachbereich Mathematik und Naturwissenschaften
&
Fachbereich Informatik

Exact Methods and Heuristic Approaches for Setup Minimization of One-Dimensional Cutting Stock Problems

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M. Sc.)
im Studiengang Data Science

vorgelegt von

Jan-Erik Justkowiak

Referent:	Prof. Dr. Julia Kallrath (Hochschule Darmstadt)
Korreferent:	Prof. Dr. Tobias Bedenk (Hochschule Darmstadt)
Ausgabedatum:	15. Oktober 2018
Abgabedatum:	18. März 2019

Jan-Erik Justkowiak
Römerstraße 38
64291 Darmstadt-Arheilgen

Hochschule Darmstadt
Fachbereich Mathematik und Naturwissenschaften
Matrikelnummer: 736604

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 18. März 2019

Zusammenfassung

Das eindimensionale Zuschnittproblem mit einer minimalen Anzahl von unterschiedlichen Schnittmustern gehört zur Klasse der schwer lösbaren Probleme. Neben der Formulierung als reines gemischt-ganzzahliges nichtlineares Optimierungsproblem werden Lösungen durch exakte Methoden, z. B. Branch & Price oder durch Heuristiken, z. B. Ausschöpfungsmethoden, ermittelt. Der Nachteil dieser Verfahren ist der hohe Implementierungsaufwand.

Alternativ zu diesen Methoden präsentieren wir mehrere neuartige und leicht implementierbare (praxistaugliche) Modellformulierungen und Heuristiken zur Lösung des eindimensionalen Verschnittproblems mit Musterminimierung und exakter Erfüllung der Nachfrage. Im Gegensatz zu den meisten Ansätzen in der Literatur verzichten wir auf die Verwendung von ganzzahligen Variablen und nichtlinearen Termen in der Modellierung oder speziellen Lösungsstrategien wie Column Generation und Branch & Price. Stattdessen verbessern wir die Ergebnisse und die Laufzeit, indem wir - neben der Entwicklung einer performanten Modellformulierung durch die ausschließliche Verwendung von Binärvariablen - problem-spezifische Eigenschaften ausnutzen. Zusätzlich präsentieren wir einen heuristischen Greedy-Algorithmus zur Generierung von Schnittmustern. Dabei werden inkrementell solche Schnittmuster erzeugt, die eine maximale Anzahl von Aufträgen exakt erfüllen. Darüber hinaus stellen wir einen Ansatz zur Partitionierung einer komplexen Probleminstanz vor. Anhand der Nachfragemengen werden zwei abgeleitete Unterinstanzen gebildet, die auf Grundlage einer monolithischen Modellformulierung in der Regel erheblich leichter gelöst werden können.

Die Modelle und Heuristiken wurden in `GAMS` implementiert und mit `GAMS/CPLEX` gelöst. Eine breite Referenzmenge von Probleminstanzen aus der Literatur bildet die Basis der numerischen Experimente. Dabei unterscheiden wir zwischen 80 Probleminstanzen aus realen Anwendungen in der Industrie sowie 1980 zufällig generierten (`CUTGEN`) Instanzen. Durch die Verwendung der monolithischen Modellformulierungen können wir optimale Lösungen für alle Probleminstanzen aus den realen Anwendungen bestimmen. Darunter sind mehrere Instanzen, für die in der Literatur bislang keine Optimallösungen bekannt sind. Die von unserer robustesten Heuristik bestimmten Lösungen werden durchschnittlich in wenigen Sekunden ermittelt und sind dabei im Durchschnitt um nur 13 % schlechter (bzgl. der Anzahl der Schnittmuster) als die Optimallösungen (die Berechnungsgrundlage dieser Bewertung bilden 1060 Optimallösungen, die zuvor unter Nutzung der monolithischen Modellformulierungen - erstmalig in der Literatur - bestimmt werden konnten). Im Vergleich mit den optimalen Lösungen für die Instanzen aus den industriellen Anwendungen sind die heuristischen Lösungen in absoluten Zahlen im Allgemeinen um höchstens ein oder zwei Schnittmuster schlechter (durchschnittlich um 0,8 Schnittmuster).

Schlagwörter: Zuschnittproblem, Papierindustrie, exakte Optimierung, heuristische Verfahren, Musterminimierung, Verschnittminimierung

Abstract

Minimal number-of-pattern solutions of the one-dimensional *cutting stock* problem (which is known to be strongly NP-hard) have - beyond mixed integer nonlinear programming - been solved by exact methods, *e. g.*, Branch & Price, or heuristics like exhaustion methods. Both suffer from high implementation efforts.

Alternatively, we present some novel and easy to implement model formulations and heuristic approaches in order to formulate and solve the one-dimensional cutting stock problem for setup minimization with exact demand fulfilment, being suitable for industrial applications. In particular, we have developed some exact linear models using only binary variables, *i. e.*, in contrast to most approaches in literature we avoid integer variables, non-linear terms and special solution techniques like column generation and Branch & Price. Instead, we improve the performance by exploiting problem-specific properties. In addition, we have developed a Greedy algorithm with iterative pattern generation, where we try to maximize the number of order widths being fulfilled exactly by the currently generated pattern and an approach for splitting the original problem instances into smaller sub-instances based on the demand levels, which are more likely to be solvable by a monolithic formulation.

The models and heuristics have been implemented in `GAMS` and solved with `GAMS/CPLEX`. Our approaches have been tested on a broad range of benchmark instances used in literature, including 80 real-world instances from industrial applications as well as 1980 randomly generated `CUTGEN` instances. By solving the monolithic model formulation, we are able to calculate and prove optimal solutions for all real-world instances. For some of these instances, no optimal solutions have been known so far. The most stable heuristic being proposed provides solutions within seconds on average, being only 13 % worse in terms of pattern count compared to the optimal solutions on average (this evaluation is based on 1060 instances being solved to optimality - for the first time in literature - by applying the monolithic model formulations). Compared to the optimal solutions for the instances from the industrial applications, these heuristic solutions are in general at most one or two patterns worse (on average by 0.8 patterns).

Keywords: cutting stock problem, paper industry, exact optimization, heuristic algorithms, setup minimization, cutoff reduction, trimloss minimization

Acknowledgement

Foremost, I would like to thank my thesis advisor Prof. Dr. Julia Kallrath (Darmstadt University of Applied Science) for the excellent mentoring. Her friendly guidance and helpful expert advice in all the time of research on and writing of this thesis was invaluable. Besides from supervising this thesis, Prof. Julia Kallrath offered me the opportunity to attend as undergraduate assistant on several lectures for many years, including operations research. In addition, Prof. Dr. Julia Kallrath and Prof. Dr. Josef Kallrath (University of Florida, Gainesville) supported me on the first steps with regard to the postgraduate period.

Special thanks are due to Prof. Dr. Josef Kallrath for his feedback and suggestions on the manuscript, but more important I have gained a broader and deeper understanding and insights on modeling and on solving complex optimization problems just from being in contact from time to time.

In addition, I would like to thank GAMS Software GmbH (Frechen) for providing a free GAMS license.

I would also wish to express my gratitude to Prof. Dr. Werner Helm (Darmstadt University of Applied Science). Even though we were barely in touch during my studies in the master program for the last years, Prof. Helm has always supported and encouraged me during my bachelor program, also introducing me to the field of operations research at all and supervised me during my internship and as well as during my bachelor thesis.

Last but not least, I would like to thank the co-advisor Prof. Dr. Tobias Bedenk (Darmstadt University of Applied Science). The participation in his lecture on mixed integer linear programming was quite beneficial with respect to the topic of this thesis.

Contents

1. Introduction

With an annual production volume of 411 million tons in 2016, the pulp and paper industry is an important economical sector worldwide. Germany is the fourth largest producer of paper and paperboard with an annual production of 22.6 million tons in 2016, close behind Japan with a production of 26.3 million tons. The largest producers (by nations in 2016) are the United States of America with an annual production of 72.1 million tons and China with 111.3 million tons. The German pulp and paper industry employs 39800 people, generating a total annual revenue of 14.24 billion Euro in 2016 (see also [22]).

Cost-saving and environment-friendly production techniques and the planning of efficient operating processes are of great importance for every company. With respect to the pulp and paper industry, effective solution techniques for frequently occurring blending and cutting stock problems are of special interest. The latter is the problem of cutting standardized paper-rolls into quantities (demands) of smaller rolls (items/order widths) being requested by customers, while minimizing the material usage, *i. e.*, the quantities of paper-rolls being cut resp. while minimizing the waste (cutoff reduction), leading to cost- and resource-saving, environment-friendly production processes. This problem involves the construction of cutting patterns while determining the application quantities on the paper-rolls for each pattern. However, it does not consider the costs when changing from one pattern to another (setup costs) by re-adjusting the knife arrangement. The problem of reducing the setup costs by using as few different patterns as possible is known as setup minimization problem in literature.

Note that we introduced and motivated the cutting stock problem (with setup minimization) from the perspective of the pulp and paper industry, but their applications are not limited to this specific industrial sector. The problem of cutting large rolls into smaller items also occurs in the steel- or plastic film industry for instance.

In this thesis, we present several novel model formulations and heuristic approaches in order to solve the setup minimization problem with exact demand fulfilment. Additionally, cutoff reduction or trimloss minimization is observed as secondary objective. Following the literature review, we will present the contents of this thesis in detail.

Literature Review

Cutting stock problems have long been investigated in the operations research field and are among the most prominent optimization problems, as they are hard to solve, while providing good solutions is of great practical interest.

Next to the rich body of literature focussing on the cutoff minimization problem, *e. g.*, the well-known paper of Gilmore and Gomory, presenting a comlumn generation approach (1961, see [1]), there are several publications tackling the setup minimization problem, also referred as pattern reduction, which is known to be significantly more complex compared to the cutoff minimization problem.

Foerster and Wäscher proposed a two-step heuristic approach (2000, see [21]). Initially, the cutoff minimization problem is solved without observing the number of patterns, which is reduced afterwards by iteratively combining the patterns of the obtained solution.

An exact algorithm for minimizing the number of patterns is proposed by F. Vander-

beck (2000, see [3]). He developed a Branch & Price & Cut procedure involving a column generation approach in order to solve the pattern minimization problem being formulated as integer quadratic linear programming problem.

S. Umetani, M. Yagiura, T. Ibaraki presented an iterated local search algorithm with adaptive pattern generation (2003, [7]). The heuristic is used to search a best solution for a fixed number of patterns that minimizes the quadratic deviation of the produced items compared to the demand levels.

G. Belov and G. Scheithauer developed a Branch & Price algorithm in order to solve the setup minimization problem being formulated as an integer programming model (2003, see [8]).

R. Johnston and E. Sadinlija presented an exact model formulation resolving the non-linearity in the `MINLP` formulation of the cutting stock problem by the novel use of binary variables (2004, see [4]).

A hybrid heuristic in order to reduce the number of different patterns in cutting stock problems is proposed by H. Yanasse and M. Limeira (2006, see [14]). Initially, good patterns are generated with limited waste that fulfil the demands of at least two items. Afterwards, pattern reduction techniques are applied (in particular, the combining heuristic proposed by Foerster and Wäscher).

Y. Cui, X. Zhao, Y. Yang and P. Yu present a sequential heuristic algorithm for pattern reduction (2008, see [18]). A pattern is generated using a subset of unassigned items, the pattern multiplicity is determined and the assigned items are deleted from the set of unassigned items. The procedure is repeated, until all items are assigned. The subset is determined such that the multiplicity of the generated pattern can be as large as possible, observing some constraint to keep the cutoff reasonable.

G. Cerqueira and H. Yanasse developed a heuristic pattern reduction procedure by grouping items according to their demands (2009, see [15]). The items are separated in two disjoint groups in the beginning. Afterwards, patterns are generated for each group and those with limited cutoff are accepted. Finally, a residual problem observing the items which are not satisfied yet is solved and a pattern reduction procedure of literature is applied.

J. Kallrath, S. Rebennack, J. Kallrath and R. Kusche proposed an exhaustion method, involving a pattern generation heuristic and `MILP` techniques (2014, see [5]). Depending on the number of orders, either the model by Johnston and Sadinlija is used directly or they successively generate new patterns with maximum multiplicities and low waste. In case there is only a small number of orders left, the model of Johnston and Sadinlija is solved.

E. Banbal and J. Kallrath presented numerical results for solving the setup minimization problem with exact demand fulfilment formulated as `MINLP` using different solvers (2015, see [6]). In addition, the authors proposed two decomposition approaches in order to solve smaller sub-instances of lower complexity derived of the original problem instance based on some characteristics like the demand levels or the widths ordered.

A two-stage heuristic for calculating solutions with reduced setup and material cost is presented by Y. Cui, C. Zhong, Y. Yao (2015, see [19]). Initially, a set of at most 5000 patterns is generated. An integer linear programming model to minimize the sum of material and setup costs (by pattern reduction) is solved afterwards.

In summary, the setup minimization problem is often formulated as `MINLP` or `MILP`, both hard to solve for commercial solvers. Therefore, acceptable results in terms of solution time and solution quality (number of patterns) can be attained only by applying

specific and sophisticated solution strategies like Branch & Price & Cut methods or by using heuristic approaches, in general at the cost of high implementation effort.

Note that most authors investigate a problem definition allowing for overproduction while observing the cutoff simultaneously or by adding a restriction on the material usage based on the optimal solution of the cutoff minimization problem, deviating from our pure setup minimization problem with exact demand fulfilment (and cutoff reduction as secondary objective). This will be taken into account when evaluating the results.

Own Contributions and Structure of this Thesis

We present several novel exact linear model formulations for the setup minimization problem with exact demand fulfilment in Section 2.1. Based on these formulations, instances of low or medium complexity can be solved to optimality in reasonable time. In order to obtain a general performance improvement, we use binary variables only, while reducing the resulting high number of variables by exploiting problem specific properties, *e. g.*, we apply tighter bounds on the pattern multiplicities being derived by the demand levels, leading to a significantly reduce in terms of solution time. A further benefit besides the fast implementability with an algebraic modeling language is the easy adaptability of our models, *i. e.*, additional requirements can be included straightforward (*e. g.*, diverging objectives or further constraints).

In order to solve problem instances with increasing complexity, we present several novel and easy to implement heuristic approaches Section 2.2. These approaches are characterized by a transparent top-level conception. In particular, we are repeatedly solving minor complex models by the comfortable usage of an algebraic modeling language combined with an optimization solver (`CPLEX`). For instance, a greedy algorithm being proposed generates in each iteration a pattern which maximizes the number of orders being fulfilled exactly, while also observing the already generated patterns within the decision process, as one might add further widths to those patterns. This task will be realized best and most comfortable by formulating an appropriate model (which is of low complexity) being solved in each iteration to optimality.

As the requirement of pure setup minimization with exact demand fulfilment often leads to solutions with high material consumption, most of the approaches will be modified in Section 2.3 in order to observe the minimization of material consumption as secondary objective, too.

Next to an evaluation of our approaches based on numerical results calculated on a broad range of problem instances frequently used in literature (see Section 3.1 and Section 3.3), we also give some theoretical insights in relation with our approaches, *e. g.*, we prove upper and lower bounds on the pattern multiplicity or that it is sufficient to solve the relaxed variant of a model. Finally, we summarize the results while giving an assessment of which approach should be applied based on the problem data characteristics and present the current and further research in Chapter 4.

2. Cutting Stock Problems - Setup Minimization

In the beginning of this Chapter, we describe the one-dimensional cutting stock problem for setup minimization, also referred as pattern reduction. In addition, we introduce the most basic notations (*i. e.*, parameter and sets) in order to define the monolithic model formulations and heuristic approaches later on. Afterwards, we demonstrate the setup minimization problem on a small instance as an introductory example.

The monolithic model formulations are described in Section 2.1, among them the well-known standard `MINLP` (Mixed Integer Non Linear Programm), a linear model based on complete pattern generation and a novel linearisation of the standard `MINLP` using binary variables only while exploiting some problem specific properties in order to reduce the problem complexity in terms of model size and solution time.

Several novel heuristic approaches are discussed in the Section 2.2, among them varying versions of a `Greedy` algorithm and an approach for splitting the original problem instance into smaller derived sub-instances in order to get a heuristic solution of the original instance by solving the smaller sub-instances (to optimality).

Finally, the model formulations introduced in Section 2.1 and most of the heuristic approaches presented in Section 2.2 are modified in order to consider the number of master rolls being cut as secondary objective (Section 2.3).

Problem Description

The general cutting stock problem (see [1] or [13] for reference) can be described as follows: Master rolls (see Figure 2.1 for instance) of identical width and infinite length shall be cut in an efficient way to fulfil demand of smaller order widths.

http://www.badische-zeitung.de/bzcard-leserfahrten/papier-fuer-die-badische-zeitung–46612744.html

Figure 2.1.: Master-roll in the paper industry.

This results in varying objectives, for example, reducing the waste of master rolls with no further use, which accrues during the cutting process, in order to reduce the cost of materials and to protect the environment (known as cutoff reduction).

The width of the master roll (there is only a single type of master roll), the number of knifes which are used to cut the master rolls in smaller items resp. smaller rolls, the quantity of orders (demands) and the widths ordered are assumed to be known. Figure 2.2 illustrates a master roll being cut into smaller items/rolls by applying a pattern (which is defined by the item multiplicities) using a specific knife arrangement (the knifes are located at the positions 1, 2, 3 and 4). The red striped area represents cutoff.

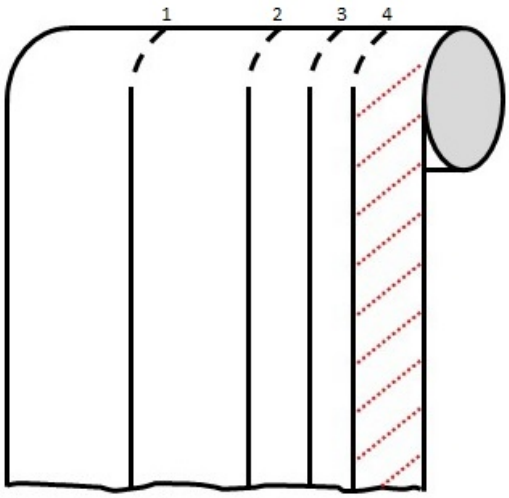

Figure 2.2.: Schematic pattern resp. setup representation.

In the setup minimization or pattern reduction problem, the setup costs of cutting machines shall be minimized by using as few different patterns as possible. When minimizing the number of setups, we have to treat the case, in which demand has to be fulfilled exactly, *i. e.*, no over- or underproduction of order widths is allowed. This condition is reasonable, since allowing over- or underproduction while not observing the cutoff resp. the over- or underproduction of orders simultaneously when minimizing the number of setups has a major impact on the problem complexity (solution times decrease while the solutions get impractical in general). For instance, if we allow unlimited overproduction, the solution to the setup minimization problem will be close to its lower bound derived in Note 1 later on in this Chapter, while there will be an enormous overproduction of nearly all orders (except for the one with the largest demand). Moreover, in the worst case, we may not have any further use of the overproduction afterwards. However, note that some authors allow for a small range of over- or underproduction in literature.

In order to formulate the setup minimization problem described as above, we introduce the following sets and parameters:

- Set of patterns $p \in \mathcal{P} := \{p_1, \ \ldots, \ p_{|\mathcal{P}|}\}$.
- Set of orders $i \in \mathcal{I} := \{i_1, \ \ldots, \ i_{|\mathcal{I}|}\}$.
- Demand D_i sorted in descending order and width w_i of order $i \in \mathcal{I}$. We assume $w_i \neq w_{i'}, \ \forall i, i' \in \mathcal{I}$ with $i \neq i'$.
- Width W of the master rolls and number K of knifes.

- Pattern multiplicity $k \in \mathcal{K} := \{1, \ldots, \max_{i \in \mathcal{I}} D_i\}$. There is no pattern which will be used more than $\max_{i \in \mathcal{I}} D_i$ times, since overproduction shall be avoided (exact demand fulfilment).
- Item multiplicity $n \in \mathcal{N} := \{1, \ldots, \min[\max_{i \in \mathcal{I}} \lfloor W/w_i \rfloor ; K]\}$ of a width i in a pattern p. The maximum item multiplicity is restricted by the available number of knifes or by the greatest number of rolls of an order width which can be cut from the master roll.

The cardinality of $\mathcal{P}$ depends on the model formulation. In case of pattern generation by complete enumeration, $|\mathcal{P}| = \#$possible patterns. Without generation of all possible patterns, $|\mathcal{P}|$ can be set to $|\mathcal{P}| = |\mathcal{I}|$ (in the worst case, the demand of each order width will be fulfilled by a different setup/pattern).

A valid solution of the one-dimensional cutting stock problem is denoted by (the parameters) $u_p \in \mathbb{Z}_0^+$, describing the multiplicity of pattern p (*i. e.*, the quantities of master rolls being cut according to pattern p) and by $a_{ip} \in \mathbb{Z}_0^+$, indicating how many pieces of width i (item multiplicity) are cut from a master roll when applying the pattern p. The term z refers to the number of different patterns. The quantity of master rolls being cut is denoted by m.

Note that some additional sets, parameters etc. are introduced later on. However, the ones presented above are the most basic ones and will be used in most of the models and heuristics. A complete overview on the notation is presented in Appendix A.

Example 1 (Solving a small Instance) *Given an instance $I = (\mathcal{I}, D_i, w_i, K, W)$ of the one-dimensional cutting stock problem, we will demonstrate how to find an optimal solution for the pattern reduction problem analytically. There are four order widths with width w_i and demand D_i, summarized in the following table. The width of the master rolls is given by $W = 1000$, the number of knifes is $K = 7$.*

Basic Data

i	w_i	D_i
i_1	*100*	*60*
i_2	*200*	*30*
i_3	*500*	*20*
i_4	*300*	*10*

At first, we will calculate a lower bound on the minimal number z^{opt} of different patterns. Since not all order widths can be combined in a single pattern, i. e., $\sum_{i \in \mathcal{I}} w_i = 1100 > 1000 = W$, we obviously need more than a single pattern in order to fulfil the demand for all order widths (exactly). However, two patterns are probably sufficient. We will now construct a solution with two different patterns by looking closely.

Obviously, one can combine the order widths in several ways (two of them presented below), each using two patterns. Since two is an lower bound on z^{opt}, both solutions are optimal with respect to the number of patterns. Note that we have to cut 30 master rolls in both solution in order to apply those patterns, yet the second one is more desirable, since the knifes must be re-adjusted only for the orders i_3 and i_4 when switching from the first pattern to the second one. However, this additional request is not considered in this thesis.

	p_1	p_2	
u_p	*20*	*10*	D_i
$w_1 = 100$	*3*	*0*	*60*
$w_2 = 200$	*0*	*3*	*30*
$w_3 = 500$	*1*	*0*	*20*
$w_4 = 300$	*0*	*1*	*10*
width	*800*	*900*	
cutoff	*200*	*100*	

	p_1	p_2	
u_p	*20*	*10*	D_i
$w_1 = 100$	*2*	*2*	*60*
$w_2 = 200$	*1*	*1*	*30*
$w_3 = 500$	*1*	*0*	*20*
$w_4 = 300$	*0*	*1*	*10*
width	*900*	*700*	
cutoff	*100*	*300*	

For instance, in the first pattern p_1 of the solution on the left, we will cut the first order i_1 three times and the third order i_3 a single time out of the master roll. The first pattern p_1 will be used 20 times. The width of the master roll and the number of knifes is not exceeded. As stated above, both solutions not only require two different patterns (primary objective), but also the same amount m of master rolls (30 pieces) have to be used in order to apply those patterns for exact demand fulfilment. In general, there are often several optimal solutions with varying values of m. Minimizing the number of master rolls as secondary objective is discussed in Section 2.3. The solution below represents the optimal solution (25 pieces) for the cutoff reduction problem.

	p_1	p_2	
u_p	*20*	*5*	D_i
$w_1 = 100$	*3*	*0*	*60*
$w_2 = 200$	*1*	*2*	*30*
$w_3 = 500$	*1*	*0*	*20*
$w_4 = 300$	*0*	*2*	*10*
width	*1000*	*1000*	
cutoff	*0*	*0*	

While the solution above is optimal for both objectives, note that minimizing the number of master rolls and the number of different patterns is contrary in general.

The procedure of determining a lower bound z^{lo} on the optimal solution z^{opt}, demonstrated in the previous example, can be generalized.

Note 1 (Lower Bound on the Number of Patterns) *The sum of all order widths divided by the length of the master roll defines a lower bound z^{lo} on the number of patterns, i. e., $z^{lo} \leq z^{opt}$ for $z^{lo} := \lceil \sum_{i \in \mathcal{I}} w_i / W \rceil$. The lower bound on the number of different patterns can be used to evaluate the goodness of the results for instances which are solvable only by heuristic approaches, i. e., it is possible to calculate the worst case gap for a given solution z. Calculating the lower bound as demonstrated above can also be seen as* `Bin Packing`*, where the master rolls corresponds to the bins, the demand levels are equal to one for all order widths and the widths w_i represents the items to be packed into (a minimal) number of bins.*

2.1. Exact Model Formulations

In this Section, we introduce some exact model formulations for the one-dimensional cutting stock problem with pattern reduction. In particular, we present the well-known MINLP in the first Subsection 2.1.1. Afterwards, we develop a linear model formulation based on complete pattern generation (Subsection 2.1.2). Additionally, the integer variables (expressing the multiplicity of a pattern resp. the item multiplicity of a width in a pattern in the MINLP formulation) are substituted by a set of binary variables. The complete pattern generation approach might not be applicable for instances with many small widths, as the number of possible combinations of the orders in a single pattern might be far too large. In Subsection 2.1.3, we develop a novel linear model by using binary variables only without complete pattern generation in advance. This formulation can be seen as an extension of the linear model in [4], but it extends the usage of binary variables, as we have seen a significant further decrease in terms of solution time. Furthermore, we will exploit some problem specific properties in order to reduce the large number of binary variables and the solution time.

In Section 2.1.4, we will present novel model formulations for the pure setup minimization problem with exact demand fulfilment and multiple types of master rolls.

Solving instances with optimization solvers like CPLEX or GUROBI based on exact model formulations has many advantages in general; the models are easy to understand and can be implemented quite fast and comfortable with an algebraic modelling language like GAMS. In addition, the models can be adjusted easily in order to meet extensions or modifications of the problem definition, *e. g.*, by adding further constraints. At the end of the solution process, one will get an optimal solution theoretically or detect infeasibility. Additionally, the solver reports a bound, *i. e.*, the goodness of the current solution can be evaluated if the solution process is terminated before finding an optimal solution resp. before proving the optimality of the solution. The major drawback of this overall approach is the solution time for instances with increasing complexity, since the number of variables and constraints increases dramatically (in addition to the strongly NP-Hardness of the pattern reduction problem [12]). In the worst case, finding an optimal or near optimal solution in reasonable time is not possible. Sometimes, the solver will not even find a feasible point or the model generation terminates due to an out of memory abort.

However, we have seen remarkable results when solving the linear models compared to the results in [5] and [6] on some real world cutting stock problem data. In addition, approximately 10 out of 30 benchmark classes investigated in this thesis (each consisting of several instances) can be solved to optimality within one hour by using the monolithic model formulations proposed in Subsection 2.1.2 and 2.1.3.

2.1.1. Mixed Integer Non Linear Model Formulation

The setup minimization or pattern reduction problem is often formulated as MINLP, among others in [3], [5] and [6]. In order to formulate the problem, three kinds of binary resp. integer variables are used:

- Multiplicity $\mu_p \in \mathbb{Z}_0^+$ of pattern $p \in \mathcal{P}$, *i. e.*, the number of master rolls which are cut according to pattern p.
- Item multiplicity $\alpha_{ip} \in \mathbb{Z}_0^+$ of order width $i \in \mathcal{I}$ in pattern $p \in \mathcal{P}$.

- Pattern usage indicator $\delta_p \in \{0,1\}$ of pattern $p \in \mathcal{P}$, *i. e.*, if pattern p is used at least once ($\mu_p > 0$), the binary variable δ_p is equal to one (zero otherwise).

Using the previous variables, we formulate the following `MINLP` (`MINLP1` - `MINLP8`):

Mixed **I**nteger **N**on **L**inear **P**rogram - `MINLP`:

$$\min \quad z = \sum_{p \in \mathcal{P}} \delta_p \qquad \text{(MINLP1)}$$

$$\begin{aligned}
\text{s.t.} \quad \sum_{p \in \mathcal{P}} \alpha_{ip} \mu_p &= D_i && \forall i \in \mathcal{I} && \text{(MINLP2)} \\
\sum_{i \in \mathcal{I}} w_i \alpha_{ip} &\leq W && \forall p \in \mathcal{P} && \text{(MINLP3)} \\
\sum_{i \in \mathcal{I}} \alpha_{ip} &\leq K && \forall p \in \mathcal{P} && \text{(MINLP4)} \\
\delta_{p+1} &\leq \delta_p && \forall p \in \mathcal{P} \setminus \{p_{|\mathcal{I}|}\} && \text{(MINLP5)} \\
\mu_{p+1} &\leq \mu_p && \forall p \in \mathcal{P} \setminus \{p_{|\mathcal{I}|}\} && \text{(MINLP6)} \\
\mu_p &\leq M \delta_p && \forall p \in \mathcal{P} && \text{(MINLP7)} \\
\delta_p \in \{0,1\},\ \mu_p \in \mathbb{Z}_0^+,\ \alpha_{ip} &\in \mathbb{Z}_0^+ && && \text{(MINLP8)}
\end{aligned}$$

The objective function `MINLP1` is to minimize the number z of (different) patterns, which are used in the current cutting plan. Constraint `MINLP2` enforces exact demand fulfilment (note the non linear terms in this constraint). The total pattern width cannot exceed the width W of the master roll (`MINLP3`). The number of smaller rolls which are cut from the master roll is restricted by the number K of knifes (`MINLP4`). However, if the sum of all widths in a pattern is equal to the width of the master roll, one can get $K + 1$ items of course (but this rare case is not considered further on). Finally, the symmetry breaking constraints `MINLP5` and `MINLP6` are added to the model for numerical improvement by reducing the search space. In particular, constraint `MINLP5` enforces that the next pattern $p + 1$ can only be used if pattern p is already used, while `MINLP6` sorts the patterns according to their multiplicity in descending order. The variables δ_p and μ_p are connected in constraint `MINLP7`, where M is a sufficient large number. In particular, μ_p greater than zero enforces δ_p to be equal to one, while δ_p is free, if μ_p is equal to zero. Since we are minimizing the sum over δ_p in the objective, the value of δ_p will be equal to zero in the second case.

Note 2 (Number of Knifes) *Note that only a very small amount of instances in the benchmark sets have a restriction on the number of knifes (in particular, only the instances of the* `Kallrath` *benchmark set). However, we include the constraint of observing the number of knifes in the problem description, as the model formulation becomes more general and realistic. In addition, the constraint can be easily turned off for instances without a restriction by defining $K = \infty$ in the model formulation. For numerical reasons, we will simply omit the constraint in the model implementations, if there is no restriction on the number of knifes.*

While the formulation presented above is quite compact and easy to implement with an algebraic modelling language (*e. g.*, `GAMS`), the numerical performance when solving the

model with strong global MINLP solvers like BARON is acceptable only for small instances like C5, C6 or C7 of the Kallrath benchmark set with up to seven order widths and low demand levels in general. For instances with increasing complexity (being characterized by an increase in terms of $|\mathcal{I}|$ and higher quantities of demands D_i), the runtimes increase significantly, often without finding optimal solutions or proving optimality. In case of the hardest instances C49 and C50 of the Kallrath benchmark set, no feasible solution has been found within a day (see [6]).

In the next Section, we eliminate the non linear term in the demand fulfilment equation in order to reduce the solution time by generating all possible patterns in advance. Note that generating all patterns is impractical for many instances, due to the large amount of possible patterns. However, to our surprise, with today's computer technology one can solve about a quarter of all instances considered in this thesis to optimality with the complete pattern generation approach (see Chapter 3 for the numerical results and the system properties).

2.1.2. Binary Linear Model based on complete Pattern Generation

Initially, we define a model whose feasible resp. optimal solution represent a valid pattern. By using the CPLEX Solution Pool Option, all valid patterns resp. solutions of the model can be determined (theoretically). In order to formulate the problem, we define the following integer variable:

- Item multiplicity $\gamma_i \in \mathbb{Z}_0^+$ of the order i in the currently generated pattern.

Obviously, the item multiplicity of i is restricted by the number of times the width can be cut from the master roll, by the number of knifes and by the demand level D_i of the width i, *i. e.*, $\gamma_i \leq \min(\lfloor W/w_i \rfloor; K; D_i),\ \forall i \in \mathcal{I}$. By the use of the variable γ_i, we will define the following MILP (PG1 - PG4):

Pattern **G**eneration - PG:

$$\begin{aligned}
\min \quad & z = 0 && && \text{(PG1)}\\
\text{s.t.} \quad & \sum_{i\in\mathcal{I}} w_i\gamma_i && \leq \quad W && \text{(PG2)}\\
& \sum_{i\in\mathcal{I}} \gamma_i && \leq \quad K && \text{(PG3)}\\
& \gamma_i && \in \quad \mathbb{Z}_0^+ && \text{(PG4)}
\end{aligned}$$

As we only seek to generate a valid pattern which corresponds to a feasible solution of PG, we define a constant dummy objective in PG1. Constraints PG2 and PG3 are the pattern generation constraints, in particular, PG2 observes the width of the master roll, while PG3 observes the number of knifes (in general, a master roll cannot be cut in more than K items).

All generated patterns are used as input parameters a_{ip} (item multiplicity of width i in pattern p) in the next model (BLMPG) in order to eliminate the non linear term in demand fulfilment equation. However, the solution times of the resulting MILP are still to high for most instances and practical usage. Substituting the integer variables μ_p, introduced in the previous Subsection, with the sum over $|\mathcal{K}|$ binary variables δ_{kp}, while multiplying

each variable with the index k, finally results in a considerable reduction of solution time. In particular, the variable δ_{kp} is defined by

$$\delta_{kp} := \begin{cases} 1, & \text{if } p \text{ is used exactly } k \text{ times} \\ 0, & \text{otherwise.} \end{cases}$$

By the use of the previous introduced variable, we formulate the following binary linear program (BLMPG1 - BLMPG4):

Binary **L**inear **M**odel with **P**attern **G**eneration - BLMPG:

$$\min \quad z = \sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} \delta_{kp} \qquad \text{(BLMPG1)}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}} \underbrace{\sum_{k \in \mathcal{K}} k\delta_{kp}}_{\in \mathbb{Z}_0^+} a_{ip} = D_i \qquad \forall i \in \mathcal{I} \qquad \text{(BLMPG2)}$$

$$\sum_{k \in \mathcal{K}} \delta_{kp} \leq 1 \qquad \forall p \in \mathcal{P} \qquad \text{(BLMPG3)}$$

$$\delta_{kp} \in \{0, 1\} \qquad \text{(BLMPG4)}$$

The number of different patterns is to be minimized (BLMPG1). Constraint BLMPG2 enforces exact demand fulfilment. To ensure the use of at most one δ_{kp} (*i. e.*, to enforce the uniqueness multiplicity of the pattern p as claimed in the definition of δ_{kp}), we add constraint BLMPG3. Note that this constraint could be omitted (Lemma 1). Though adding the constraint has a varying, but on average, a slightly positive effect on solution time. Lemma 1 is also valid for the model formulations developed in Subsection 2.1.3.

Lemma 1 (Uniqueness of Pattern Multiplicity) *The uniqueness multiplicity of a pattern p in an optimal solution δ_{kp} for the model BLMPG is ensured without constraint BLMPG3, i. e., the constraint can be removed from the model.*

Proof. Assuming there is a pattern $\widetilde{p}$ in the optimal solution of BLMPG with several different pattern multiplicities, *i. e.*, there is a set $\widetilde{\mathcal{K}} \subset \mathcal{K}$ holding $\delta_{k\widetilde{p}} = 1, \ \forall k \in \widetilde{\mathcal{K}}$ and $\sum_{k \in \widetilde{\mathcal{K}}} \delta_{k\widetilde{p}} = \widetilde{k} > 1$. Choose $k^* \in \mathcal{K}$ in such a way, that $k^* = \sum_{k \in \widetilde{\mathcal{K}}} k\delta_{k\widetilde{p}}$ holds. Assigning $\delta_{k^*\widetilde{p}} = 1$ and $\delta_{k\widetilde{p}} = 0, \ \forall k \in \widetilde{\mathcal{K}}$ reduces the target function value $\sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} \delta_{kp}$ by $|\widetilde{\mathcal{K}}| - 1$ in contradiction to the optimality of the solution, while the equality of the demand fulfillment constraint BLMPG2 is ensured. □

In order to reduce the solution time, the values for many variables δ_{kp} could be fixed, *i. e.*, $\delta_{kp} = 0$ for $k > \min_{i \in \mathcal{I}} \lfloor D_i / \gamma_{ip} \rfloor$, $\forall (k, p) \in \mathcal{K} \times \mathcal{P}$. However, there is no observable influence on the solution time, since this is also done directly by the presolver and somehow slightly faster. As stated before, this approach is applicable for approximately one quarter of all instances. In general, it is not possible to generate all patterns if there is a great spread in the order widths, especially in the context of an unlimited number of knifes. The numerical results of this approach are presented and discussed in detail in Chapter 3.

2.1.3. Monolithic Binary Linear Model Formulation

In this Section, we derive a novel linear model formulation for the one-dimensional cutting stock problem with setup minimization and exact demand fulfilment. Instead of generating all valid patterns in advance (which is not always possible), we will introduce a large number of binary variables to avoid the non linearity in the model formulation described in Subsection 2.1.1. In order to reduce the solution time and the number of variables, we will exploit some problem- and model-specific properties and techniques. To formulate the problem as a binary linear program, we will use the binary variable δ_{kp} already introduced in the previous Subsection 2.1.2, indicating if the pattern p is used exactly k times and the binary variable γ_{iknp}, defined as follows:

$$\gamma_{iknp} := \begin{cases} 1, & \text{if } p \text{ contains } i \text{ exactly } n\text{-times and is used exactly } k \text{ times} \\ 0, & \text{otherwise.} \end{cases}$$

By the use of the previous variables, we formulate the following binary linear program (constraints `BLM1` - `BLM10`):

Binary **L**inear **M**odel - `BLM`:

$$\min \quad z = \sum_{p\in\mathcal{P}} \sum_{k\in\mathcal{K}} \delta_{kp} \tag{BLM1}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}} \sum_{k\in\mathcal{K}} \sum_{n\in\mathcal{N}} kn\gamma_{iknp} = D_i \qquad \forall i \in \mathcal{I} \tag{BLM2}$$

$$\sum_{i\in\mathcal{I}} \sum_{n\in\mathcal{N}} w_i n\gamma_{iknp} \leq W\delta_{kp} \qquad \forall (k,p) \in \mathcal{K}\times\mathcal{P} \tag{BLM3}$$

$$\sum_{i\in\mathcal{I}} \sum_{n\in\mathcal{N}} n\gamma_{iknp} \leq K\delta_{kp} \qquad \forall (k,p) \in \mathcal{K}\times\mathcal{P} \tag{BLM4}$$

$$\sum_{k\in\mathcal{K}} \delta_{k,p+1} \leq \sum_{k\in\mathcal{K}} \delta_{kp} \qquad \forall p \in \mathcal{P}\setminus\{p_{|\mathcal{I}|}\} \tag{BLM5}$$

$$\sum_{k\in\mathcal{K}} k\delta_{k,p+1} \leq \sum_{k\in\mathcal{K}} k\delta_{kp} \qquad \forall p \in \mathcal{P}\setminus\{p_{|\mathcal{I}|}\} \tag{BLM6}$$

$$\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{K}} \gamma_{iknp} \leq 1 \qquad \forall (i,p) \in \mathcal{I}\times\mathcal{P} \tag{BLM7}$$

$$\sum_{k\in\mathcal{K}} \delta_{kp} \leq 1 \qquad \forall p \in \mathcal{P} \tag{BLM8}$$

$$\sum_{k\in\mathcal{K}} k\delta_{kp_s} + \sum_{s'<s} \widetilde{D}_{s'} \geq m^{lo} - \sum_{p\in\mathcal{P},p>s} \sum_{k\in\mathcal{K}} k\delta_{kp} \qquad \forall s \in \{1,\ \ldots,\ |\mathcal{I}|\} \tag{BLM9}$$

$$\delta_{kp} \in \{0,1\},\ \gamma_{iknp} \in \{0,1\} \tag{BLM10}$$

The number of different patterns is to be minimized (`BLM1`). Constraint `BLM2` enforces exact demand fulfilment (beware of the linearity). Constraint `BLM3` ensures the logical connection between δ_{kp} and γ_{iknp}, in such a way, that the pattern p is interpreted as used, if it is generated by cutting the master roll into smaller items while observing the width W. Another connection of the variables via constraint `BLM4` affects the solution time positively (note that there are other possibilities to connect the variables, for example in

BLM7, with differing impact on the solution time). Constraint BLM5 enforces that the next pattern $p+1$ will only be used, if pattern p is already used, while BLM6 sorts the patterns in descending order (symmetry breaking constraints). Both constraints tighten up the solution space to reduce the solution time. The model is completed by BLM7 and BLM8 to ensure the uniqueness of the item multiplicity n of width i resp. the multiplicity k of a pattern p. Note that the constraint BLM8 could be omitted in consequence of Lemma 1. The constraint BLM9 is added to improve the performance and will be motivated later on in Note 3.

Constraints BLM5 and BLM6 need to be explained briefly. If pattern p is not used, the sum on the right site of the inequality is zero, enforcing each term in the sum on the left site to be equal to zero, *i. e.*, $\delta_{k,p+1} = 0,\ \forall k \in \mathcal{K}$. Thus, pattern $p+1$ cannot be used for any multiplicity k.

If p is used, the sum on the right side of the inequality, respectively exactly one δ_{pk} according to constraint BLM8, is equal to one. This allows a single variable on the left side to be equals one for an arbitrary multiplicity k, *i. e.*, the pattern $p+1$ can be used. Other cases do not appear, due to constraint BLM8.

$$\sum_{k\in\mathcal{K}} \delta_{k,p+1} \ \leq \ \sum_{k\in\mathcal{K}} \delta_{kp} \qquad \forall p \in \mathcal{P} \setminus \{p_{|\mathcal{I}|}\} \qquad \text{(BLM5)}$$

The ordering constraint BLM6 works in a quite similar way. If p is used with multiplicity k^*, *i. e.*, $\delta_{pk^*} = 1$ and $\delta_{pk} = 0,\ \forall\ k \in \mathcal{K} \setminus \{k^*\}$, the right side of the inequality BLM6 is equal to k^*. This allows the usage of pattern $p+1$, but only with smaller or equal multiplicity $k^{'}$ (corresponding to the value of the left side). Consequently, $k^{'} \leq k^*$.

$$\underbrace{\sum_{k\in\mathcal{K}} k\delta_{k,p+1}}_{=\,k^{'}} \ \leq \ \underbrace{\sum_{k\in\mathcal{K}} k\delta_{kp}}_{=\,k^*} \qquad \forall p \in \mathcal{P} \setminus \{p_{|\mathcal{I}|}\} \qquad \text{(BLM6)}$$

The number of variables can be reduced significantly before starting the solution process. As a consequence of the ordering-constraint BLM6, the values for many variables can be fixed in advance according to Lemma 2, *i. e.*,

$$\delta_{kp_s} = 0 = \gamma_{iknp_s},\ \forall (s,i,k,n) \ \text{ for } \ k > \widetilde{D}_s, \ \text{where } s \in \{1,\ \ldots,\ |\mathcal{I}|\},$$

leading to a significant decrease in terms of solution time, where $\widetilde{D}_s$ denotes the demands sorted in descending order. We will motivate Lemma 2 based on the data of Example 1. The first pattern will not be used more than $D_1 = 60$ times, since overproduction must be avoided. In addition, the second pattern p_2 will not be used more that $D_2 = 30$ times in the optimal solution. If we assume the multiplicity of p_2 to be greater than D_2, non of the orders i_2, i_3 or i_4 can be part of the first two patterns, since the patterns are sorted in descending order (*i. e.*, the first pattern p_1 will be used at least 30 times as well) and overproduction of i_2, i_3 or i_4 must be avoided. Therefore, the first two patterns are used to cover up the demand of the order width i_1 exclusively, which cannot be optimal.

Lemma 2 (Upper Bounds on Pattern Multiplicity) *In an optimal solution δ_{kp} of model BLM with sorted pattern multiplicity, the multiplicity of pattern p_s is bounded by $\widetilde{D}_s$, i. e., $\sum_{k\in\mathcal{K}} k\delta_{kp_s} \leq \widetilde{D}_s,\ \forall s \in \{1,\ \ldots,\ |\mathcal{I}|\}$, where $\widetilde{D}_s$ denotes the demand levels sorted in descending order.*

Proof. Let $s \in \{1, \ldots, |\mathcal{I}|\}$. If we assume the multiplicity of pattern p_s to be greater than the corresponding $\widetilde{D}_s$, no order width with demand levels smaller or equal than $\widetilde{D}_s$ can be part of the patterns $p_1, \ldots, p_s$, because overproduction must be avoided and the patterns are sorted in descending order. That implies the need of the first s patterns to cover up the demand for at most $s-1$ order widths in contradiction to the optimality of the solution (in the worst case, we need a different pattern for each order width). □

Note that there is a trivial example (the widths ordered are all wider than $W/2$) where the pattern multiplicities are equal to their upper bounds.
Furthermore, the item multiplicity n of a specific width i in a pattern p is restricted by the number of times the order width can be cut from the master roll $\lfloor W/w_i \rfloor$, their demand D_i and by the number of knifes K, *i. e.*,

$$\gamma_{iknp} = 0, \ \forall (i,k,n,p) \ \text{ for } \ n > \min\{\lfloor W/w_i \rfloor ; \ D_i; \ K\}.$$

Of course, $\gamma_{iknp} = 0, \ \forall (i,k,n,p)$ for $nk > D_i$, since the term nk describes the number of produced rolls of type i by a particular pattern p (this is also done directly by the solver during presolve).

One can also calculate some weak lower bounds on the pattern multiplicity in order to fix the values of more variables before starting the solution process, at least by considering the first patterns. For most instances, the lower bounds are negative (and therefore worthless), except for the first three or four patterns on average. We will motivate the following Lemma 3 based on some example data. The width of the master rolls is given by $W = 1000$, the demand and the width of the orders are summarized in the following table.

Basic Data

i	w_i	D_i
i_1	750	40
i_2	400	30
i_3	600	25
i_4	500	20

Initially, we will calculate a lower bound m^{lo} on the number of master rolls being used in total. If we assume that there are patterns with zero cutoff, we can derive

$$m^{lo} = \left\lceil \frac{40 \cdot 750 + 30 \cdot 400 + 25 \cdot 600 + 20 \cdot 500}{1000} \right\rceil = 67,$$

i. e., we have to cut at least 67 master rolls.

Since four is an upper bound on the number of different patterns to be used and the patterns are sorted in descending order according to their multiplicities, we can derive that the first pattern p_1 will be applied at least $\lceil 67/4 \rceil = 17$ times. The patterns p_2, p_3 and p_4 will be applied at least $67 - 40 = 27$ times, since the first pattern will not be used more than 40 times according to Lemma 2. Hence, the second pattern p_2 will be applied at least $\lceil 27/3 \rceil = 9$ times. The lower bounds on $p3$ and p_4 are negative already.

Lemma 3 (Lower Bounds on Pattern Multiplicity) *In an optimal solution δ_{kp} of the model `BLM` with sorted pattern multiplicity, the multiplicity of pattern p_s is greater or equal l_s, defined by*

$$l_s := \left\lceil \frac{m^{lo} - \sum_{s'<s} \widetilde{D}_{s'}}{z^{up} - s + 1} \right\rceil, \ \forall s \in \{1, \ \ldots, \ z^{up}\},$$

where $m^{lo} := \left\lceil \frac{1}{W} \sum_{i \in \mathcal{I}} D_i w_i \right\rceil$ is a (weak) lower bound on the number of master rolls to be used, $\widetilde{D}_s$ denotes the demands sorted in descending order and z^{up} is an upper bound on the number of different patterns (setups) to be used.

Proof. Let $s \in \{1,\ \ldots,\ z^{up}\}$. Then, $m^{lo} - \sum_{s'<s} \widetilde{D}_{s'}$ is a lower bound on the number of master rolls $\sum_{p\in\mathcal{P},p\geq s} \sum_{k\in\mathcal{K}} k\delta_{kp}$ to be used for the patterns $p_s, p_{s+1}, ..., p_{|\mathcal{I}|}$, since

$$\sum_{p\in\mathcal{P},p<s} \sum_{k\in\mathcal{K}} k\delta_{kp} \leq \sum_{s'<s} \widetilde{D}_{s'}$$

due to Lemma 2 (the number of master rolls to be used for the first $p_1, ..., p_{s-1}$ patterns is smaller or equal then the sum of the highest demands $\widetilde{D}_1, ..., \widetilde{D}_{s-1}$), *i. e.*,

$$\sum_{p\in\mathcal{P},p\geq s} \sum_{k\in\mathcal{K}} k\delta_{kp} \geq m^{lo} - \sum_{s'<s} \widetilde{D}_{s'}.$$

Since the patterns are sorted in descending order, one can derive

$$(z^{up} - s + 1) \sum_{k\in\mathcal{K}} k\delta_{kp_s} \geq \sum_{p\in\mathcal{P},p\geq s} \sum_{k\in\mathcal{K}} k\delta_{kp} \ \Rightarrow\ \sum_{k\in\mathcal{K}} k\delta_{kp_s} \geq \left\lceil \frac{m^{lo} - \sum_{s'<s} \widetilde{D}_{s'}}{z^{up} - s + 1} \right\rceil = l_s.$$

□

In consequence, the following values of the variables can be fixed (for $s \in \{1,\ \ldots,\ z^{up}\}$):

$$\delta_{kp_s} = 0,\ \forall k \in \mathcal{K} \ \text{ with } \ k < l_s,\ y_{iknp_s} = 0,\ \forall (i,k,n) \in \mathcal{I}\times\mathcal{K}\times\mathcal{N} \ \text{ with } \ k < l_s.$$

Furthermore, no order width with lower demand than l_s can be part of the pattern, *i. e.*,

$$y_{iknp_s} = 0,\ \forall (i,k,n) \in \mathcal{I}\times\mathcal{K}\times\mathcal{N} \ \text{ with } \ D_i < l_s.$$

Note 3 (Quality of the Lower Bounds on Pattern Multiplicity) *As stated before, the lower bounds l_s are very weak for $z^{up} = |\mathcal{I}|$, i. e., the lower bounds for the third or fourth patterns will be already negative and low for the first patterns for most instances, i. e., most of those lower bounds are worthless.*

When fixing the values of the variables as described above, the solution time decreases slightly on average, hence, the lower bounds on the pattern multiplicity are added to the model.

Instead of the worst case estimation in the last step of the proof of Lemma 3, one can also add the following constraint to the model:

$$\sum_{k\in\mathcal{K}} k\delta_{kp_s} \geq m^{lo} - \sum_{s'<s} \widetilde{D}_{s'} - \sum_{p\in\mathcal{P},p>s} \sum_{k\in\mathcal{K}} k\delta_{kp} \qquad \forall s \in \{1,\ \ldots,\ |\mathcal{I}|\} \quad (\texttt{BLM9}).$$

The effect on the solution times is clearly positive on average, hence the constraint `BLM9` *is added to the model* `BLM` *(see also Section 3.3 for the numerical results).*

Note 4 (Reducing the Number of Patterns in Model BLM) *It is not reasonable to reduce the number of patterns in model* `BLM` *directly, i. e., setting $|\mathcal{P}| = z^{up} < |\mathcal{I}|$, since the solver needs to much time to find a feasible solution, while the number of variables and equations decreases of course due to the value fixing motivated by Lemma 3. As a result, the effect on the solution time is negative on average.*

By investigating the optimal solutions for some instance, we have noticed that the number of different widths per pattern, in the following denoted by $I_p^{max},\ \forall p \in \mathcal{P}$ and the number of patterns per width, in the following denoted by $P_i^{max},\ \forall i \in \mathcal{I}$, are in general much lower as theoretically possible.

Number of Widths per Pattern and Number of Patterns per Width
Therefore, we studied the effect when limiting the number of different widths per pattern and when limiting the number of patterns per width. Since we cannot see a general way to determine the values of $I_p^{max},\ \forall p \in \mathcal{P}$ and $P_i^{max},\ \forall i \in \mathcal{I}$ in advance, we determine those values by evaluating the optimal solutions generated by the model `BLM`. Afterwards, we solved the model `BLM` again, while adding the following constraint to limit the number of different widths per pattern:

$$\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} \gamma_{iknp} \leq I_p^{max} \qquad \forall p \in \mathcal{P},$$

resp. for a limit on the number of patterns per width:

$$\sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{n\in\mathcal{N}} \gamma_{iknp} \leq P_i^{max} \qquad \forall i \in \mathcal{I}.$$

We studied the effect on the solution time, which is clearly negative on average, since the finding of feasible solutions will be more difficulty for the solver. Additionally, it remains unclear how to solve the major problem of determining the values of $I_p^{max},\ \forall p \in \mathcal{P}$ and $P_i^{max},\ \forall i \in \mathcal{I}$ in advance.

2.1.4. Multiple Types of Master Rolls

In order to describe the pure setup minimization problem with exact demand fulfilment and multiple types of master rolls, we define the following sets and parameters in addition to those introduced in the beginning of Chapter 2:

- Set of different types of master rolls $j \in \mathcal{J} = \{1, \ldots, |\mathcal{J}|\}$.
- Width W_j of master roll type j, sorted in ascending order, *i. e.*, $W_1 < ... < W_{|\mathcal{J}|}$.
- Capacity (stock level) C_j of master roll type j.

The setup minimization problem with multiple types of master rolls can be seen as generalization of the setup minimization problem with a single type of master roll. However, take in mind the following note.

Note 5 (Changes in the Problem Definition)
While the capacity of master rolls was assumed to be unrestricted in case of a single type of master roll, this assumption is no longer reasonable when minimizing the number of setups for multiple types of master rolls, as one could focus on the widest master rolls to obtain the optimal solution for the pure setup minimization problem. By using smaller master rolls, the number of setups might eventually increase due to a decrease in possible patterns applicable on the smaller master rolls.

In order to define a reasonable problem statement, we have to add at least one of the further requirements to the setup minimization problem:

1. *Observing a (potentially active) capacity limit C_j on the stock of master rolls for all $j \in \mathcal{J}$.*
2. *Focussing on setup minimization as primary objective, but preferring the usage of smaller master rolls as secondary objective (by observing the cutoff).*

In the following, we will consider only the first remark, *i. e.*, we observe capacity limits (stock levels) on the different types of master rolls while presenting some model formulations closely related to `BLMPG` resp. `BLM`.

Exact Binary Linear Model based on complete Pattern Generation
At first, we formulate the model based on complete pattern generation. For this we define the following variables:

$$\eta_p := \begin{cases} 1, & \text{if } p \text{ is applied} \\ 0, & \text{otherwise} \end{cases}$$

and

$$\mu_{pjk} := \begin{cases} 1, & \text{if } p \text{ is applied on master roll type } j \text{ with multiplicity } k \\ 0, & \text{otherwise.} \end{cases}$$

The `CPLEX Solution Pool` options is used to generate all valid patterns by solving the model `PG`, see also Section 2.1.2. Afterwards, we evaluate the generated patterns, *i. e.*, we determine if the pattern p can be applied on master roll type j. The results will be stored as parameter $e_{pj} \in \{0, 1\}$ (e_{pj} is equal to one, if p can be applied on j).

By the use of the previous variables and parameters, we formulate the following binary linear program based on complete pattern generation (constraints BLMPG-MMR1 - BLMPG-MMR6):

Bin. **L**in. **M**odel with **P**attern **G**eneration and **M**ult. Types of **M**aster **R**olls - BLMPG-MMR:

$$\min \quad z = \sum_{p \in \mathcal{P}} \eta_p \qquad \text{(BLMPG-MMR1)}$$

$$\begin{aligned}
\text{s.t.} \quad \sum_{p \in \mathcal{P}} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} k \mu_{pjk} \gamma_{ip} &= D_i \qquad \forall i \in \mathcal{I} && \text{(BLMPG-MMR2)} \\
\sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} k \mu_{pjk} &\leq C_j \qquad \forall j \in \mathcal{J} && \text{(BLMPG-MMR3)} \\
\sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \mu_{pjk} &\leq |\mathcal{J}| \eta_p \quad \forall p \in \mathcal{P} && \text{(BLMPG-MMR4)} \\
\sum_{k \in \mathcal{K}} \mu_{pjk} &\leq e_{pj} \qquad \forall (p,j) \in \mathcal{P} \times \mathcal{J} && \text{(BLMPG-MMR5)} \\
\eta_p \in \{0,1\},\ \mu_{pjk} &\in \{0,1\} && \text{(BLMPG-MMR6)}
\end{aligned}$$

The objective is to minimize the number of different patterns (BLMPG-MMR1). Constraint BLMPG-MMR2 enforces exact demand fulfilment, while the stock levels of the different master rolls are observed by constraint BLMPG-MMR3. A pattern p is considered as used, if it is applied on any master roll type j (BLMPG-MMR4). Constraint BLMPG-MMR5 is used to connect the variable μ_{pjk} with the parameter e_{pj}, while also ensuring the uniqueness of the pattern multiplicity k of p on master roll type j.

The values of many variables μ_{pjk} can be fixed, *e. g.*, $\mu_{pjk} = 0$, if $k > \min_{i \in \mathcal{I}} [\lfloor D_i / \gamma_{ip} \rfloor]$ $\forall (p,j,k) \in \mathcal{P} \times \mathcal{J} \times \mathcal{K}$. In addition, μ_{pjk} can be set to zero, if $k > C_j$ $\forall (p,j,k) \in \mathcal{P} \times \mathcal{J} \times \mathcal{K}$ (constraint BLMPG-MMR3).

Exact Binary Linear Model Formulation

If it's not possible to generate all patterns (note that this gets more likely as there are probably some wider master rolls compared to the single type of master roll problem), we suggest a linear model formulation based on the following binary variables:

$$\gamma_{iknp} := \begin{cases} 1, & \text{if } p \text{ contains } i \text{ exactly } n\text{-times and is used exactly } k \text{ times} \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_{pjk} := \begin{cases} 1, & \text{if } p \text{ is applied on roll type } j \text{ exactly } k \text{ times} \\ 0, & \text{otherwise} \end{cases}$$

$$\upsilon_{pj} := \begin{cases} 1, & \text{if } p \text{ can be applied on roll type } j \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{pk} := \begin{cases} 1, & \text{if } p \text{ is used exactly } k \text{ times} \\ 0, & \text{otherwise} \end{cases}$$

By the use of the previous variables, we formulate the following binary linear program (constraints BLM-MMR1 - BLM-MMR14):

Binary **L**inear **M**odel with **M**ultiple Types of **M**aster **R**olls - BLM-MMR:

$$\min \quad z = \sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} \delta_{pk} \tag{BLM-MMR1}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{n\in\mathcal{N}} kn\gamma_{iknp} = D_i \qquad \forall i \in \mathcal{I} \tag{BLM-MMR2}$$

$$\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} w_i n\gamma_{iknp} \leq W_{|\mathcal{J}|}\delta_{pk} \qquad \forall (p,k) \in \mathcal{P}\times\mathcal{K} \tag{BLM-MMR3}$$

$$\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} w_i n\gamma_{iknp} \leq W_{|\mathcal{J}|}\upsilon_{p|\mathcal{J}|} - \sum_{j\in\mathcal{J}\setminus\{|\mathcal{J}|\}} (W_{j+1}-W_j)\upsilon_{pj} \qquad \forall (p,k) \in \mathcal{P}\times\mathcal{K} \tag{BLM-MMR4}$$

$$\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} n\gamma_{iknp} \leq K\delta_{kp} \qquad \forall (p,k) \in \mathcal{P}\times\mathcal{K} \tag{BLM-MMR5}$$

$$\upsilon_{pj} \leq \upsilon_{p,j+1} \qquad \forall (p,j) \in \mathcal{P}\times\mathcal{J}\setminus\{|\mathcal{J}|\} \tag{BLM-MMR6}$$

$$\sum_{k\in\mathcal{K}} k\delta_{kp} = \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} k\mu_{pjk} \qquad \forall p \in \mathcal{P} \tag{BLM-MMR7}$$

$$\sum_{k\in\mathcal{K}} \mu_{pjk} \leq \upsilon_{pj} \qquad \forall (p,j) \in \mathcal{P}\times\mathcal{J} \tag{BLM-MMR8}$$

$$\sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} k\mu_{pjk} \leq C_j \qquad \forall j \in \mathcal{J} \tag{BLM-MMR9}$$

$$\sum_{k\in\mathcal{K}}\sum_{n\in\mathcal{N}} \gamma_{iknp} \leq 1 \qquad \forall (p,i) \in \mathcal{P}\times\mathcal{I} \tag{BLM-MMR10}$$

$$\sum_{k\in\mathcal{K}} \delta_{pk} \leq 1 \qquad \forall p \in \mathcal{P} \tag{BLM-MMR11}$$

$$\sum_{k\in\mathcal{K}} k\delta_{pk} \geq \sum_{k\in\mathcal{K}} k\delta_{p+1,k} \qquad \forall p \in \mathcal{P} \tag{BLM-MMR12}$$

$$\sum_{k\in\mathcal{K}} \delta_{pk} \geq \sum_{k\in\mathcal{K}} \delta_{p+1,k} \qquad \forall p \in \mathcal{P} \tag{BLM-MMR13}$$

$$\delta_{kp} \in \{0,1\},\ \gamma_{iknp} \in \{0,1\},\ \mu_{pjk} \in \{0,1\}, \upsilon_{pj} \in \{0,1\} \tag{BLM-MMR14}$$

The number of different patterns is to be minimized (BLM-MMR1), while the second constraint ensures the exact demand fulfilment (BLM-MMR2). Although we allow the pattern generation based on the widest master roll type by constraint BLM-MMR3, we evaluate for each pattern on which master roll types it can be applied to (constraints BLM-MMR4 and BLM-MMR6). The number of knifes is observed by the constraint BLM-MMR5. In addition, the third and sixth constraint are used to connect the variables γ_{iknp} and δ_{pk}. The total multiplicity of the pattern p must be equal to the sum of the pattern multiplicity's being applied on the different types of master rolls (BLM-MMR7). If a pattern p cannot be applied on roll type j, μ_{pjk} must be equal to zero for all $k \in \mathcal{K}$ (constraint BLM-MMR8). The stock levels must be observed for all types of master rolls (BLM-MMR9). The uniqueness of the patterns multiplicity k and the uniqueness of the item multiplicity n of an order width i are ensured by the constraints BLM-MMR10 and BLM-MMR11, while BLM-MMR11 could be omitted of course (see Lemma 1). Finally, the patterns are sorted in descending order

(BLM-MMR12), while the next pattern can only be applied, if the previous pattern is already used (BLM-MMR13). Both constraints are added to reduce the search space and to improve performance.

First numerical results indicate that instances of small and medium complexity of the Kallrath benchmark set can be solved to optimality in reasonable time by using the formulations presented above. Note here that we just added two types of master rolls with width $0.7 \cdot W$ resp. $1.3 \cdot W$ to the already existing type of width W (*i. e.*, $|\mathcal{J}| = 3$) and some random stock levels C_j for testing purposes only.

2.2. Heuristic Approaches

In this Section, we will present some heuristic approaches for solving the one-dimensional cutting stock problem with setup minimization and exact demand fulfilment.

The idea for the first approach presented in Subsection 2.2.1 is to adapt the complete pattern generation approach demonstrated in Subsection 2.1.2 (model BLMPG). Instead of generating all valid patterns, which is not always possible, we confine the pattern generation on efficient patterns, *e. g.*, patterns with a small amount of trim loss (cutoff).

We will present some variations of a Greedy algorithm in Subsection 2.2.2. The basic idea of this algorithm is to generate patterns stepwise, while we try to maximise the number of order widths which are fulfilled exactly by generating a single pattern in the current iteration. Already fulfilled order widths are removed from the set of all order widths after each iteration.

In Subsection 2.2.3, we present a novel approach for splitting an original problem instance into smaller derived sub-instances. Those much smaller and less complex sub-instances are formulated according to the model formulation BLM introduced in Subsection 2.1.3, which is quite performant for instances with small or medium complexity, and solved with CPLEX afterwards. The unification of the solutions of the sub-instances provides a feasible solution to the original problem instance.

Heuristic approaches are in general the only way to calculate acceptable solutions in terms of solution time and solution quality also for large and complex instances of the cutting stock problem with setup minimization. For instance, the Belov benchmark sets provides problem sizes with up to 150 order widths, for whom it is not even possible to generate the model BLM. However, the drawback is, that we have no guarantee to find the optimal solution (or even good solutions, *i. e.*, solutions with a small gap). In addition, we have troubles to evaluate the goodness of the heuristic solution in general.

2.2.1. Combining Generation of efficient Patterns with free Patterns

Initially, we generate a set $\mathcal{P}^g$ of efficient patterns with efficiency rate $0 < e \leq 1$, *e. g.*, patterns with fife percent trim loss ($e = 0.05$), by solving the pattern generation model PG (constraints PG1-PG4) introduced in Subsection 2.1.2 with the CPLEX Solution Pool functionality while adding the following constraint:

$$\sum_{i \in \mathcal{I}} w_i \gamma_i \geq W(1 - e). \tag{PG5}$$

Afterwards, we solve the model BLMPG based on the set of efficient patterns. If we obtain an optimal solution $z^{P^g_{opt}}$, it provides an upper bound on the optimal solution z^{opt}.

In case of infeasibility, we add free patterns one after another to the set $\mathcal{P}^f$ of the free patterns in order to obtain an (optimal) solution $z^{P^g_{opt}}$. These free patterns in $\mathcal{P}^f$ are then added to $\mathcal{P}^g$ and removed from $\mathcal{P}^f$. Note that the free patterns corresponds to the binary variable γ_{iknp} with $p \in \mathcal{P}^f$ introduced in Section 2.1.3, while the generated patterns are represented by the parameter γ^g_{ip}, denoting the item multiplicity of width $i \in \mathcal{I}$ in pattern $p \in \mathcal{P}^g$.

Note 6 (Worst Case Number of free Patterns in Case of Infeasibility) *In the worst case, one must add $|\mathcal{I}|$ free patterns to $\mathcal{P}^f$ in order to obtain an (optimal) solution $z^{P^g_{opt}}$ when solving* `BLMPG` *at first. Obviously, it holds $z^{P^g_{opt}} = z^{opt}$ in that case. For a given efficiency rate $0 < e \leq 1$, i. e., patterns with a cutoff greater than $e \cdot W$ will not be generated since they are not efficient, one can easily construct such a pathological worst case scenario by defining the order widths in the following way: $W/2 < w_i < (1-e)W,\ \forall i \in \mathcal{I}$. For practical reasons, the efficiency rate e can be assumed to be (much) smaller than 0,5.*

Another worst case scenario considers the number of knifes, while the order widths can be arbitrary small: $K = 1$, $w_i < (1-e)W,\ \forall i \in \mathcal{I}$ and $w_i + w_{i'} \neq W,\ \forall i, i' \in \mathcal{I}$ with $i \neq i'$ (the last constraint prevents the unlikely case that there can be cut two order widths with one knife in a single pattern).

After calculating a solution $z^{P^g_{opt}}$, we warm start a slightly modified version of `BLM` (described in the following), named `BLMEPG`, with constraints `BLMEPG1`, `BLMEPG2`, `BLM3`-`BLM7` and `BLMEPG8`, in order to calculate z^{opt} by adding $z^{P^g_{opt}} - 1$ free patterns to $\mathcal{P}^f$, *i. e.*, $\mathcal{P}^f = \left\{p_1, ..., p_{z^{P^g_{opt}}-1}\right\}$.

Lower values for $\mathcal{P}^f$ do not guarantee the solution to be optimal (Lemma 4), but are useful to reduce the solution time (heuristic approach).

We have to observe both kind of patterns in the objective of course, *i. e.*,

$$\min\ z = \sum_{p \in \mathcal{P}^g \cup \mathcal{P}^f} \sum_{k \in \mathcal{K}} \delta_{kp}. \qquad \texttt{(BLMEPG1)}$$

The demand fulfilment equation is defined by

$$\sum_{p \in \mathcal{P}^g} \sum_{k \in \mathcal{K}} k \delta_{kp} \gamma^g_{ip} + \sum_{p \in \mathcal{P}^f} \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} kn \gamma_{iknp} = D_i \qquad \forall i \in \mathcal{I}. \qquad \texttt{(BLMEPG2)}$$

Note that the term $k\delta_{kp}\gamma^g_{ip}$ is linear, since γ^g_{ip} is a parameter. The constraints `BLM3` to `BLM7` from the model `BLM` must be applied only for the free patterns, *i. e.*, $\forall p \in \mathcal{P}^f$. The inequality `BLM8` is applied for both, the generated patterns and the free patterns, since the uniqueness multiplicity of all pattern shall be ensured, *i. e.*,

$$\sum_{k \in \mathcal{K}} \delta_{kp} \leq 1 \qquad \forall p \in \mathcal{P}^g \cup \mathcal{P}^f. \qquad \texttt{(BLMEPG8)}$$

Note that we omit constraint `BLM9` for simplicity.

Lemma 4 (Smallest valid Number of free Patterns for Model BLMEPG) *Given an optimal solution $z^{P^g_{opt}}$ of the model* `BLMPG` *based on pattern generation with efficiency rate e, then $z^{P^g_{opt}} - 1$ is a sufficient large number of free patterns to guarantee finding an optimal solution z^{opt} of the model* `BLMEPG`*, while there are also configurations, where $z^{P^g_{opt}} - 1$ is the smallest number of free patterns necessary to calculate an optimal solution.*

Proof. Since $z^{P^g_{opt}}$ is an upper bound to z^{opt}, adding $z^{P^g_{opt}} - 1$ free patterns is obviously sufficient to find an optimal solution of the model BLMEPG. However, one can construct worst case scenarios, where all free patterns $\mathcal{P}^f = \left\{p_1, ..., p_{z^{P^g_{opt}}-1}\right\}$ are active to reduce the number of different patterns (by one), while in the optimal solution, none of the generated efficient patterns can be used. For instance:

Width of master roll $W = 100$, set of orders $\mathcal{I} = \{i_1, ..., i_4\}$, efficiency rate $e = 0,12$. The table summarizes the order widths w_i, the demand D_i, all efficient generated patterns and the optimal solution based on the those patterns.

	p_1	p_2	p_3	p_4	
$w_i \backslash \mu_p$	4	1	6	4	D_i
$w_1 = 30$	3	0	0	0	12
$w_2 = 44$	0	2	1	1	12
$w_3 = 51$	0	0	1	0	6
$w_4 = 52$	0	0	0	1	4
$\sum w_i \gamma^g_{ip}$	90	88	95	96	

The minimal number of setups based on the generated efficient patterns is $z^{P^g_{opt}} = 4$, while the optimal solution z^{opt} for this configuration based on free and generated patterns (shown in the next table) actually needs $z^{opt} = z^{P^g_{opt}} - 1 = 3$ free patterns. Note that there is no optimal solution using any of the generated (efficient) patterns, *i. e.*, we would not have found the optimal solution, if we allowed less than $z^{P^g_{opt}} - 1$ free patterns.

	p_1	p_2	p_3	
$w_i \backslash \mu_p$	12	6	4	D_i
$w_1 = 30$	1	0	0	12
$w_2 = 44$	1	0	0	12
$w_3 = 51$	0	1	0	6
$w_4 = 52$	0	0	1	4
$\sum w_i \gamma^f_{ip}$	74	51	52	

□

Note that the smallest number of free patterns necessary to find an optimal solution is probably smaller, if there had been already added free patterns to obtain a feasible solution when solving BLMPG based on the efficient patterns initially.

The solution times of the modified model BLMEPG have increased on most instances. Holding a large amount of generated patterns in combination with free patterns often results in an out of memory abort. Therefore, we suggest two variations.

Instead of solving the model BLMEPG, we use the generated patterns only to get a good bound z^{up} to z^{opt} by solving BLMPG. Afterwards, we warm start the original model BLM resp. the model BLMEPG (but by defining $|\mathcal{P}^g| = 0$ and $|\mathcal{P}^f| = z^{up}$ in order to omit the generated patterns). Note that the solution must be sorted in advance to fulfil the symmetry breaking constraints BLM5 and BLM6.

The second variation is very obvious and a heuristic approach (while everything discussed above was exact of course). Instead of solving an exact model formulation like BLM

or BLMEPG subsequently, we denote the solution of the model BLMPG based on the efficient generated patterns as heuristic solution and terminate the process at this point (while there are some major differences, the idea of generating only a subset of all patterns in order to solve a MILP model afterwards is well known, see [19] for instance). However, if the model is infeasible at first, we have to add some free patterns.

The usage of generated patterns with small cutoff often leads to solutions with a small number of master rolls m to be cut (secondary objective). Though, all approaches in this Subsection will not be discussed further nor are the results presented and compared to the other approaches, since we could not investigate the behaviour of the approaches on a broad range of benchmark instances.

2.2.2. Greedy Algorithms

In this Subsection, we propose (some variations of) a novel heuristic greedy algorithm in order to solve the one-dimensional cutting stock problem with pattern reduction. A greedy algorithm searches the local optimum in each step, while there is in general no guarantee that the returned solution is globally optimal.

The basic idea of our approach is to generate single patterns successive, while we try to maximize the number of order widths being satisfied exactly by a single pattern in each step (getting an local optimum). Order widths which are fulfilled exactly are removed after each iteration until the demand of all order widths is satisfied exactly. The number of iterations is equivalent to the number of the generated patterns and will be returned as heuristic solution z.

Algorithm Greedy1
First of all, we have to introduce some additional notations:

- A binary variable $\chi_i \ \forall i \in \mathcal{I}$, defined as

$$\chi_i := \begin{cases} 1, & \text{if the order } i \text{ is satisfied exactly} \\ 0, & \text{otherwise.} \end{cases}$$

- Set $\mathcal{I}' \subseteq \mathcal{I}$ of order widths which are not fulfilled exactly yet. Initially, it holds $\mathcal{I}' = \mathcal{I}$ (*i. e.*, no order width is fulfilled exactly).

- Number of order widths I^P which can be fulfilled exactly by the current pattern.

- Remaining length W^r of the recently generated pattern which can be used to cut further items of the not yet fulfilled order widths.

- Remaining number K^r of knifes of the recently generated pattern which can be used to cut further items of the not yet fulfilled order widths.

The variable z, defined as the number of different patterns in the previous Section, will be used as an iteration counter, *i. e.*, we increase z by one for each generated pattern resp. iteration, starting with $z = 0$.

By the use of the previous variables and parameters, we formulate the following binary linear program (G1A1 - G1A7):

Model for the **G**reedy**1** **A**lgorithm - `G1A`

$$\max \quad I^P = \sum_{i \in \mathcal{I}'} \chi_i \qquad \text{(G1A1)}$$

$$\begin{aligned}
\text{s.t.} \quad \sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} w_i n \gamma_{iknp_z} &\leq W \delta_{kp_z} && \forall k \in \mathcal{K} && \text{(G1A2)} \\
\sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} n \gamma_{iknp_z} &\leq K \delta_{kp_z} && \forall k \in \mathcal{K} && \text{(G1A2)} \\
\sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} k n \gamma_{iknp_z} &= D_i \chi_i && \forall i \in \mathcal{I}' && \text{(G1A4)} \\
\sum_{k \in \mathcal{K}} \delta_{kp_z} &\leq 1 && && \text{(G1A5)} \\
\sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} \gamma_{iknp_z} &\leq 1 && \forall i \in \mathcal{I}' && \text{(G1A6)}
\end{aligned}$$

$$\gamma_{iknp} \in \{0,1\},\ \delta_{kp} \in \{0,1\},\ \chi_i \in \{0,1\} \qquad \text{(G1A7)}$$

Note that we use the variables δ_{kp} and γ_{iknp} of the `BLM` formulation, while the last index p is fixed to the current number of patterns z, which increases in each iteration of the algorithm.

The objective `G1A1` is to maximize the number of order widths whose demand can be fulfilled exactly by the current pattern. The width of the pattern and the number of knifes is observed by the constraints `G1A2` and `G1A3`, while $\chi_i = 1$ enforces the demand of i to be fulfilled exactly by constraint `G1A4`. The uniqueness multiplicity of the pattern with respect to the variable δ_{kp} is ensured by the constraint `G1A5`. The uniqueness multiplicity of the current pattern and the uniqueness item multiplicity n of a width i with respect to the variable γ_{iknp} is ensured by the constraint `G1A6`.

Based on the model `G1A`, we propose the following greedy algorithm, referred as `Greedy1`. In each iteration, the algorithm generates a pattern which maximizes the number of orders satisfied by this pattern (*i. e.*, the demand is fulfilled exactly) by solving `G1A`. After each iteration, there might be enough width W^r of the master roll left to cut some further items in order to reduce the demand of some yet not fulfilled order widths. In detail, for every order width i in the set of the yet not fulfilled orders $\mathcal{I}'$ (starting with the first one which will have the highest demand probably), we try to add i with the highest possible item multiplicity to the current pattern while observing the remaining length W^r of the master roll and the remaining number K^r of knifes. Note here that no order i in $\mathcal{I}'$ will be fulfilled exactly in this post processing step, since otherwise we would have gotten this solution directly from solving the model `G1A`. The pseudocode of the `Greedy1` algorithm is presented below. The parameters u_p and a_{ip} are used to store the solution.

Greedy1 $(\mathcal{I}, D_i, w_i, K, W)$

1: $\mathcal{I}' = \mathcal{I}$
2: $z = 0,\ u_p = 0,\ a_{ip} = 0$
3:
4: **while** $|\mathcal{I}'| > 0$ **do**
5: $\quad z \leftarrow z + 1$
6: $\quad$ solve `G1A`$(\mathcal{I}', z, D_i, w_i, K, W)$
7: $\quad u_{p_z} = \sum_{k \in \mathcal{K}} k\delta_{kp_z}$
8: $\quad a_{ip_z} = \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} n\gamma_{iknp_z}$
9: $\quad W^r = W - \sum_{i \in \mathcal{I}'} w_i a_{ip_z}$
10: $\quad K^r = K - \sum_{i \in \mathcal{I}'} a_{ip_z}$
11: $\quad \mathcal{I}' \leftarrow \mathcal{I}' \setminus \{i \in \mathcal{I}' | \chi_i = 1\}$
12: $\quad$ **for each** $i \in \mathcal{I}'$ **do**
13: $\quad\quad$ reduce D_i
14: $\quad$ **end foreach**
15: $\quad \mathcal{N}, \mathcal{K} \leftarrow$ getMaxMultiplicities$(\mathcal{I}')$
16: **end while**
17:
18: **return:** $z,\ u_p,\ a_{ip}$

Line 15 is motivated due to the obtainable reduction of variables when solving `G1A` in the next iteration by reducing the maximum item multiplicity resp. the maximum pattern multiplicity. The numerical results of `Greedy1` are presented and compared to the other approaches in Section 3.3.

Example 2 (Demonstrating Greedy1) *The `Greedy1` algorithm will be demonstrated on the instance `fiber13a_9080` of the `Fiber` benchmark set. The number of knifes is unrestricted and the master roll width is given by 9080. The following table summarizes the order widths and the demands of the instance* $I = (\mathcal{I}, D_i, w_i, K, W)$.

1. Iteration

i	w_i	D_i
i_1	1000	158
i_2	985	32
i_3	1250	30
i_4	920	23
i_5	940	16
i_6	923	10
i_7	1100	6
i_8	1050	4

When solving the model `G1A` for the first time, the demand of the order widths i_5 *and* i_8 *is fulfilled exactly, i. e.,* $\chi_{i_5} = 1$ *and* $\chi_{i_8} = 1$. *The first pattern* p_1 *will be used four times, the item multiplicity of order* i_5 *is equal to four and the item multiplicity of order* i_8 *is equal to one. Afterwards, the orders* i_5 *and* i_8 *are removed from* $\mathcal{I}'$, *i. e.,* $\mathcal{I}' = \{i_1, i_2, i_3, i_4, i_6, i_7\}$. *At this point, further widths can be added to* p_1, *since there are 4270 length units unused. Starting with the first width in* $\mathcal{I}'$, *we try to add further widths to* p_1. *In consequence, the width* i_1 *is added four times to the pattern* p_1, *which will reduce the demand of* i_1 *to* $D_{i_1} = 142$.

2. Iteration

i	w_i	D_i
i_1	1000	142
i_2	985	32
i_3	1250	30
i_4	920	23
i_6	923	10
i_7	1100	6

Solving the model `G1A` *for the updated set $\mathcal{I}'$ and the reduced demand D_i, the demand for the widths i_3 and i_7 is fulfilled exactly by adding i_3 five times and i_7 a single time to the next pattern p_2, which will be used six times in total. The orders i_3 and i_7 are removed from $\mathcal{I}'$. There are 1730 length units unused in pattern p_2. In consequence, the first order i_1 is added a single time to p_2, which will reduce the demand of i_1 to $D_{i_1} = 136$.*

3. Iteration

i	w_i	D_i
i_1	1000	136
i_2	985	32
i_4	920	23
i_6	923	10

In this iteration, only the demand of a single width can be fulfilled exactly. The solver decides for the order width i_1, which will be added to the pattern p_3 eight times, while the pattern will be used seventeen times in total. Since there are 1080 length units unused in the current pattern, the order width i_2 is added a single time to p_3, reducing the demand of i_2 to $D_{i_2} = 15$.

4. Iteration

i	w_i	D_i
i_2	985	15
i_4	920	23
i_6	923	10

Solving the model `G1A` *for the data on the left will fulfil the demand for i_2 and i_6 exactly, by adding i_2 three times and i_6 twice to the pattern p_4, which will be used 5 times in total. Since there are 4279 length units unused in the current pattern, the order width i_4 is added four times to p_4, reducing the demand of i_4 to $D_{i_4} = 3$.*

5. Iteration

i	w_i	D_i
i_4	920	3

The width i_4 will be added a single time to the last pattern p_5, which will be used three times in total (even if it would be better the other way around, we have no influence in the decision of the solver).

Note that the solution $z = 5$ generated by `Greedy1` *(summarized in the following table) is one pattern worse than the optimal solution.*

	p_1	p_2	p_3	p_4	p_5	
$w_i \backslash u_p$	4	6	17	5	3	D_i
$w_1 = 1000$	4	1	8	0	0	158
$w_2 = 985$	0	0	1	3	0	32
$w_3 = 1250$	0	5	0	0	0	30
$w_4 = 920$	0	0	0	4	1	23
$w_5 = 940$	4	0	0	0	0	16
$w_6 = 923$	0	0	0	2	0	10
$w_7 = 1100$	0	1	0	0	0	6
$w_8 = 1050$	1	0	0	0	0	4
$\sum w_i a_{ip}$	8810	8350	8985	8481	920	

Algorithm Greedy2
This extension of the `Greedy1` algorithm includes the decision how to fill up the already generated patterns with not yet fulfilled order widths to the model formulation (presented below), in order to replace the static filling procedure proposed in `Greedy1`. In addition, we try not only to fill up the last generated pattern, but also all yet generated patterns by including them into the model.

First of all, we have to introduce some additional notations:

- A binary variable $\overline{\gamma}_{inp}$ for all $(i, n, p) \in \mathcal{I} \times \mathcal{N} \times \mathcal{P}$, defined as

$$\overline{\gamma}_{inp} := \begin{cases} 1, & \text{if the order } i \text{ is added to the pattern } p \text{ exactly } n \text{ times} \\ 0, & \text{otherwise.} \end{cases}$$

  Note that the set of patterns $\mathcal{P}$ depends on the number of generated patterns z, *i. e.*, $\mathcal{P} = \{p_1, ..., p_z\}$, where z is incremented by one in each iteration starting with $z = 0$.

- Remaining length W_p^r of a pattern $p \in \mathcal{P} \setminus \{p_z\}$ which can be used to cut further items of the not yet fulfilled order widths.

- Remaining number K_p^r of knifes of a pattern $p \in \mathcal{P} \setminus \{p_z\}$ which can be used to cut further items of the not yet fulfilled order widths.

After solving the model `G1A` initially, we will solve the model `G2A` (`G2A1 - G2A10`), defined below, in each iteration.

Model for the **G**reedy**2** **A**lgorithm - `G2A`

$$\max \quad I^P = \sum_{i \in \mathcal{I}'} \chi_i \tag{G2A1}$$

$$\text{s. t.} \quad \sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} w_i n \gamma_{iknp_z} \leq W \delta_{kp_z} \quad \forall k \in \mathcal{K} \tag{G2A2}$$

$$\sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} n \gamma_{iknp_z} \leq K \delta_{kp_z} \quad \forall k \in \mathcal{K} \tag{G2A3}$$

$$\sum_{p \in \mathcal{P} \setminus \{p_z\}} u_p \sum_{n \in \mathcal{N}} n \overline{\gamma}_{inp} + \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} k n \gamma_{iknp_z} = D_i \chi_i \quad \forall i \in \mathcal{I}' \tag{G2A4}$$

$$\sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} w_i n \overline{\gamma}_{inp} \leq W_p^r \quad \forall p \in \mathcal{P} \setminus \{p_z\} \tag{G2A5}$$

$$\sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} n \overline{\gamma}_{inp} \leq K_p^r \quad \forall p \in \mathcal{P} \setminus \{p_z\} \tag{G2A6}$$

$$\sum_{k \in \mathcal{K}} \delta_{kp_z} \leq 1 \tag{G2A7}$$

$$\sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} \gamma_{iknp_z} \leq 1 \quad \forall k \in \mathcal{K} \tag{G2A8}$$

$$\sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} \overline{\gamma}_{inp} \leq 1 \quad \forall p \in \mathcal{P} \setminus \{p_z\} \tag{G2A9}$$

$$\gamma_{iknp} \in \{0, 1\}, \ \delta_{kp} \in \{0, 1\}, \ \chi_i \in \{0, 1\} \tag{G2A10}$$

There is no change in the objective G2A1 compared to G1A. In each iteration, we try to maximize the number of widths being fulfilled exactly by generating the new pattern, while we try to add further order widths to all yet generated patterns in contradiction to Greedy1, were this decision is made statically before solving the model again in each iteration. The width of the new pattern and the number of knifes is observed by the constraints G2A2 and G2A3, while $\chi_i = 1$ enforces the demand of i to be fulfilled exactly by constraint G2A4. Note here that we add a term respecting all yet generated patterns with their fixed pattern multiplicity u_p, while we try to add further widths to all yet generated patterns by using the variable $\overline{\gamma}_{inp}$ in order to fulfil the demand of order $i \in \mathcal{I}'$. Adding further order widths to the already generated patterns is restricted to their remaining length and their remaining number of knifes (constraints G2A5 and G2A6). The uniqueness multiplicity of the currently generated pattern is ensured by the constraint G2A7. The uniqueness multiplicity of the current pattern and the uniqueness item multiplicity n of a width i is ensured by the constraint G2A8, while the uniqueness item multiplicity n of a width i for the already generated patterns is ensured by the constraint G2A9.

Note that we use the variables δ_{kp} and γ_{iknp} defined in the formulation of the model BLM, while the last index for the pattern p is fixed to the current number of patterns z, which increases with every iteration of the algorithm. The pseudocode of Greedy2 is demonstrated below. The parameters u_p and a_{ip} are used to store the solution.

Greedy2$(\mathcal{I}, D_i, w_i, K, W)$

1: $\mathcal{I}' = \mathcal{I}$, $\mathcal{P} = \{\}$
2: $z = 0$, $u_p = 0$, $a_{ip} = 0$
3:
4: **while** $|\mathcal{I}'| > 0$ **do**
5: $\quad z \leftarrow z + 1$
6: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{p_z\}$
7: $\quad$ solve G2A$(\mathcal{I}', \mathcal{P}, z, D_i, w_i, K, W, K_p^r, W_p^r)$
8: $\quad u_{p_z} = \sum_{k \in \mathcal{K}} k\delta_{kp_z}$
9: $\quad a_{ip_z} = \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} n\gamma_{iknp_z}$
10: $\quad W_z^r = W - \sum_{i \in \mathcal{I}'} w_i a_{ip_z}$
11: $\quad K_z^r = K - \sum_{i \in \mathcal{I}'} a_{ip_z}$
12: $\quad$ **for each** $p \in \mathcal{P} \setminus \{p_z\}$ **do**
13: $\quad\quad a_{ip} \leftarrow a_{ip} + \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} n\overline{\gamma}_{inp}$
14: $\quad\quad W_p^r \leftarrow W_p^r - \sum_{i \in \mathcal{I}'} \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} nw_i\overline{\gamma}_{inp}$
15: $\quad\quad K_p^r \leftarrow K_p^r - \sum_{i \in \mathcal{I}'} \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} n\overline{\gamma}_{inp}$
16: $\quad$ **end foreach**
17: $\quad \mathcal{I}' \leftarrow \mathcal{I}' \setminus \{i \in \mathcal{I}' | \chi_i = 1\}$
18: $\quad \mathcal{N}, \mathcal{K} \leftarrow$ getMaxMultiplicities$(\mathcal{I}')$
19: **end while**
20:
21: **return:** z, u_p, a_{ip}

Line 18 is motivated due to the obtainable reduction of variables when solving G1A in the next iteration by reducing the maximum item multiplicity resp. the maximum pattern multiplicity. The numerical results of Greedy2 are presented and compared to the other approaches in Section 3.3.

Example 3 (Demonstrating Greedy2) *The* `Greedy2` *algorithm will be demonstrated on the instance* `fiber13a_9080` *of the* `Fiber` *benchmark set. The number of knifes is unrestricted and the master roll width is given by 9080. The following table summarizes the order widths and the demands of the instance* $I = (\mathcal{I}, D_i, w_i, K, W)$.

1. Iteration

i	w_i	D_i
i_1	1000	158
i_2	985	32
i_3	1250	30
i_4	920	23
i_5	940	16
i_6	923	10
i_7	1100	6
i_8	1050	4

When solving the model `G1A` *(=*`G2A` *for* $z = 1$*) initially, the demand of the order widths* i_5 *and* i_8 *is fulfilled exactly, i. e.,* $\chi_{i5} = 1$ *and* $\chi_{i8} = 1$*. The first pattern* p_1 *will be used four times, the item multiplicity of order* i_5 *is equal to four and the item multiplicity of order* i_8 *is equal to one. Afterwards, the orders* i_5 *and* i_8 *are removed from* $\mathcal{I}'$*, i. e.,* $\mathcal{I}' = \{i_1, i_2, i_3, i_4, i_6, i_7\}$*. At this point, further widths could be added to* p_1*, since there are 4270 length units unused (i. e.,* $W^r_{p_1} = 4720$*). Note that* `Greedy1` *would add the first order with item multiplicity four to* p_1*. The decision which orders are added is now redirected to the first call of model* `G2A` *in the next iteration.*

2. Iteration

i	w_i	D_i
i_1	1000	158
i_2	985	32
i_3	1250	30
i_4	920	23
i_6	923	10
i_7	1100	6

In this iteration, three order widths can be fulfilled exactly by generating a new pattern and by adding further order widths to p_1*. The demand for order* i_2 *will be fulfilled by adding* i_2 *with frequency three to* p_1*, which is used four times and by adding* i_2 *with item multiplicity two to the pattern* p_2*, which will be used ten times. In addition, adding* i_3 *with item multiplicity three and* i_6 *with item multiplicity one to* p_2 *will fulfil the demand for* i_3 *and* i_6 *exactly. All three order widths are removed from* $\mathcal{I}'$*. The remaining length in* p_1 *is* $W^r_{p_1} = 1315$ *and in* p_2 *is* $W^r_{p_2} = 2437$.

3. Iteration

i	w_i	D_i
i_1	1000	158
i_4	920	23
i_7	1100	6

The first order i_1 *will be added to the pattern* p_2 *with item multiplicity two, reducing the remaining length of* p_3 *to* $W^r_{p_2} = 437$*. Adding* i_1 *with item multiplicity six to the new pattern* p_3 *which is used 23 times will fulfil the demand of* i_1 *exactly. By adding* i_4 *with item multiplicity one to* p_3*, also the demand of* i_4 *will be fulfilled exactly. The remaining length of* p_3 *will be* $W^r_{p_3} = 2160$.

4. Iteration

i	w_i	D_i
i_7	1100	6

The remaining width i_7 *can be added to all yet generated patterns theoretically, but there would be an underproduction when adding* i_7 *to* p_1 *and an overproduction when adding* i_7 *to* p_2 *or* p_3*. In consequence, a fourth pattern must be generated.*

While `Greedy2` *is a heuristic approach of course, note that the solution* $z = 4$ *(summarized in the following table) is also the optimal solution for this instance.*

	p_1	p_2	p_3	p_4	
$w_i \backslash u_p$	*4*	*10*	*23*	*6*	D_i
$w_1 = 1000$	*0*	*2*	*6*	*0*	*158*
$w_2 = 985$	*3*	*2*	*0*	*0*	*32*
$w_3 = 1250$	*0*	*3*	*0*	*0*	*30*
$w_4 = 920$	*0*	*0*	*1*	*0*	*23*
$w_5 = 940$	*4*	*0*	*0*	*0*	*16*
$w_6 = 923$	*0*	*1*	*0*	*0*	*10*
$w_7 = 1100$	*0*	*0*	*0*	*1*	*6*
$w_8 = 1050$	*1*	*0*	*0*	*0*	*4*
$\sum w_i a_{ip}$	*7765*	*8643*	*6920*	*1100*	

Algorithm Greedy3
This algorithm represents a variation of `Greedy2`, *i. e.*, we do not modify the idea of the heuristic itself, but we have seen that some constraints in the model `G1A` resp. `G2A` can be omitted in order to keep the heuristic as slim and simple as possible. Afterwards, we studied the effect on the solution time and solution quality (which seems to increase slightly). In particular, the constraints `G1A6` of the model `G1A` resp. the constraints `G2A8` and `G2A9` of the model `G2A`, ensuring the uniqueness multiplicity k and item multiplicity n of the width i in the pattern p according to the definition of the variable γ_{iknp}, can be omitted (Lemma 5). We will focus on `Greedy2` resp. the model `G2A` for this variation, as the solution quality is already superior compared to `Greedy1`. Note that `Greedy3` will find divergent solutions compared to `Greedy2` in general, since the solver may decide for differing order widths to be fulfilled exactly in the first iterations (but with the identical objective value) by omitting the constraints, which will affect the decisions and the solution quality in later iterations. For the same reason, the proposed `Greedy` algorithms are also sensitive with regard to the sequence of the constraints in the implementation.

Lemma 5 (Uniqueness of Width Frequency) *The constraint `G1A6` of the model `G1A` resp. the constraints `G2A8` and `G2A9` of the model `G2A`, ensuring the uniqueness item multiplicity n of an order width i in a pattern p (and the uniqueness multiplicity k of p), can be omitted.*

Proof. We will proof that the constraint `G1A6` of the model `G1A` can be omitted (in order to proof that the constraints `G2A8` and `G2A9` of the model `G2A` can be omitted, one can proceed analogously). First of all, note that the uniqueness multiplicity k of p with respect to the variable γ_{iknp} is ensured by the constraint `G1A2` in combination with `G1A5`, *i. e.*, $\gamma_{iknp} = 1$ and $\gamma_{ik'np} = 1$ only for $k = k'$. In consequence, omitting the constraint `G1A6` will not affect the uniqueness of the pattern multiplicity k.

Every feasible solution δ_{kp}, γ_{iknp} of the model `G1A` without the constraints ensuring the unique item multiplicity n of an order width i in pattern p has a corresponding feasible solution in the non relaxed model `G1A` (and vice versa). Assuming there is a solution γ_{iknp} of the relaxed model, where the item multiplicity of the width i in the pattern p is not uniquely determined, *i. e.*,

$$\exists (i,p) \in \mathcal{I}' \times \mathcal{P} \quad \text{with} \quad \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} \gamma_{iknp} > 1.$$

We define n_{ip} as the total item multiplicity of the width i in pattern p, *i. e.*,

$$n_{ip} := \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} n \gamma_{iknp} \qquad \forall (i,p) \in \mathcal{I}' \times \mathcal{P}.$$

Obviously, $n_{ip} \leq |\mathcal{N}|$, since the total frequency is restricted by the maximum item multiplicity $|\mathcal{N}|$ derived from the constraints `G1A2` or `G1A3`. In addition, $\sum_{i \in \mathcal{I}'} n_{ip} \leq K,\ \forall p \in \mathcal{P}$, since γ_{iknp} is a feasible solution.

One can now construct the corresponding feasible solution to the non relaxed model by redefining the solution in the following way:

$$\gamma_{iknp} = 0,\ \forall (i,k,n,p) \in \mathcal{I}' \times \mathcal{K} \times \mathcal{N} \setminus \{n_{ip}\} \times \mathcal{P}$$

and

$$\gamma_{ikn_{ip}p} := \begin{cases} 1, \text{ if } \delta_{kp} = 1 \\ 0, \text{ else.} \end{cases} \qquad \forall (i,k,p) \in \mathcal{I}' \times \mathcal{K} \times \mathcal{P}.$$

In consequence, the solution gets feasible to the constraint `G1A6` (which ensures the uniqueness of the item multiplicity n), while the feasibility to the other constraints is not affected. We will demonstrate this exemplarily for the constraint `G1A2` observing the number of knifes (the feasibility to the other constraints can be verified analogously). Note that the values of the variable γ_{iknp} are redefined as explained above, while the values of δ_{kp} are not affected, since the pattern multiplicity must not be redefined:

$$K\delta_{kp} \geq \sum_{i \in \mathcal{I}'} n_{ip}\delta_{kp} = \sum_{i \in \mathcal{I}'} n_{ip}\gamma_{ikn_{ip}p}\delta_{kp} = \sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} n\gamma_{iknp}\delta_{kp} \qquad \forall (p,k) \in \mathcal{P} \times \mathcal{K}.$$

It holds $\gamma_{iknp}\delta_{kp} = \gamma_{iknp}$, since $\gamma_{iknp} = 1$ implies $\delta_{kp} = 1$, because the pattern multiplicity k for the pattern p is uniquely determined. If $\gamma_{iknp} = 0$, the product $\gamma_{iknp}\delta_{kp}$ will be zero, too (while the value of δ_{kp} might be zero or one). In consequence, we can derive

$$K\delta_{kp} \geq \sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} n\gamma_{iknp}\delta_{kp} = \sum_{i \in \mathcal{I}'} \sum_{n \in \mathcal{N}} n\gamma_{iknp} \qquad \forall (p,k) \in \mathcal{P} \times \mathcal{K}.$$

Note that the uniqueness of the pattern multiplicity k is of particular importance when verifying the feasibility of the redefined solution to the demand fulfilment constraint, as the index k is also used as a factor in this equation. □

Consequently, one could also omit the constraint `BLM7` of the model formulation `BLM` in Section 2.1.3. However, the effect on the solution time was clearly negative when investigating the solution time of `BLM` on a small amount of instances.

Note 7 (Approximation Quality of Greedy3) *In order to evaluate the quality of the solutions returned by a heuristic algorithm, it is of special interest to determine the approximation factor t. The approximation factor for a minimization problem is defined as the smallest number t with $z \leq t \cdot z^{opt}$, where z denotes the objective function value of the heuristic solution returned by* `Greedy3` *and z^{opt} denotes the corresponding optimal objective function value for an arbitrary problem instance (absolute performance guarantee).*

However, note that the proposed algorithm `Greedy3` *is sensitive with respect to the sequence of the constraints or the arrangement of the data, i. e., the algorithm will return different solutions for the same problem instance if the constraints or the data are rearranged. That is because there are in general several solutions with identical objective value in the first iterations (e. g., the maximum number of orders being fulfilled exactly in the first iteration is equals three by either adding i_2, i_5 and i_{13} or by adding i_3, i_4 and i_{10} to the current pattern). Those decisions in the first iterations have a major influence on the results in the later iterations, leading to different solutions. In addition, one has no insight into the decision process of the solver.*

Both circumstances make it impossible to predict the behaviour of the algorithm deterministically (in the worst case).

Therefore, we are just going to discuss some (weak) lower and upper bounds for the approximation factor t. A lower bound on t can be derived by investigating the behaviour of the algorithm on tiny problem instances.

A. *For problem instances with a single order width, i. e., $|\mathcal{I}| = 1$, the optimal solution has the objective function value $z^{opt} = 1$, while* `Greedy3` *obviously returns the objective function value of $z = 1$, too, as the single order width can be fulfilled exactly by a single pattern in the first iteration. In case of problem instances with a single order width, the approximation factor t is equal to one, consequently.*

B. *For problem instances with exactly two order widths, i. e., $|\mathcal{I}| = 2$, we have to distinguish two cases:*

   1. *Both order widths can be combined in a single pattern, i. e., the minimal number of patterns is $z^{opt} = 1$. Indeed,* `Greedy3` *will return the solution $z = 1$, as both order widths will be combined within the first generated pattern (while the demand will be fulfilled exactly). The approximation factor t is equal to one.*
   2. *It is not possible to combine both order widths in a single pattern, i. e., the minimal number of patterns is $z^{opt} = 2$. In consequence, the solution returned by* `Greedy3` *will fulfil one of the order widths exactly by the first generated pattern, while the remaining order width will be fulfilled exactly by the second generated pattern, i. e., $z = 2$. Consequently, the approximation factor t is equal to one in case of problem instances with two order widths.*

C. *As we have seen, algorithm* `Greedy3` *will solve problem instances with a single or two order widths to optimality. However, if $|\mathcal{I}| = 3$, one can construct a problem instance with $z^{opt} = 2$ and $z = 3$, i. e., the (absolute) approximation factor t is greater or equal $3/2$ (but note that the actual performance observed in the computational experiments is in general much better). The order widths i with demands D_i and width w_i of the instance I can be seen in the left table, while the optimal solution and the solution returned by* `Greedy3` *can be seen in the middle or right table, resp. The number of knifes K is unlimited and the width W of the master roll is given by $W = 100$.*

Instance I

i	w_i	D_i
i_1	20	20
i_2	75	10
i_3	80	10

	p_1	p_2	
u_p	10	10	D_i
$w_1 = 20$	1	1	20
$w_2 = 75$	1	0	10
$w_3 = 80$	0	1	10
width	95	100	
cutoff	5	0	

	p_1	p_2	p_3	
u_p	20	10	10	D_i
$w_1 = 20$	1	0	0	20
$w_2 = 75$	0	1	0	10
$w_3 = 80$	0	0	1	10
width	20	75	80	
cutoff	80	25	20	

Note that `Greedy3` *does not find the optimal solution, as the first order i_1 is fulfilled in the first iteration with pattern multiplicity 20, while it would be better to fulfil the second and third order in the first resp. second iteration, as the first order can be added to these patterns in the third iteration without generating a third pattern. Also note that* `Greedy3` *may find the optimal solution, if the data or the sequence of the constraints are rearranged due to the sensitivity of the algorithm mentioned in the beginning of this note.*

We will not have any further insights by investigating problem instances with $|\mathcal{I}| = 3$

and $z^{opt} = 1$, as the algorithm will return the solution $z = 1$, since all order widths can be combined in a single pattern.

Determining an upper bound on the factor t is far more difficult for the reasons mentioned at the beginning of this note (while determining an upper bound is also of far more interest). However, it is clear that ***Greedy3*** *will never return a solution worse than the upper bound $|\mathcal{I}|$ on the number of patterns, as the number of iterations is equal to $|\mathcal{I}|$ in the worst case at and least one order width is being fulfilled in each iteration (weak relative performance guarantee).*

As the suggested greedy heuristics provide solutions in very short time but in general without finding or proving optimal solutions, one should consider to exploit the heuristic solution by using it as initial solution when solving an exact model formulation in order to calculate optimal solutions (theoretically) and to get a better estimation of the solution quality by observing the lower bound reported by the solver.

Exact Approach G3-BLM: Combining Greedy3 with BLM
Each instance is solved by the `Greedy3` heuristic initially in order to warm start the solver based on the `BLM` formulation afterwards. Note that the lower bounds motivated by Lemma 3 can be improved due to a better upper bound z^{up} (*i. e.*, the solution returned by `Greedy3`) on the number of patterns. Also note that the solution must be sorted in advance to be feasible for the formulation `BLM`, as the model includes the pattern ordering constraints `BLM5` resp. `BLM6`.

G3-BLM $(\mathcal{I}, D_i, w_i, W, K)$

1: $z^{up}, \delta_{kp}, \gamma_{iknp} \leftarrow$ `Greedy3`$(\mathcal{I}, D_i, w_i, W, K)$ ▷ get a feasible solution
2:
3: $\delta_{kp}, \gamma_{iknp} \leftarrow \text{sort}(\delta_{kp}, \gamma_{iknp})$
4: $l_s \leftarrow \text{getLowerBounds}(D_i, w_i, z^{up})$
5: $z \leftarrow$ solve `BLM`$(\mathcal{I}, D_i, w_i, W, K, \delta_{kp}, \gamma_{iknp}, l_s)$ ▷ warm start `BLM`
6:
7: **return:** z

The numerical results of `G3-BLM` are discussed in Section 3.3.

2.2.3. Partitioning Algorithms

In this Subsection, we present a novel approach for splitting the original problem instance into smaller and less complex derived sub-instances. While there are many potential criteria on how to split the original instance, *e. g.*, grouping the order widths by some characteristics like the demand levels (see also [15]), this approach is motivated by the fact, that the performance when solving an instance based on the monolithic `BLM` formulation decreases significantly, if the demand levels are high on average. In addition, a monolithic model formulation based on the `BLM` formulation, but with much fewer variables, is presented.

Demand Dividing Algorithm - DDA
The idea of this heuristic approach is to reduce the potential maximum multiplicity of the patterns by reducing the demand levels D_i in order to reduce the number of variables

significantly when solving an instance based on the BLM formulation afterwards, leading to a considerable reduce of the solution time.

Therefore, we consider the representation $D_i = f_i \cdot v + r_i$ of the demands with the divisor $v \in \mathbb{N}_0$, quotient $f_i \in \mathbb{N}_0$ and remainder $r_i \in \{0, ..., v-1\}$. From the original instance $I = (\mathcal{I}, D_i, w_i, K, W)$, we derive the following instances: $I^r = (\mathcal{I}^r, r_i, w_i, K, W)$, keeping the remainders as demand and $I^{fv} = (\mathcal{I}^{fv}, f_i \cdot v, w_i, K, W)$, keeping $f_i \cdot v = D_i - r_i$ as demand. Note that $i \in \mathcal{I}^r \Leftrightarrow r_i > 0,\ \forall i \in \mathcal{I}$ and $i \in \mathcal{I}^{fv} \Leftrightarrow f_i > 0,\ \forall i \in \mathcal{I}$. Obviously, solving both instances separately generates a feasible solution for the original instance I. For the numerical experiments, the divisor is set to $v = 10$ (see also Note 8), leading to a significant reduce of variables for the instance I^r, since the maximum multiplicity is restricted by the maximum demand $v - 1 = 9$. Though, the demands and therefore the maximum multiplicity for the instance I^{fv} are almost unchanged at this point. For instance, if the maximum demand was 158, the maximum demand in the instance I^{fv} is equals 150.

However, instead of solving I^{fv}, one could also solve the instance $I^f = (\mathcal{I}^f, f_i, w_i, K, W)$ with $i \in \mathcal{I}^f \Leftrightarrow f_i > 0,\ \forall i \in \mathcal{I}$ and objective function value z^f to get a feasible solution for I^{fv} with objective function value $z^{fv} = z^f$ (Lemma 6). The number of variables in the model for the instance I^f is much lower, since the maximum demand is divided by v. Hence, the maximum multiplicity is reduced from 150 to 15 for the example above, leading to an enormous reduction of solution time compared to the instance I^{fv} in general.

Example 4 (Demonstrating DDA) *The DDA algorithm will be demonstrated on the instance fiber13a_9080 of the Fiber benchmark set. The number of knifes is unrestricted and the master roll width is given by 9080. The table on the left summarizes the order widths and the demands of the original instance $I = (\mathcal{I}, D_i, w_i, K, W)$.*

By choosing $v = 10$, we construct the instance $I^r = (\mathcal{I}^r, r_i, w_i, K, W)$, holding the remainders when dividing the demand D_i by v and the instance $I^{fv} = (\mathcal{I}^{fv}, f_i \cdot v, w_i, K, W)$, holding the major parts of the demands, i. e., $D_i - r_i$. As mentioned in the description above, instead of solving I^r and I^{fv} separately based on the BLM formulation, we solve I^r and I^f in order to reduce the maximum element in the set $\mathcal{K}$ from 150 to 15, leading to a significantly reduce of variables and equations in the BLM formulation.

Instance I

i	w_i	D_i
i_1	1000	158
i_2	985	32
i_3	1250	30
i_4	920	23
i_5	940	16
i_6	923	10
i_7	1100	6
i_8	1050	4

Instance I^r

i	w_i	D_i^r
i_1	1000	8
i_2	985	2
i_4	920	3
i_5	940	6
i_7	1100	6
i_8	1050	4

Instance I^{fv}

i	w_i	D_i^{fv}
i_1	1000	150
i_2	985	30
i_3	1250	30
i_4	920	20
i_5	940	10
i_6	923	10

Instance I^f

i	w_i	D_i^f
i_1	1000	15
i_2	985	3
i_3	1250	3
i_4	920	2
i_5	940	1
i_6	923	1

The following table summarizes the optimal solutions of the instance I^r on the left and the optimal solution of I^f on the right.

	p_1	p_2	
$w_i \backslash u_p$	2	3	D_i^r
$w_1 = 1000$	1	2	8
$w_2 = 985$	1	0	2
$w_4 = 920$	0	1	3
$w_5 = 940$	0	2	6
$w_7 = 1100$	0	2	6
$w_8 = 1050$	2	0	4
$\sum w_i a_{ip}^r$	4085	7000	

	p_1	p_2	
$w_i \backslash u_p$	3	1	D_i^f
$w_1 = 1000$	5	0	15
$w_2 = 985$	1	0	3
$w_3 = 1250$	0	3	3
$w_4 = 920$	0	2	2
$w_5 = 940$	0	1	1
$w_6 = 923$	0	1	1
$\sum w_i a_{ip}^f$	5985	7453	

One can now easily construct a feasible solution for the original instance I (shown below) by constructing the unification of all patterns shown above and by multiplying the multiplicity of the patterns in the optimal solution of I^f with the factor $v = 10$.

	p_1	p_2	p_3	p_4	
$w_i \backslash u_p$	2	3	30	10	D_i
$w_1 = 1000$	1	2	5	0	158
$w_2 = 985$	1	0	1	0	32
$w_3 = 1250$	0	0	0	3	30
$w_4 = 920$	0	1	0	2	23
$w_5 = 940$	0	2	0	1	16
$w_6 = 923$	0	0	0	1	10
$w_7 = 1100$	0	2	0	0	6
$w_8 = 1050$	2	0	0	0	4
$\sum w_i a_{ip}$	4085	7000	5985	7453	

While `DDA` is a heuristic approach of course, note that the solution above is also optimal for the original instance I.

Although the `BLM` formulation is applied to solve the sub-instances I^r and I^f, we use the model formulation `MINLP` (see 2.1.1) and the variables $\delta_p \in \{0, 1\}$ (pattern usage indicator), $\mu_p \in \mathbb{Z}_0^+$ (pattern multiplicity) and $\alpha_{ip} \in \mathbb{Z}_0^+$ (item multiplicity of i in p) for now in order to simplify the notation being used in Lemma 6.

Lemma 6 (Dividing the Demand by a common Divisor) *If the solution δ_p^f, μ_p^f and α_{ip}^f is optimal for the instance $I^f = (\mathcal{I}^f, f_i, w_i, K, W)$ with objective function value z^f, than the solution δ_p^{fv}, μ_p^{fv} and α_{ip}^{fv} defined by*

$$\delta_p^{fv} = \delta_p^f, \quad \mu_p^{fv} = v\mu_p^f \quad \text{and} \quad \alpha_{ip}^{fv} = \alpha_{ip}^f \qquad \forall (p, i) \in \mathcal{P} \times \mathcal{I}^{fv}$$

is feasible for the instance $I^{fv} = (\mathcal{I}^{fv}, vf_i, w_i, K, W)$ with objective function value $z^{fv} = z^f$, where v is a positive integer (and $\mathcal{I}^{fv} = \mathcal{I}^f$).

Proof. As there is no change with respect to the values of the variable α_{ip}^{fv} compared to the optimal (and feasible) solution α_{ip}^f, the feasibility to the constraints `MINLP3` and `MINLP4` (observing the width of the master roll and the number of knifes) is obvious. This also applies to the ordering constraints `MINLP5` and `MINLP6` with respect to the variables δ_p^{fv} and μ_p^{fv}, as μ_p^f is multiplied with a constant factor.

The feasibility of $\delta_p^{fv} = \delta_p^f$ and $\mu_p^{fv} = v\mu_p^f$ with respect to the big M constraint `MINLP6`

is ensured, as the big M constant is defined in dependence of the largest demand level, which is also multiplied by v for the instance I^{fv}. In addition, it holds

$$\sum_{p\in\mathcal{P}} \alpha_{ip}^{fv}\mu_p^{fv} = \sum_{p\in\mathcal{P}} \alpha_{ip}^{f} v\mu_p^{f} = v\cdot\sum_{p\in\mathcal{P}} \alpha_{ip}^{f}\mu_p^{f} = vf_i \qquad \forall i\in\mathcal{I}^{fv}$$

and

$$z^{fv} = \sum_{p\in\mathcal{P}} \delta_p^{fv} = \sum_{p\in\mathcal{P}} \delta_p^{f} = z^{f}.$$

□

The pseudocode of DDA is given on page 37. Note that we introduce a duplicate checking in line 31, as the proposed algorithm performs very bad on instances with broad widths on average compared to the width of the master roll. For instance, an order width which cannot be combined with any other order width while having demand levels greater zero for both instances I^r and I^f increases the number of patterns z by two, while the pattern is a duplicate actually. It is worth to check for duplicates; at least for instances where DDA performs very bad, a reduction by five or six patterns is not uncommon.

The numerical results of the DDA heuristic are presented and compared to the other approaches in Section 3.3.

Note 8 (Determining the Divisor v) *The choice to define v statically by $v = 10$ is basically motivated by the enormous reduction of variables in order to calculate optimal solutions for the sub-instances I^r and I^f based on the model formulation BLM afterwards in shorter time. However, one should also consider the following notes:*

- *If there are several orders with demand level greater than 100, there are (much) more elements in the set $\mathcal{K}^f = \{1, ..., \max_{i\in\mathcal{I}^f} f_i\}$ of the quotients (e. g., 57, if the highest demand level is 576) than in the set $\mathcal{K}^r = \{1, ..., \max_{i\in\mathcal{I}^r} r_i\}$ of remainders (where the largest element is $v - 1 = 9$ in the worst case). Since these sets for the pattern multiplicity have a major influence on the number of variables, the static choice of v will probably result in unbalanced instances I^r and I^f (while the number of variables also depends on the number of orders in each sub-instance, of course). In consequence, the solution times for unbalanced sub-instances vary significantly. Instead of the statical decision, one could define the divisor v depending on the highest demand level, i. e., $v = \lceil\sqrt{\max_{i\in\mathcal{I}} D_i}\rceil$. This will lead to more balanced sub-instances I^f and I^r, since the sets $\mathcal{K}^f$ and $\mathcal{K}^r$ are more balanced, as the largest element in both sets is approximately equals v.*

- *Based on numerical results, we have observed that it is slightly preferable if more order widths occur exclusively in I^f or either I^r. Thus, we suggest to define the divisor v as the (greatest) number dividing the most demand levels without rest. However, to guarantee a decrease of variables and solution time, v should be greater than a specific bound, e. g., greater than five. The drawback of this approach is, that the divisor v will be close to its (user defined) lower bound in general, as the smaller numbers are more likely to divide more demand levels without rest. Thus, the reduction in terms of variables and solution time is smaller on the other hand.*

DDA$(\mathcal{I}, D_i, w_i, K, W, v)$

$\mathcal{I}^r = \{\}$, $\mathcal{I}^f = \{\}$

for each $i \in \mathcal{I}$ **do** ▷ define sets for the remainder model
  $D_i^r = \text{mod}(D_i, v)$ ▷ keep only the remainders as demand
  if $D_i^r > 0$ **then**
    $\mathcal{I}^r \leftarrow \mathcal{I}^r \cup \{i\}$
  end if
end foreach
$\mathcal{P}^r = \{p_1, ..., p_{|\mathcal{I}^r|}\}$
$\mathcal{K}^r = \{1, ..., \max_{i \in \mathcal{I}^r} D_i^r\}$
$\mathcal{N}^r = \{1, ..., \min\left[\max_{i \in \mathcal{I}^r} \lfloor W/w_i \rfloor ; K\right]$

if $|\mathcal{I}^r| > 0$ **then** ▷ solve only for the remaining demands
  $\delta_{kp}^r, \gamma_{iknp}^r \leftarrow$ solve `BLM`$(\mathcal{I}^r, \mathcal{P}^r, \mathcal{K}^r, \mathcal{N}^r, D_i^r, w_i, K, W)$
end if

for each $i \in \mathcal{I}$ **do** ▷ define sets for the quotients model
  $D_i^f = \text{floor}(D_i/v)$ ▷ keep only the quotients as demand
  if $D_i^f > 0$ **then**
    $\mathcal{I}^f \leftarrow \mathcal{I}^f \cup \{i\}$
  end if
end foreach
$\mathcal{P}^f = \{p_1, ..., p_{|\mathcal{I}^f|}\}$
$\mathcal{K}^f = \{1, ..., \max_{i \in \mathcal{I}^f} D_i^f\}$
$\mathcal{N}^f = \{1, ..., \min\left[\max_{i \in \mathcal{I}^f} \lfloor W/w_i \rfloor ; K\right]$

if $|\mathcal{I}^f| > 0$ **then** ▷ solve only for the quotients demands
  $\delta_{kp}^f, \gamma_{iknp}^f \leftarrow$ solve `BLM`$(\mathcal{I}^f, \mathcal{P}^f, \mathcal{K}^f, \mathcal{N}^f, D_i^f, w_i, K, W)$
end if

$z \leftarrow \text{duplicateCheck}\left(\delta_{kp}^r, \gamma_{iknp}^r, \delta_{kp}^f, \gamma_{iknp}^f\right)$

return: z

For the same reasons as mentioned in preface of `G3-BLM`, one should consider to use the solution returned by `DDA` as initial solution when solving a problem instance based on the `BLM` formulation.

Exact Approach DDA-BLM: Combining DDA with BLM
Each instance is solved by the `DDA` heuristic initially in order to warm start the solver based on the `BLM` formulation afterwards with the solution returned by `DDA`. Note that the lower bounds motivated by Lemma 3 can be improved due to a better upper bound z^{up} (*i. e.*, the solution returned by `DDA`) on the number of setups. Also note that the solution must be sorted in advance to be feasible for the formulation `BLM`, as the model includes the pattern ordering constraints `BLM5` resp. `BLM6`. In addition, one must construct the feasible

solution for BLM with respect to the original instance I in advance, since DDA returns only the solutions for the sub-instances I^f and I^r.

DDA-BLM$(\mathcal{I}, D_i, w_i, W, K)$

1: $z^{up}, \delta_{kp}, \gamma_{iknp} \leftarrow$ transform(DDA$(\mathcal{I}, D_i, w_i, W, K)$) ▷ get a feasible solution
2:
3: $\delta_{kp}, \gamma_{iknp} \leftarrow$ sort$(\delta_{kp}, \gamma_{iknp})$
4: $l_s \leftarrow$ getLowerBounds(D_s, w_s, z^{up})
5: $z \leftarrow$ solve BLM$(\mathcal{I}, D_i, w_i, W, K, \delta_{kp}, \gamma_{iknp}, l_s)$ ▷ warm start BLM
6:
7: **return:** z

At the end of this section, we want to present an alternative version of the model formulation BLM with much fewer (binary) variables in total, also motivated by the $D_i = f_i v + r_i$ demand representation.

Exact Model Formulation with fewer Variables

The performance of the model BLM decreases significantly, if the demand levels D_i are high on average (due to large values of $|\mathcal{K}|$ and D_i, the number of variables and constraints increases drastically), *e. g.*, the instance C4 of the Kallrath benchmark set with only four order widths but with demands ranging up from 97 to 610 is solved to optimality in about 30 seconds, while other instances with similar size in terms of $|\mathcal{I}|$, but with much smaller demand levels, are solved to optimality in less than a second.

Based on the idea of splitting the problem instance I into the instances I^f and I^r by exploiting the representation $D_i = f_i v + r_i$ of the demand levels, we suggest to split the variables δ_{kp} and γ_{iknp} of the BLM formulation by the index k into an index set referring to f_i and an index set referring to r_i. Therefore, we introduce the following variables:

$$\delta^f_{kp} := \begin{cases} 1, & \text{if } p \text{ is used exactly } kv \text{ times with regard to the quotients } f_i \\ 0, & \text{otherwise,} \end{cases}$$

resp.

$$\gamma^f_{iknp} := \begin{cases} 1, & \text{if } p \text{ contains } i \text{ exactly } n\text{-times and is used exactly } kv \text{ times} \\ & \text{with regard to the quotients } f_i \\ 0, & \text{otherwise,} \end{cases}$$

where $k \in \mathcal{K}^f_0 := \{0, 1, ..., \max_{i \in \mathcal{I}^f} f_i\}$ and

$$\delta^r_{kp} := \begin{cases} 1, & \text{if } p \text{ is used exactly } k \text{ times with regard to the remainders } r_i \\ 0, & \text{otherwise,} \end{cases}$$

resp.

$$\gamma^r_{iknp} := \begin{cases} 1, & \text{if } p \text{ contains } i \text{ exactly } n\text{-times and is used exactly } k \text{ times} \\ & \text{with regard to the remainders } r_i \\ 0, & \text{otherwise,} \end{cases}$$

where $k \in \mathcal{K}^r_0 := \{0, 1, ..., \max_{i \in \mathcal{I}^r} r_i\}$. The binary variable δ_p, introduced in Section 2.1.1, indicates the usage of the pattern $p \in \mathcal{P}$. By the use of the previous variables, we formulate the following binary linear program (constraints BLMFV1 - BLMFV15):

Binary **L**inear **M**odel with **F**ewer **V**ariables - `BLMFV`:

$$\min z = \sum_{p\in\mathcal{P}} \delta_p \qquad \text{(BLMFV1)}$$

s. t.

$$\sum_{p\in\mathcal{P}}\sum_{n\in\mathcal{N}}\left(\sum_{k\in\mathcal{K}_0^f} kvn\gamma_{iknp}^f + \sum_{k\in\mathcal{K}_0^r} kn\gamma_{iknp}^r\right) = D_i \qquad \forall i\in\mathcal{I} \qquad \text{(BLMFV2)}$$

$$\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} w_i n\gamma_{iknp}^f \le W\delta_{kp}^f \qquad \forall (k,p)\in\mathcal{K}_0^f\times P \qquad \text{(BLMFV3)}$$

$$\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} w_i n\gamma_{iknp}^r \le W\delta_{kp}^r \qquad \forall (k,p)\in\mathcal{K}_0^r\times P \qquad \text{(BLMFV4)}$$

$$\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} n\gamma_{iknp}^f \le K\delta_{kp}^f \qquad \forall (k,p)\in\mathcal{K}_0^f\times P \qquad \text{(BLMFV5)}$$

$$\sum_{i\in\mathcal{I}}\sum_{n\in\mathcal{N}} n\gamma_{iknp}^r \le K\delta_{kp}^r \qquad \forall (k,p)\in\mathcal{K}_0^r\times P \qquad \text{(BLMFV6)}$$

$$\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_0^f} \gamma_{iknp}^f \le 1 \qquad \forall (i,p)\in\mathcal{I}\times\mathcal{P} \qquad \text{(BLMFV7)}$$

$$\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_0^r} \gamma_{iknp}^r \le 1 \qquad \forall (i,p)\in\mathcal{I}\times\mathcal{P} \qquad \text{(BLMFV8)}$$

$$\sum_{k\in\mathcal{K}_0^f} \delta_{kp}^f \le \delta p \qquad \forall p\in\mathcal{P} \qquad \text{(BLMFV9)}$$

$$\sum_{k\in\mathcal{K}_0^r} \delta_{kp}^r \le \delta p \qquad \forall p\in\mathcal{P} \qquad \text{(BLMFV10)}$$

$$\sum_{k\in\mathcal{K}_0^f} \gamma_{iknp}^f - \sum_{k\in\mathcal{K}_0^r} \gamma_{iknp}^r = 0 \qquad \forall (p,i,n)\in\mathcal{P}\times\mathcal{I}\times\mathcal{N} \qquad \text{(BLMFV11)}$$

$$\sum_{k\in\mathcal{K}_0^f} kv\delta_{kp_s}^f + \sum_{k\in\mathcal{K}_0^r} k\delta_{kp_s}^r \ge m^{lo} - \sum_{s'<s}\widetilde{D}_{s'} - \sum_{p\in\mathcal{P},p>s}\left(\sum_{k\in\mathcal{K}_0^f} kv\delta_{kp}^f + \sum_{k\in\mathcal{K}_0^r} k\delta_{kp}^r\right)$$
$$\forall s\in\{1,\ \ldots,\ |\mathcal{I}|\} \qquad \text{(BLMFV12)}$$

$$\sum_{k\in\mathcal{K}_0^f} kv\delta_{k,p+1}^f + \sum_{k\in\mathcal{K}_0^r} k\delta_{k,p+1}^r \le \sum_{k\in\mathcal{K}_0^f} kv\delta_{kp}^f + \sum_{k\in\mathcal{K}_0^r} k\delta_{kp}^r \quad \forall p\in\mathcal{P}\setminus\{p_{|\mathcal{I}|}\} \qquad \text{(BLMFV13)}$$

$$\delta_{p+1} \le \delta_p \qquad \forall p\in\mathcal{P}\setminus\{p_{|\mathcal{I}|}\} \qquad \text{(BLMFV14)}$$

$$\delta_{kp}^f\in\{0,1\},\ \delta_{kp}^r\in\{0,1\},\ \gamma_{iknp}^f\in\{0,1\},\ \gamma_{iknp}^r\in\{0,1\},\ \delta_p\in\{0,1\} \qquad \text{(BLMFV15)}$$

Note that the term "Fewer Variables" refers to the total number of variables in the model formulation, which is smaller compared to `BLM`, as the sets $\mathcal{K}_0^f$ and $\mathcal{K}_0^r$ have much fewer elements compared to $\mathcal{K}$, while the number of different types of variables (see `BLMFV15`) increased, of course.

The number of different patterns shall be minimized (`BLMFV1`). Constraint `BLMFV2` ensures the exact demand fulfilment. The variables γ_{iknp}^f and δ_{kp}^f resp. γ_{iknp}^r and δ_{kp}^r are connected by the constraints `BLMFV3` and `BLMFV4` resp. `BLMFV5` and `BLMFV6`. In addition, the width W of the master roll and the number K of knifes is observed. The uniqueness of the item multiplicity n of width i and the uniqueness of the pattern multiplicity k in pattern p is ensured

for both variables γ^f_{iknp} and γ^r_{iknp} by the constraints `BLMFV7` and `BLMFV8`. Note that the pattern p is used (*i. e.*, $\delta_p = 1$), if the patterns is applied with any multiplicity k (constraints `BLMFV9` and `BLMFV10`), which is enforced to be unique additionally. The constraint `BLMFV11` is of great importance, as it ensures the identical pattern composition of pattern p for both types of variables γ^f_{iknp} resp. γ^r_{iknp}, while the pattern multiplicity k can be varying. Note that it is important to allow for the dummy multiplicity $k = 0$ of a pattern p in order to fulfil the equation, even if the pattern does not contribute anything with regard to the range of the remainders r_i or with regard to the range of the quotients f_i. The model is completed by the symmetry breaking constraints `BLMFV13` and `BLMFV14` and by the constraint `BLMFV12` (see Note 4) for numerical improvement.

Note that the upper and lower bounds on the pattern multiplicity introduced in Section 2.1.2 can only by applied on the variables γ^f_{iknp} and δ^f_{kp} with regard to the quotients f_i. The numerical performance of `BLMFV` is discussed in Section 3.3.

2.3. Setup Minimization & Cutoff Reduction

There are often several optimal solutions to the pattern minimization problem, but with a varying number of master rolls m to be used (see also Example 1). Some of the solutions have unnecessarily large cutoff, because it is not considered at any point in the models and heuristics presented so far. One advantage of the compact model formulation developed in Section 2.1.3 is the easy adaptability. In order to find an optimal solution to the pattern minimization problem (primary goal), while cutting as few master rolls as possible (secondary goal), one can simply adapt the objective function or add constraints to the model formulation. Note that the cutoff reduction is equivalent to the reduction of the number of master rolls being cut, if the overproduction of the order widths is handled as cutoff (therefore we will use both terms synonymously). In order to avoid introducing a further variable for the cutoff per pattern, we will focus on the number of master rolls to be cut which can already be expressed by the variable δ_{kp} introduced in Section 2.1.2.

At first, we are going to present the exact approaches `BLMPG-NR1` and `BLM-NR1`. Both have in common, that the setup minimization problem is solved to optimality initially (based on the `BLMPG` resp. `BLM` formulations introduced in Subsection 2.1.2 resp. 2.1.3). The number of master rolls is reduced afterwards by adapting the objective function while adding an additional constraint to observe the minimal number of patterns.

The monolithic model formulations `BLMPG-NR2` resp. `BLM-NR2` presented afterwards both reduce the number of patterns as well as the number of master rolls being cut simultaneously by adding a penalty term to the objective.

Finally, we suggest a modification of the heuristic approaches `Greedy3` resp. `DDA`, referenced as `Greedy3`* resp. `DDA-NR1` hereafter, in order to include the reduction of master rolls being cut.

Exact Approaches: BLMPG-NR1 and BLM-NR1

The optimal solution z^{opt} of the setup minimization problem obtained by solving `BLMPG` resp. `BLM` initially is used to warm start the solution process of the model `BLMPG-NR1` (defined by the new objective function `BLMPG1`*, constraints `BLMPG2` - `BLMPG4` of the `BLMPG` formulation and completed by the additional constraint `BLMPG5`) resp. the model `BLM-NR1` (defined by the new objective `BLM1`* and the constraints `BLM2` - `BLM11` of the `BLM` formulation). In both models, we try to minimize the number of master rolls to be cut by

$$\min\ m = \sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} k\delta_{kp}. \qquad (\texttt{BLMPG1}^* \text{ resp. } \texttt{BLM1}^*)$$

The minimal number of different patterns to be used is observed by adding the constraint `BLMPG5` to the model `BLMPG-NR1`, *i. e.*,

$$\sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} \delta_{kp} = z^{opt}. \qquad (\texttt{BLMPG5})$$

Note that this constraint is not added to the model formulation `BLM-NR1`, as we can simply reduce the set of patterns by defining $\mathcal{P} = \{p_1, \ldots, p_{z^{opt}}\}$, also leading to a decrease in terms of variables and constraints. In addition, we can improve the lower bounds on the pattern multiplicity motivated by Lemma 3 in the formulation `BLM-NR1`, due to a better upper bound z^{up} on the minimal number of patterns (in particular, the upper bound is z^{opt} itself).

Exact monolithic Formulations: BLMPG-NR2 and BLM-NR2
In this approach, pattern and cutoff minimization are combined directly in a monolithic model formulation, *i. e.*, the number of patterns as well as the number of master rolls is observed by the objective function `BLMPG1`** resp. `BLM1`**, defined as

$$\min\ z + m = \min \underbrace{\sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} \delta_{kp}}_{\in\mathbb{Z}_0^+} + \underbrace{\frac{1}{D^{\mathrm{T}}}\sum_{p\in\mathcal{P}}\sum_{k\in\mathcal{K}} k\delta_{kp}}_{\in(0;1)}. \qquad (\texttt{BLMPG1}^{**}\text{ resp. }\texttt{BLM1}^{**})$$

To prefer pattern minimization (primary goal - the first double sum defines the number of different patterns to be used) over cutoff minimization (secondary goal), the second term defines the number of master rolls to be cut must be scaled down, *e. g.*, by using the inverse of

$$D^{\mathrm{T}} = \sum_{i\in\mathcal{I}} D_i$$

as scaling factor. Note that the second double sum is scaled down to a value between 0 and 1, since the number of master rolls to be cut is equals D^{T} in the worst case.

Heuristic Approach: Greedy3*
This approach is based on the `Greedy3` heuristic suggested in Section 2.2.2, as the results of `Greedy3` are superior compared to `Greedy1` and slightly better compared to `Greedy2`. There are only very limited options to modify the heuristic in order to observe the number of master rolls being cut as secondary objective, as the pattern multiplicity cannot be changed after the generation of a pattern. However, one can reduce the multiplicity of the currently generated pattern easily by adding a penalty term within the objective function. Note that this modification is the only derivation compared to `Greedy3`. In particular, we redefine the objective function `G2A1` in the following way:

$$\max\ I^P = \underbrace{\sum_{i\in\mathcal{I}'} \chi_i}_{\in\mathbb{Z}_0^+} - \underbrace{\frac{1}{1+\max\limits_{i\in\mathcal{I}'} D_i}\sum_{k\in\mathcal{K}} k\delta_{kp_z}}_{\in(0;1)}. \qquad (\texttt{G2A1}^{*})$$

The primary objective is to maximize the number of order widths being fulfilled exactly by the current pattern z, while solutions with a greater amount of master rolls being cut by applying the current pattern are penalised by subtracting a value between zero and one (a higher pattern multiplicity will reduce the objective value more than a pattern with a lower multiplicity). Note that adding one within the denominator is of great importance, as otherwise the solver might return a solution with $I^P = 0$ by generating an empty pattern with multiplicity zero, instead of fulfilling a single order width by a pattern with multiplicity $\max_{i\in\mathcal{I}'} D_i$. In this case, the algorithm will not terminate while increasing the number of patterns incrementally.

The numerical results indicate that one has to do a trade-off between reducing the number of patterns and reducing the number of master rolls, *i. e.*, there is a decrease of the number of master rolls being cut, while the number of patterns increases moderately (see also Section 3.3).

This observation can be explained: Each pattern which is generated in the first iterations fulfils several order widths exactly in general. If there are several order widths

within the currently generated pattern, reducing the pattern multiplicity by increasing the item multiplicity of the orders being part of the current pattern becomes very unlikely. However, in the later iterations, the currently generated pattern often fulfils only a single order width exactly, *e. g.*, by adding an order with width 400 and demand 22 with the item multiplicity one to the pattern which will be used 22 times. If the width of the master roll is equals 1000, the item multiplicity will be equals two and the pattern multiplicity will be equals eleven when running `Greedy3*`. In consequence, less master rolls are being cut. However, as the recently generated pattern in this example has only 200 units of width left, adding further order widths in later iterations to this pattern becomes very unlikely, probably leading to the moderate increase in terms of z (number of patterns).

Heuristic Approach: DDA-NR1

It is worth to modify the `DDA` approach in order to reduce the number of master rolls being cut as secondary objective, as it can be done quite easily (while the solution times increase) and the reduction of master rolls being used is quite significant for some problem classes.

In the `DDA` approach, we basically solve to less complex instances derived of the original instance based on the `BLM` formulation. The idea for reducing the number of master rolls is straightforward: After solving the first derived sub-instance (with demand levels equivalent to the remainders), we warm start the model `BLM-NR1` with the obtained solution of the pattern reduction problem, introduced above, in order to reduce the number of master rolls to be cut in this part of the solution. Afterwards, the second derived sub-instance (with demand levels equivalent to the quotations) is solved. Again, we warm start the model `BLM-NR1` based on the data and the obtained solution for the second sub-instance in order to reduce the number of master rolls. Finally, combining both solutions generates a feasible solution to the original problem instance, but the number of master rolls to be cut is (much) smaller in general compared to the solution of the unmodified `DDA` approach. Note that the number of patterns might differ slightly compared to `DDA` due to the duplicate checking, as the pattern structure varied probably by solving `BLM-NR1` afterwards.

3. Numerical Experiments

In the first Section of this Chapter, we will briefly describe the problem characteristics, *i. e.*, the number of order widths, the range of widths and demand levels as well as the background of the instances in the benchmark classes which are used to evaluate our approaches presented in Chapter 2. Some details of the `GAMS` implementation are mentioned in Section 3.2. In Section 3.3, we will finally discuss and compare the numerical results of our approaches on the benchmark classes among each other and with further approaches published in literature.

3.1. Benchmark Data

The benchmark data is subdivided into five classes named `Kallrath`, `Vanderbeck`, `Fiber`, `Belov` and `Foerster & Wäscher`, each of them consisting of several instances. Note that the instances in the `Kallrath`, `Vanderbeck` and `Fiber` class are from real-world applications, while the instances in the `Belov` and `Foerster & Wäscher` class are generated using a tool called `CUTGEN`, a problem instance generator for the one-dimensional cutting stock problem coded by T. Gau and G. Wäscher (see [10]). Also note that the `Belov` and `Foerster & Wäscher` classes are further subdivided into several subclasses due to the large amount of instances in these classes. In total, we dispose of 2060 instances of the one-dimensional cutting stock problem for setup minimization across all classes, which is a (much) greater amount than in most publications in literature.

Kallrath Instances
The first benchmark set consists of 25 real-world instances of the paper industry. The number of order widths ranges from 1 to 50 (see [5], p. 12). There is a limit on the number of knifes for most instances in this set. The instances can be accessed from the website [1] and are already available as `GAMS` files. Numerical results for this benchmark class are presented in [5] and [6].

Vanderbeck Instances
This benchmark set consists of 16 real-world instances. For a short description on their background see [3], p. 24. Vanderbeck describes these instances as a representative selection of the hardest one-dimensional cutting stock problems that arise in practise ([3], p. 24.). Note that this assessment may no longer apply today, as it originally referred to the year 2000. There is no limit on the number of knifes. These instances have been transformed manually from text files into the `GAMS` format.

Fiber Instances
The problem instances of this benchmark class and the following description are taken from the website [2], Section „Test instances in a chemical fiber company“. Numerical results have been published among others in [7]. There are 39 instances in total (however,

[1] `https://www.sciencedirect.com/science/article/pii/S0377221714002562`
[2] `https://sites.google.com/site/shunjiumetani/benchmark`

the website mentioned 40 for some reason). The number of order widths ranges from 6 to 29, while the demand levels ranges from 2 to 264. The width of the master roll is either 5180 or 9080, while the widths of the orders range from 500 to 2000. There is no limit on the number of knifes. These instances have been transformed manually from text files into the `GAMS` format. The real world background is an application in a chemical fiber company in Japan.

Belov Instances
The problem instances of this benchmark class are taken from the website [3], Section „One-dimensional setup minimization“. See also [8] and [9] for reference. There are 180 problem instances in total, subdivided into nine classes each consisting of 20 problem instances, which have been converted from text files into the `GAMS` format by a `Python` parser, especially written for this purpose only. Table 3.2 summarizes the problem characteristics based on the description in [8], p. 11. The width W of the master roll is equals 10000 and the number of knifes is unrestricted.

Class	$\|\mathcal{I}\|$	min w_i	max w_i	min D_i	max D_i
bel20_1	20	100	7000	1	100
bel50_2	50	100	2000	1	100
bel50_3	50	100	4000	1	100
bel50_4	50	100	7000	1	100
bel50_5	50	2000	4000	1	100
bel50_6	50	2000	7000	1	100
bel50_7	50	100	7000	1	10
bel50_8	50	100	7000	50	100
bel150_9	150	100	7000	1	100

Table 3.2.: Problem Data Characteristics - `Belov` Class

Foerster & Wäscher Instances
This class is by far the benchmark set with the highest amount of problem instances; 1800 in total, subdivided into 18 classes each consisting of 100 instances. The problem instances are taken from the website [4], Section „Randomly generated test instances“. The data is generated by `CUTGEN`, a problem instance generator-tool for one-dimensional cutting stock problems and have been converted from text files into the `GAMS` format by using the parser mentioned above. Computational results have been published among others in [7] and [11] for reference. Table 3.5 summarizes some problem characteristics (see [7], p. 12). The width W of the master roll is equals 10000 for each class. There is no limit on the number of knifes.

[3] `http://www.math.tu-dresden.de/~capad/cpd-ti.html`
[4] `https://sites.google.com/site/shunjiumetani/benchmark`

Class	$\lvert\mathcal{I}\rvert$	min w_i	max w_i	$\varnothing D_i$
type01	10	10	200	10
type02	10	10	200	100
type03	20	10	200	10
type04	20	10	200	100
type05	40	10	200	10
type06	40	10	200	100
type07	10	10	800	10
type08	10	10	800	100
type09	20	10	800	10

Class	$\lvert\mathcal{I}\rvert$	min w_i	max w_i	$\varnothing D_i$
type10	20	10	800	100
type11	40	10	800	10
type12	40	10	800	100
type13	10	200	800	10
type14	10	200	800	100
type15	20	200	800	10
type16	20	200	800	100
type17	40	200	800	10
type18	40	200	800	100

Table 3.5.: Problem Data Characteristics - `Foerster & Wäscher` Class

3.2. Implementation

The model formulations and heuristic approaches (see Table 3.6) proposed in Chapter 2 are implemented with GAMS 25.2.0 (**G**eneral **A**lgebraic **M**odeling **S**ystem).

Approach	Short Description	Exact
PG	Model for Pattern Generation (using the CPLEX Solution Pool Option)	-
BLMPG	Binary Linear Model based on complete Pattern Generation	✓
BLMPG-NR1	Solving BLMPG initially, reducing the Number of Master Rolls subsequently	✓
BLMPG-NR2	Observing the Number of Patterns and Master Rolls in the objective of BLMPG	✓
BLM	Binary Linear Model for Setup Minimization with exact Demand Fulfilment	✓
BLM-NR1	Solving BLM initially, reducing the Number of Master Rolls subsequently	✓
BLM-NR2	Observing the Number of Patterns and Master Rolls in the objective of BLM	✓
BLMEP	Model based on generated Efficient Patterns combined with free Patterns	✓
BLMFV	Binary Linear Model similar to BLM but with Fewer Variables	✓
BLMPG-MMR	Formulation similar to BLMPG, but observing multiple Types of Master Rolls	✓
BLM-MMR	Formulation similar to BLM, but observing multiple Types of Master Rolls	✓
Greedy1	Greedy Algorithm based on the G1A Model	✗
Greedy2	Greedy Algorithm based on the G2A Model	✗
Greedy3	Greedy Algorithm based on the G3A Model	✗
Greedy3*	Greedy Algorithm based on the G3A Model including cutoff reduction	✗
DDA	Demand Dividing Algorithm	✗
DDA-NR1	Demand Dividing Algorithm including cutoff reduction	✗
G3-BLM	Combining Greedy3 with the BLM formulation	✓
DDA-BLM	Combining DDA with the BLM formulation	✓

Table 3.6.: Overview on the different Approaches

The instances of the Kallrath benchmark set were already stored as GAMS files, while the Vanderbeck and Fiber instances have been converted to the GAMS format manually. For the great amount of instances in the Foerster and Wäscher and Belov benchmark set, a simple parser is written with Python and called from the GAMS Embedded Code Facility in order to convert the text files automatically. The orders widths will be sorted in descending order according to their demands levels when using the parser.

The following example code defines the data of the instance C3 of the Kallrath benchmark set, *i. e.*, the set of orders $\mathcal{I}$, the demand levels D_i, the width of the single orders w_i and of the master roll W and the number of knifes K. Note that some labels in the code differ from the notation used in this document, *e. g.*, the length of the master roll is named L.

```
1  Set i(*) 'set of order widths' /
2     'A1'
3     'A2'
4     'A3' /;
5
6  Parameter w(*) 'width of order width i' /
7     'A1' 500
8     'A2' 550
9     'A3' 600 /;
10
11 Parameter D(*) 'demand (number of pieces of width i)' /
12    'A1' 8
13    'A2' 4
14    'A3' 2 /;
```

```
15
16 Scalar L        'length of the master roll' / 2440 /;
17 Scalar MAXKNIFE 'upper bound on the number of knifes' / 7 /;
18 Scalar CTRKNIFE 'if the number of knifes is restricted by MAXKNIFE' / 1 /;
```

Based on those data files, the following GAMS code determines the sets $\mathcal{P}$, $\mathcal{K}$ and $\mathcal{N}$, which have been defined in the beginning of Chapter 2. Those sets are used in all approaches. They are stored in separate files which are included into the model file later on.

```
1  $onEcho > generateSets.gms
2  $include ".\data\data_SR\C%instance%.gms";
3
4  Scalar maxP, maxK, maxN;
5  maxP = card(i);
6  maxK = smax(i, D(i));
7  maxN = min(smax(i, floor(L/w(i))),MAXKNIFE$CTRKNIFE + 99$(not CTRKNIFE));
8
9  File out%instance% / .\data\sets\Sets_SR_%instance%.gms /
10 put  out%instance%;
11 put 'Set p / p1*p'maxP:<10:0 ' /;'/;
12 put 'Set k /  1*' maxK:<10:0 ' /;'/;
13 put 'Set n /  1*' maxN:<10:0 ' /;'/;
14 putClose;
15 $offEcho
16
17 $onEcho > makeGenerateSets.gms
18 Set instance / 1*7 /;
19
20 File generateAll / .\generateAll.gms /;
21 put  generateAll;
22
23 loop(instance,
24    put 'execute "gams generateSets.gms --instance='instance.tl:15'";' /;
25 );
26 putClose;
27 $offEcho
28
29 $call gams makeGenerateSets.gms
30 execute 'gams generateAll.gms';
```

Executing the previous code generates the data files defining the sets for the instances C1 - C7 (which can be specified in the GAMS set directly within the model makeGenerateSets in line 18). Note that this model should be executed only once, *i. e.*, when new data instances are added. The file written for the instance C3 defining the sets $\mathcal{P}$, $\mathcal{K}$ and $\mathcal{N}$ will look as follows.

```
1 Set p / p1*p3 /;
2 Set k /  1*8  /;
3 Set n /  1*4  /;
```

At this point, only the implementation of the BLM formulation is presented, as the different codes are to extensive at this point. Note the time stamps in the code below in line 4 and line 116. The elapsed solution time includes the time for the data input, the preprocessing time (*e. g.*, for variable fixing), the model generation time and the actual time to solve the model. Note that the time for calculating the sets $\mathcal{P}$, $\mathcal{K}$ and $\mathcal{N}$ is not

included, as it is marginal of course.

```
$set instance 3
$set timeLimit 3600

Scalar starttime; starttime = jnow;
$include ".\data\data_SR\C%instance%.gms";
$include ".\data\sets\Sets_SR_%instance%.gms";

Binary Variable
   y(p,k,i,n) 'if p contains i exactly n times and p is used exactly k times'
   v(p,k)     'if p is used exactly k times';

Variable nP 'number of patterns';

Scalar loNR 'lower bound on the number of rolls';
loNR = ceil(sum(i, w(i)*D(i))/L);

Parameter
   loP(p) 'lower bound on the multiplicity of p'
   lo(p)  'dummy parameter used for the calculation of loP';

lo('p1') = ceil(loNR);
loop((p,i)$(ord(p) > 1 and ord(p) < card(p)),
   if(ord(p) = ord(i) + 1,
      lo(p) = lo(p-1) - D(i);
   );
);

loP('p1') = ceil(loNR/card(p));
loop((p,i)$(ord(p) > 1 and ord(p) < card(p)),
   if(ord(p) = ord(i) + 1,
      loP(p) = ceil(lo(p)/(card(p) - ord(i)));
   );
);

* Variable fixing
loop(p,
   loop(k,
      loop(n,
         loop(i,
            if(ord(k) < loP(p),
               v.fx(p,k)     = 0;
               y.fx(p,k,i,n) = 0;
            );
            if(D(i) < loP(p),
               y.fx(p,k,i,n) = 0;
            );
            if(ord(p) = ord(i) and ord(k) > D(i),
               y.fx(p,k,i,n) = 0;
               v.fx(p,k)     = 0;
            );
            if(ord(n) > min{FLOOR[L/W(i)],D(i)}
               or (CTRKNIFE = 1 and ord(n) > MAXKNIFE)
               or ord(n) > D(i),
               y.fx(p,k,i,n) = 0;
            );
         );
      );
   );
```

```
59 );
60
61 Alias (s,p);
62
63 Equation
64    BLM1      'calculation of nP (number of patterns)'
65    BLM2(i)   'exact demand fulfillment of width i'
66    BLM3(p,k) 'observing the width of the master roll'
67    BLM4(p,k) 'observing the number of knifes'
68    BLM5(p)   'symmetry breaking constraint'
69    BLM6(p)   'symmetry breaking constraint (descending pattern multiplicity'
70    BLM7(p,i) 'ensuring the uniqueness frequency and multiplicity'
71    BLM8(p)   'ensuring the uniqueness multiplicity'
72    BLM11(s)  'see note 3 in the document';
73
74 BLM1..
75    nP =e= sum(p, sum(k, v(p,k)));
76
77 BLM2(i)..
78    sum(p, sum(k, sum(n, ord(n)*ord(k)*y(p,k,i,n)))) =e= D(i);
79
80 BLM3(p,k)..
81    sum(i, sum(n, ord(n)*y(p,k,i,n)*w(i))) =l= L*v(p,k);
82
83 BLM4(p,k)$(CTRKNIFE = 1)..
84    sum(i, sum(n, ord(n)*y(p,k,i,n))) =l= MAXKNIFE*v(p,k);
85
86 BLM5(p)$(ord(p) < card(p))..
87    sum(k, v(p+1,k)) =l= sum(k, v(p,k));
88
89 BLM6(p)$(ord(p) < card(p))..
90    sum(k, ord(k)*v(p+1,k)) =l= sum(k, ord(k)*v(p,k));
91
92 BLM7(p,i)..
93    sum((n,k), y(p,k,i,n)) =l= 1;
94
95 BLM8(p)..
96    sum(k, v(p,k)) =l= 1;
97
98 BLM11(s)..
99    sum(k, ord(k)*v(s,k)) =g= loNR - sum(i$(ord(i)<ord(s)), D(i))
100                                  - sum((p,k)$(ord(p)>ord(s)), ord(k)*v(p,k));
101
102 Model BLM / BLM1, BLM2, BLM3, BLM4, BLM5, BLM6, BLM7, BLM8, BLM11 /;
103
104 option
105    mip     = cplex
106    optCr   = 0
107    threads = 4;
108
109 MinNP.resLim = %timeLimit%;
110 $onEcho > cplex.opt
111 names no
112 $offEcho
113 MinNP.optFile = 1;
114 $log Solving %instance%
115 solve MinNP using mip min nP;
116 Scalar elapsed; elapsed = (jnow - starttime)*24*3600;
```

After solving an instance, the following solution report is created to display and verify

the results. Note that the GAMS code generating the solution report is not presented at this point nor in the appendix, as it is extensive and varies slightly for every model formulation.

```
1  Solution Summary    Total        1        2
2  ----------------------------------------------
3  delta  (p used):        2        1        1
4  Number Rolls   :        5        4        1
5  ---------------------------------------------------------------
6  Widhts     Width      aip                    Out(i)   D(i)
7  A1           500                2        0        8      8
8  A2           550                1        0        4      4
9  A3           600                0        2        2      2
10 ---------------------------------------------------------------
11 Max  7 cuts:                    3        2
12 Used:                 7400    1550     1200
13 Offcut:               4800     890     1240
14 ----------------------------------------------
15 Total:               12200    2440     2440
```

The minimum number of different setups for the instance C3 is two. The first pattern, containing the order A1 two times and the order A2 once, will be used four times in total, the second pattern containing the order A3 two times will be used only once. There are also some additional input data, among them the number of cuts and the cutoff for each pattern and the number of master rolls used in total.

By running the following code, several instances (entered in line 7) are solved automatically in turn, while also generating a compact solution report listening all instances with their solution status and the cutting plan for each instance. One can specify the time limit in seconds per instance and the model resp. heuristic solution approach in the first two lines.

```
1  $set timeLimit 3600
2  $set model BLM
3
4  $onEcho > makeTestRun.gms
5  Set instance
6  /
7     1*7
8  /;
9
10 File executeTestRun / executeTestRun.gms /;
11 put  executeTestRun;
12
13 loop(instance,
14 put 'execute "gams %model% --instance=' instance.tl:15 ' --timeLimit='%
       timelimit%'";' /;
15 );
16 putClose;
17
18 Execute 'del results%model%.txt';
19 File results / results%model%.txt /;
20 put  results;
21 put 'Instance     nP     relGap     absGap     solStatus     solTime' /;
22 $offEcho
23
24 $call gams makeTestRun.gms lo=%gams.lo%
```

```
25 $ifE errorLevel<>0 $abort Error generating test run!
26
27 $call gams executeTestRun.gms lo=%gams.lo%
28 $ifE errorLevel<>0 $abort Error executing test run!
```

After solving the seven instances of the `Kallrath` benchmark set specified above, the following report is created, summarizing the number of patterns and the number of master rolls being cut, solution status, absolute and relative gap and the solution time for each instance:

```
1 Instance        nP      nR      relGap      absGap      solStatus      solTime
2 1             1.00       1        0.00        0.00           1.00         0.18
3 2             2.00      21        0.00        0.00           1.00         0.18
4 3             2.00       4        0.00        0.00           1.00         0.17
5 4             4.00     707        0.00        0.00           1.00        27.54
6 5             2.00       5        0.00        0.00           1.00         0.18
7 6             4.00      33        0.00        0.00           1.00         3.31
8 7             6.00      73        0.00        0.00           1.00         1.51
```

All instances specified in this example are solved to optimality, as the relative gap (relGap) resp. the absolute gap (absGap) are equal to zero. In addition, the solution status equal to 1 indicates optimality and successful termination.

Although each approach generates a feasible solution theoretically, note that the feasibility of a solution, stored within the parameters u_p and a_{ip}, is always checked after solving an instance just in case. The following code adds a note to the solution report, if the demand is not fulfilled exactly or if the width of a patterns exceeds the width of the master roll or if the number of cuts exceeds the number of knifes.

```
1 Parameter
2    used(p)  'width of the pattern p'
3    aip(i,p) 'frequency of width i in pattern p'
4    cuts(p)  'number of cuts in pattern p'
5
6 used(p)  = sum(i, sum(n, Sum(k, ord(n)*y.l(p,k,i,n)*w(i))));
7 aip(i,p) = sum(n, sum(k, ord(n)*y.l(p,k,i,n)));
8
9 Parameter
10   errorD(i) 'over- or underproduction of width i'
11   errorW(p) 'if a pattern is not valid';
12
13 errorD(i) = abs(D(i) - sum(p, u(p)*aip(i,p)));
14 errorW(p) = 1$(used(p) > L or (cuts(p) > MAXKNIFE)$(CTRKNIFE = 1));
15 if(sum(i, errorD(i)) > 0 or sum(p, errorW(p)) > 0,
16    put 'No feasible solution!';
17 );
```

3.3. Results

The numerical experiments have been performed on an Intel i5-7200 2.5GHz with 8GB RAM, using four threads. As mentioned before, the model formulations and heuristics are implemented with `GAMS 25.2.0` and solved by `CPLEX 12.8.0.0` (running on default setting, even so aggressive `Probing` seems to be beneficial). The elapsed time in seconds includes the data import, pre- and post-processing steps, model generation and the actual model solution time. Note that it is important not only to concern the actual model solution time reported by the solver, because some instances are solved quite fast, *e. g.*, with the `BLMPG` formulation, but the pre- and post-processing time takes up a multiple of the time in order to handle a large amount of generated patterns.

Table 3.7 summarizes the different approaches for which we present and discuss numerical results. In addition, we compare our results to those published in literature. Note that `Greedy3` as well as the approaches with suffix `NR1` are superior in comparison with the additional approaches like `Greedy1` or `BLM-NR2` summarized in Table 3.6.

Approach	Short Description	Exact
`BLMPG`	Binary Linear Model based on complete Pattern Generation	✓
`BLMPG-NR1`	Solving `BLMPG` initially, reducing the Number of Master Rolls subsequently	✓
`BLM`	Binary Linear Model	✓
`BLM-NR1`	Solving `BLM` initially, reducing the Number of Master Rolls subsequently	✓
`BLMFV`	Binary Linear Model similar to `BLM` but with Fewer Variables	✓
`Greedy3`	Greedy Algorithm based on the `G3A` Model	✗
`Greedy3`*	Greedy Algorithm based on the `G3A` Model including cutoff reduction	✗
`DDA`	Demand Dividing Algorithm	✗
`DDA-NR1`	Demand Dividing Algorithm including cutoff reduction	✗
`G3-BLM`	Combining `Greedy3` with the `BLM` formulation	✓

Table 3.7.: Overview on the different Approaches

Basically, we will distinguish between two settings:

1. In order to obtain optimal solutions for all real-world problems, we are going to solve the instances of the `Kallrath`, `Vanderbeck` and `Fiber` benchmark class by running the model formulations `BLMPG` (if applicable) and `BLM` with a time limit of 86400 seconds on each instance. In addition, the results of the `Greedy3` and `DDA` heuristics (with $v = 10$) are added for comparison. The results are presented within the Tables 3.8, 3.9 and 3.10. Afterwards, the results of the approaches `BLMPG-NR1`, `BLM-NR1`, `Greedy3`* and `DDA-NR1` for minimizing the number of different patterns (primary objective) and for minimizing the number of master rolls being cut (secondary objective) are presented (see Tables 3.11, 3.12 and 3.13). The time limit per instance is 86400 seconds, too.

2. In the second setting, we include all 2060 instances distributed over 30 classes. The time limit per instance is 3600 seconds. Table 3.14 summarizes the results for the approaches `BLMPG`, `BLM`, `Greedy3` and `DDA` (with $v = 10$) on pure setup minimization, while Table 3.16 summarizes the results for the approaches `BLMPG-NR1`, `BLM-NR1`, `Greedy3`* and `DDA-NR1` (with $v = 10$) for minimizing the number of different patterns (primary objective) and for minimizing the number of master rolls being cut (secondary objective). Except for `Greedy3` resp. `Greedy3`*, some approaches might

not be applicable on several classes, *e. g.*, as it is not possible to generate all patterns for BLMPG or as the solver does not find a (promising) feasible solution within one hour based on the BLM formulation. In the worst case, BLM terminates during model generation due to an out of memory abort, *e. g.*, when trying to solve problem instances with 150 order widths. If an instances cannot be solved to optimality within the given time limit, the current solution found will be reported as result and the execution is terminated.

The results of some other approaches like G3-BLM and BLMFV will be discussed separately later on.

In order to interpret the following result tables appropriately, we have to make some remarks in advance. As they apply to all tables, they will be listed exclusively here:

- The average number of patterns per benchmark class is denoted by $\overline{z}$, the average number of master rolls being cut is denoted by $\overline{m}$.
- If the number of patterns is optimal, the value for z is **bold**. This also applies to whole benchmark classes, *i. e.*, if all instances within a class are solved to optimality, the average number of patterns $\overline{z}$ will be **bold**, too.
- The average number of patterns will be ***cursive and bold*** resp. *cursive*, if at most 10 % resp. at most 50 % of a benchmark class are not solved to optimality.
- Both previous remarks also apply to the number m of master rolls being cut. However, note that we focus on pattern reduction primarily, *i. e.*, the value of m or $\overline{m}$ will only be marked as optimal, if the corresponding value of z resp. $\overline{z}$ is optimal at first. Also note that even in case the value for m or $\overline{m}$ is marked as optimal, the value is in general much worse than the optimal solution of the pure cutoff reduction problem, as pattern reduction is the primary objective.
- The superscript [a] resp. [b] indicate, that the calculation is based only on a subset of the instances of a specific class, *i. e.*, only for the first 20 out of 100 instances resp. for the first 5 out of 20 instances. This is a restriction for practical reasons, as we lack of time to compute some results for classes where the average solution time is high.
- If an approach is in general not applicable, *i. e.*, the majority of the instances of a class cannot be solved with a specific approach, *e. g.*, as it is by far not possible to generate all patterns in advance for BLMPG, the instance/class is marked by a ✗.
- If an approach is in general applicable, *i. e.*, the majority of the instances of a class can be solved (not necessarily to optimality within 3600 seconds) with a specific approach (but not all instances), the instance/class is marked by a —. Therefore, we exclude those results from the overall comparison in order to be on the safe side. Some examples are: 21 out of 25 instances of the Kallrath class can be solved to optimality quite fast with BLMPG (see Table 3.8) or 9 out of 10 instances of the bel20_1 class (see Table 3.2 resp. 3.14) are solved with a small relative gap on average within 3600 seconds by BLMPG, while the tenth instances is aborted due to an out of memory error. Both fields will be marked by a — in Table 3.14 and 3.16. Note that the monolithic formulation BLM is in general applicable, however, complex instances are still hard to solve (at least within 3600 seconds). If there are no promising results after 3600 seconds, *i. e.*, there is no significant improvement

compared to the upper bound $|\mathcal{I}|$, the fields will also be marked by a — (*e. g.*, for most of the `Belov` classes and some of the `Foerster & Wäschher` sub-classes).

It's obvious that the last two remarks also depend on the system (hard- and software) and that the evaluation (*i. e.*, ✗ vs. —) and the handling of the results are a matter of consideration in many cases, *e. g.*, we could also enter the worst case number of patterns $|\mathcal{I}|$ for the tenth instance of `bel20_1` in order to calculate the mean pattern count of this class.

Setup Minimization - Real-world Instances

Table 3.8 summarizes the results of the `BLMPG`, `BLM`, `Greedy3` and `DDA` approaches on the instances of the `Kallrath` class. In particular, the number z of different patterns, the number m of master rolls and the solution time in seconds for each instances is reported.

	BLMPG			**BLM**			**Greedy3**			**DDA**		
Instance	z	m	**sec**	z	m	**sec**	z	m	**sec**	z	m	**sec**
C01	**1**	5	**1**	**1**	1	0	1	**1**	0	**1**	1	0
C02	**2**	17	1	**2**	21	0	**2**	17	0	**2**	17	0
C03	**2**	5	1	**2**	4	0	**2**	6	0	**2**	4	0
C04	**4**	690	4	**4**	707	28	**4**	1008	3	7	567	2
C05	**2**	5	1	**2**	5	0	**2**	5	1	**2**	5	0
C06	**4**	26	3	**4**	33	4	**4**	33	1	**4**	35	1
C07	**6**	78	1	**6**	73	2	7	77	2	8	73	1
C09	**4**	31	2	**4**	34	1	5	41	1	5	35	1
C10	**5**	39	1	**5**	32	1	**5**	39	2	6	37	1
C11	**8**	56	1	**8**	56	2	**8**	56	2	**8**	56	1
C12	**6**	39	2	**6**	46	1	7	37	1	**6**	36	3
C13	**4**	66	14	**4**	66	1	**4**	77	1	**4**	66	1
C14	**4**	7	6	**4**	7	1	5	9	1	**4**	7	1
C15	**7**	33	4	**7**	33	5	8	55	2	8	53	4
C16	**8**	20	1	**8**	21	1	**8**	20	2	**8**	20	1
C17	**5**	7	2	**5**	**7**	0	6	10	1	**5**	7	1
C18	**10**	103	8	**10**	124	231	11	128	3	11	115	5
C19	✗			**3**	6	2	**3**	6	1	**3**	6	2
C27	✗			**6**	28	91	7	30	3	7	44	11
C28	**19**	31	1	**19**	32	2	21	34	6	**19**	32	3
C29	**24**	119	1	**24**	119	0	**24**	119	5	**24**	119	2
C32	✗			**9**	22	11	10	23	3	**9**	29	8
C42	✗			**11**	35	34	12	37	4	**11**	35	26
C49	**15**	469	80	**15**	468	30768	18	756	9	18	483	9
C50	**15**	480	449	**15**	477	38565	19	583	9	18	584	11

Table 3.8.: Computational Results: `Kallrath` Class - Setup Minimization

Note that `BLMPG` is not applicable on four instances, as the number of generated patterns is far too large (at least several hundreds of thousands).

At first we are going to compare the exact model formulations `BLMPG` and `BLM`. As can be seen, the model based on complete pattern generation (if applicable) is superior in terms of solution time compared to `BLM`, however, all instances can be solved to optimality based on the `BLM` formulation with $\overline{z} = 7.36$. The first 23 instances of low or medium complexity are solved in short time (at most 231 seconds on `C18`, but in general within ten seconds). Though, the solution time rises significantly to about ten hours when solving the instance `C49` or `C50`, since both are very complex, *i. e.*, the number of orders is greater than 30 and the demand levels are high on average.

It is of great interest to compare the results of the linearised model formulation `BLM`

with the MINLP formulation presented in 2.1.1 in order to evaluate the obtained performance improvement of the novel model formulation. Results for the MINLP formulation based on the Kallrath class are computed in [6] (published in 2015, solver: BARON). While most of the first thirteen instances of low complexity can be solved within a few seconds, too, the solution time increases drastically for the majority of the last twelve instances of medium or high complexity (except for C17, C19 and C29, which are solved in a few seconds as well). In particular, C15 and C16 are solved in about 7000 resp. 10000 seconds, C18 and C42 cannot be solved within 86400 seconds (absolute gap: 5 resp. 1 pattern). The instances C27, C28 and C32 are solved in around 2200, 10000 and 1800 seconds, while no feasible solution was found for C49 and C50 within 86400 seconds.

Even so we have to take in count that there are newer solver versions available today, we can postulate that the BLM formulation is more performant by the factor 10^4 (approximately) in terms of solution time (on instances with medium or high complexity) compared to the MINLP formulation based on those results.

Except for the instances C28, C49 and C50, the Greedy3 algorithm computes optimal or near optimal solutions with an absolute gap smaller or equal two in under ten seconds for all instances, while the DDA procedure provides optimal or near optimal solutions except for the instances C04, C07, C49 and C50 in short solution times as well. The average number of patterns for Greedy3 resp. DDA is equals to 8.12 resp. 8.00, *i. e.*, DDA provides slightly better results on average on this class.

The Exhaustion Method (see [5] for reference, published in 2014) provides results with an average pattern count at around 7.60. We cannot determine an exact value for comparison, as the results differ on three instances slightly, *e. g.*, we found the optimal value of C07 to be 6 (in accordance with [6]) instead of 4 reported in [5]. On the other hand, we found the optimal solution of C19 to be 3 (in accordance with [6]), while the optimal solution is claimed to be equals 4 in [5]. Apart from this marginal differences, it can be said that the Exhaustion Method provides slightly better solutions compared to Greedy3 and DDA on this class (0.5 patterns less on average). In addition, the number of master rolls tends to be lower (see also Table 3.11 for a comparison with Greedy3* and DDA-NR1). On the other hand, note that the solution time is 52 seconds on average compared to two seconds of Greedy3 and four seconds of DDA. Also take in mind that the effort required to implement the Exhaustion Method is greater.

Table 3.9 summarizes the results of the BLMPG, BLM, Greedy3 and DDA approaches on the instances of the Vanderbeck benchmark class. Note that BLMPG is not applicable on eight out of sixteen instances, as the number of generated patterns is far too large.

	BLMPG			BLM			Greedy3			DDA		
Instance	z	m	**sec**	z	m	**sec**	z	m	**sec**	z	m	**sec**
d43p21	✗			**9**	45	123	10	51	3	11	47	21
d16p6	**8**	43	1	**8**	41	7	**8**	48	2	9	58	5
11p4	✗			**4**	189	34634	7	606	12	5	185	5
7p18	**4**	141	22	**4**	107	67	5	429	2	6	149	1
d33p20	✗			**7**	31	54	9	45	2	**7**	31	7
12p19	**4**	26	108	**4**	25	37	5	39	1	5	27	1
14p12	✗			**5**	64	849	6	136	5	6	69	2
30p0	✗			**7**	95	19755	9	337	10	9	105	9
kT01	**3**	15	4	**3**	24	1	**3**	19	1	**3**	21	1
kT02	**15**	71	2	**15**	71	7	17	80	4	17	70	4
kT03	**6**	78	2	**6**	119	3	7	77	1	8	73	1
kT04	**8**	41	3	**8**	51	9	**8**	48	2	9	58	7
kT05	**6**	55	1	**6**	55	1	**6**	85	1	7	55	1
kT06	✗			**3**	67	1440	4	57	5	4	57	3
kT07	✗			**4**	68	631	5	165	4	5	115	3
kT09	✗			**5**	117	2501	6	232	10	**5**	144	3

Table 3.9.: Computational Results: `Vanderbeck` Class - Setup Minimization

All instances which can be formulated according to the `BLMPG` formulation are solved to optimality in short time. Fourteen out of sixteen instances can be solved to optimality in short or acceptable time based on the `BLM` formulation, while the solution times for the instances `11p4` and `30p0` are high (several hours), as the demand levels and/or the number of orders are high on average.

The heuristic approaches provide results in short time (far under 30 seconds in general) with optimal or near optimal solutions (absolute gap smaller or equal two, except for `Greedy3` on instance `11p4`), while `Greedy3` with an average pattern count of 7.19 performs slightly better compared to `DDA` with an average value of 7.25 patterns. However, also note that the number of master rolls to be cut in the solutions computed by `Greedy3` is approximately twice the size on average compared to those computed by `DDA` (secondary objective).

To the best of our knowledge, only F. Vanderbeck (Branch & Cut & Price, see [3]) and G. Belov & G. Scheithauer (Branch & Price, see [8]) provide computational results for this benchmark class. Although both authors stipulate exact demand fulfilment, the comparability is limited, as both approaches minimize the number of different patterns based on the optimal solution m^{opt} of the classical cutoff reduction model, *i. e.*, there is a further constraint to limit the total pattern multiplicity like

$$\sum_{p \in \mathcal{P}} \mu_p \leq m^{opt},$$

where μ_p denotes the multiplicity of pattern p. In contrast to this, we focus on minimizing the number of different patterns primarily. Consequently, our optimal solutions on pure setup minimization with an average value of 6.13 are better in terms of z compared to [3] with an average value of 7.31 and [8] with an average value of 7.50, at the cost of higher values of m on average ($\overline{m} = 73$ compared to $m^{opt} = 58$ „used“ in the Vanderbeck and Belov approach). Note that both authors cannot prove there solutions to be optimal on several instances. However, the strengths of Vanderbeck's and Belov's methods only becomes clear if you take in mind the computational time and the dates of publication. In particular, the average computational time of Vanderbeck resp. Belov is around 437 sec-

onds resp. 142 seconds on far inferior systems in terms of hard- and software (Vanderbeck in 1999, Belov in 2003) compared to the average computational time of BLM with 3758 seconds (note the influence of the two outliers on this average value). On the other hand, the implementation effort of Vanderbecks's or Belov's approach is far higher compared to the comfortable implementation of BLM, Greedy3 and DDA with an algebraic modelling language. We will resume the comparison once our heuristics have been adjusted in order to minimize m as secondary objective (see Table 3.12).

Table 3.10 summarizes the results of the BLMPG, BLM, Greedy3 and DDA approaches on the instances of the Fiber benchmark class. Note that BLMPG is not applicable on 14 out of 39 instances, as the number of generated patterns is far too large.

As can be seen that all instances which can be formulated according to BLMPG are solved to optimality in short or acceptable time (the longest solution time is around 1000 seconds for the instance fiber19_5180). The results of BLM are very convincing on this benchmark class, as there are no outliers compared to the previous benchmark classes. In particular, each instances can be solved to optimality with an average pattern count of 4.51 in under 3600 seconds, which is quite acceptable.

The heuristic approaches Greedy3 and DDA provide results in short time (at most 12 seconds per instance) with optimal or near optimal solutions (absolute gap smaller or equal two), while Greedy3 with an average pattern count of 5.23 performs slightly better compared to DDA with an average of 5.49 patterns. However, as already noticed on the previous benchmark class, take in mind that the number of master rolls to be cut in the solutions computed by Greedy3 is again approximately twice the size on average compared to those computed by DDA (secondary objective).

Computational results on this benchmark class have been published by several authors, among them Umetani Et al. back in 2003, see [7] and Cui Et al. in 2015, see [19]. However, the problem definition investigated in [7] (the number of patterns is fixed to a specific value while the quadratic derivation of the produced item levels from the demands levels is minimized) differs to much from our problem description in order to carry out a meaningful comparison. We found the problem definition by Cui closer to ours, but there are also some major differences. The authors in [19] allow for unlimited overproduction while they observe both, the number of master rolls and the number of different patterns within the objective. In addition, both terms are multiplied with the cost per master roll resp. pattern (weighting factors), *i. e.*, the objective is to minimize the total material and setup cost. The average pattern count reported in [19] of 7.9 is significant higher than the optimal pattern count of 4.51 calculated with BLM for the pure setup minimization problem. Also the heuristics Greedy3 and DDA provide solutions with fewer patterns on average ($\overline{z} = 5.23$ and $\overline{z} = 5.49$ in three resp. two seconds on average). On the other hand, the solutions reported in [19] are much better in terms of m. In particular, 61 master rolls are cut on average compared to 98 by BLM and 222 resp. 131 computed by Greedy3 resp. DDA. Those significant differences are probably best explained by preferring solutions with lower material costs over solutions wit lower setup cost by adjusting the weighting factors in the objective. Note that the heuristics Greedy3* and DDA-NR1 compute solutions much closer to the value $\overline{m} = 61$ with an almost constant pattern count (see discussion in relation to Table 3.13). The average computation time of the results in [19] is 1.5 seconds on a system similar to ours.

	BLMPG			BLM			Greedy3			DDA		
Instance	z	m	**sec**	z	m	**sec**	z	m	**sec**	z	m	**sec**
fiber06_5180	**4**	84	3	**4**	49	4	**4**	89	1	5	44	1
fiber06_9080	**3**	23	14	**3**	21	2	**3**	26	2	4	35	1
fiber07_5180	**3**	34	2	**3**	45	2	**3**	48	1	4	55	0
fiber07_9080	**3**	21	7	**3**	39	3	4	165	1	4	35	1
fiber08_5180	**3**	197	4	**3**	141	5	4	225	1	4	387	1
fiber08_9080	**3**	54	13	**3**	70	4	**3**	205	1	4	382	1
fiber09_5180	**4**	117	2	**4**	80	3	**4**	89	2	5	88	1
fiber09_9080	**4**	39	13	**4**	57	3	**4**	78	1	**4**	83	1
fiber10_5180	**4**	86	4	**4**	109	4	5	175	2	5	245	2
fiber10_9080	**3**	45	31	**3**	42	8	4	177	1	4	145	1
fiber11_5180	**4**	251	10	**4**	155	6	**4**	142	2	5	69	3
fiber11_9080	**3**	55	168	**3**	55	9	4	263	2	4	57	2
fiber13a_5180	**5**	65	7	**5**	65	12	6	132	2	6	77	1
fiber13a_9080	**4**	55	84	**4**	35	13	**4**	43	1	**4**	45	2
fiber13b_5180	**4**	44	29	**4**	47	3	**4**	129	1	**4**	125	2
fiber13b_9080	✗			**3**	116	3	4	119	1	4	124	2
fiber14_5180	**4**	92	19	**4**	92	3	**4**	158	1	5	86	1
fiber14_9080	✗			**3**	29	15	4	109	2	4	76	1
fiber15_5180	**4**	106	5	**4**	105	15	5	306	2	5	112	1
fiber15_9080	**3**	35	53	**3**	102	6	4	296	2	4	204	2
fiber16_5180	**6**	134	466	**6**	97	351	8	250	4	7	121	2
fiber16_9080	✗			**5**	64	362	**5**	103	3	6	67	3
fiber17_5180	**5**	86	48	**5**	88	238	6	220	3	**5**	88	2
fiber17_9080	✗			**4**	57	183	5	201	4	5	66	2
fiber18_5180	**6**	280	100	**6**	332	159	**6**	339	3	**6**	317	2
fiber18_9080	✗			**4**	56	253	5	303	3	6	73	4
fiber19_5180	**6**	272	1086	**6**	469	694	7	205	4	7	491	2
fiber19_9080	✗			**4**	90	293	5	254	4	5	137	2
fiber20_5180	**6**	33	39	**6**	41	21	**6**	67	2	7	37	4
fiber20_9080	✗			**4**	24	13	**4**	24	1	5	23	4
fiber23_5180	**7**	172	90	**7**	165	511	8	503	5	8	162	2
fiber23_9080	✗			**5**	109	1077	7	477	7	6	87	3
fiber26_5180	**7**	323	574	**7**	197	2472	8	957	10	9	236	5
fiber26_9080	✗			**5**	142	1707	6	626	12	6	299	6
fiber28a_5180	✗			**6**	113	517	7	252	5	8	89	5
fiber28a_9080	✗			**5**	56	429	6	219	4	6	53	7
fiber28b_5180	✗			**8**	147	2585	10	557	7	9	142	4
fiber29_5180	✗			**7**	71	208	8	73	4	9	74	3
fiber29_9080	✗			**5**	38	72	6	37	4	6	59	4

Table 3.10.: Computational Results: `Fiber` Class - Setup Minimization

To the best of our knowledge, we can provide optimal solutions with respect to the pure setup minimization problem with exact demand fulfilment for all instances of the benchmark classes `Kallrath`, `Vanderbeck` and `Fiber` for the first time in literature by using the `BLM` formulation. Those optimal solutions are computed in short resp. acceptable time for problem instance of small or medium complexity, while there are some outliers of high complexity, which can only be solved to optimality based on a monolithic model formulation after several hours.

Both heuristics, `Greedy3` and `DDA`, provide solutions with small absolute resp. relative gap in terms of z in short time on all problem instances.

Setup Minimization & Cutoff Reduction - Real-world Instances

Table 3.11 summarizes the results of the BLMPG-NR1, BLM-NR1, Greedy3* and DDA-NR1 approaches on the instances of the Kallrath benchmark class. Note that *e. g.*, the minimal value for m for the instance C4 of 456 reported by DDA-NR1 is not marked as optimal (even so it is the smallest one), as the corresponding number of patterns for this solution is not optimal. The approach BLMPG-NR1 is not applicable on 4 out of 25 instances, as the number of generated patterns is far too large.

Minimal number of master rolls solutions constrained by the minimal number of patterns can be calculated with BLMPG-NR1 on all instances except for C19, C27, C32 and C42. There is a significant increase in terms of solution time, most obvious for the instances C18, 49 and C50 (from seconds resp. minutes to several hours). The average number of master rolls compared to the solutions of pure setup minimization computed by BLMPG is reduced from 111 to 94 (approximately 15 %).

Most of the instances can be solved with BLM-NR1 in reasonable time, except for C4 and C18 the increase of solution time compared to BLM is acceptable. Note that we do not provide solutions for C49 and C50. The estimated solution time per instance is about one day. The number of master rolls being cut is reduced by approximately 24 % compared to BLM based on the first 23 instances.

	BLMPG-NR1			**BLM-NR1**			**Greedy3***			**DDA-NR1**		
Instance	z	m	**sec**	z	m	**sec**	z	m	**sec**	z	m	**sec**
C01	**1**	**1**	1	**1**	**1**	0	**1**	**1**	1	**1**	**1**	1
C02	**2**	**13**	2	**2**	**13**	0	**2**	17	0	**2**	17	1
C03	**2**	**4**	1	**2**	**4**	0	**2**	**4**	0	**2**	**4**	1
C04	**4**	**457**	14	**4**	**457**	5707	**4**	545	3	5	456	4
C05	**2**	**5**	2	**2**	**5**	1	**2**	**5**	0	**2**	**5**	1
C06	**4**	**17**	5	**4**	**17**	7	**4**	18	1	**4**	25	1
C07	**6**	**66**	2	**6**	**66**	10	7	68	2	7	71	1
C09	**4**	**27**	2	**4**	**27**	2	5	28	2	6	32	1
C10	**5**	**27**	2	**5**	**27**	3	**5**	31	1	6	36	1
C11	**8**	**51**	1	**8**	**51**	4	9	60	2	**8**	**51**	2
C12	**6**	**36**	3	**6**	**36**	5	7	36	2	**6**	**36**	3
C13	**4**	**44**	25	**4**	**44**	7	**4**	66	1	6	44	3
C14	**4**	**7**	10	**4**	**7**	3	5	8	2	**4**	**7**	1
C15	**7**	**31**	8	**7**	**31**	10	8	36	3	8	37	8
C16	**8**	**19**	2	**8**	**19**	3	**8**	20	3	**8**	**19**	2
C17	**5**	**7**	4	**5**	**7**	1	6	8	1	**5**	**7**	1
C18	**10**	**98**	215	**10**	**98**	3175	12	114	3	12	111	10
C19	✗			**3**	**6**	2	**3**	**6**	1	**3**	**6**	2
C27	✗			**6**	**27**	121	7	28	2	7	34	13
C28	**19**	**31**	1	**19**	**31**	3	20	33	4	**19**	**31**	4
C29	**24**	**119**	2	**24**	**119**	1	**24**	**119**	5	**24**	**119**	1
C32	✗			**9**	**21**	92	10	23	3	**9**	29	11
C42	✗			**11**	**32**	90	13	38	4	**11**	34	35
C49	**15**	**445**	17185	—			19	459	8	18	449	20
C50	**15**	**460**	4633	—			19	499	8	18	459	15

Table 3.11.: Computational Results: Kallrath Class - Setup & Cutoff Minimization

The heuristics Greedy3* resp. DDA-NR1 still provide optimal or near optimal solutions in terms of z (the average pattern count increased slightly from 8.12 to 8.24 for Greedy3* resp. from 8.00 to 8.04 for DDA-NR1), while the number of master rolls being cut is reduced by approximately 29 % from $\overline{m} = 128$ to $\overline{m} = 91$ on average compared to Greedy3 resp. by approximately 14 % from $\overline{m} = 99$ to $\overline{m} = 85$ on average compared to DDA. The results

are slightly worse than the average number of master rolls being cut when applying the `Exhaustion Method` ($\overline{m} = 83$, see [5]).

Table 3.12 summarizes the results of the `BLMPG-NR1`, `BLM-NR1`, `Greedy3*` and `DDA-NR1` approaches on the instances of the `Vanderbeck` benchmark class. Note that `BLMPG-NR1` is not applicable on 8 out of 16 instances, as the number of generated patterns is far too large. However, minimal number of master rolls solutions constrained by the minimal number of patterns can be calculated for all eight instances. The solution time for `12p19` increased drastically (up to half a day), while the other instances are solved to optimality in acceptable time. The average number of master rolls compared to the solutions of pure setup minimization by using `BLMPG` is reduced from 59 to 50 (approximately 15 %).

Most of the instances can be solved with `BLM-NR1` in reasonable time. Note that we do not provide solutions for `11p4` and `kT09`, as the estimated solution time per instance is about half a day (like for the instance `30p0`). The number of master rolls being cut is reduced by approximately 13 % from $\overline{m} = 62$ to $\overline{m} = 54$ compared to `BLM` based on the instances solved by `BLM-NR1`. The heuristics `Greedy3*` resp. `DDA-NR1` still provide optimal or near optimal solutions in terms of z (the average pattern count increased slightly from 7.19 to 7.38 for `Greedy3*` resp. decreased slightly from 7.25 to 7.19 for `DDA-NR1`), while the number of master rolls being cut is reduced by approximately 54 % from $\overline{m} = 154$ to $\overline{m} = 71$ on average compared to `Greedy3` resp. by approximately 19 % from $\overline{m} = 78$ to $\overline{m} = 63$ on average compared to `DDA`.

By the Table 3.9 we have discussed our results on pure setup minimization compared to the results of Vanderbeck and Belov (see [3] and [8]). Considering the results of `Greedy3*` with $\overline{z} = 7.38$ and $\overline{m} = 71$ computed within four seconds on average resp. of `DDA-NR1` with $\overline{z} = 7.19$ and $\overline{m} = 63$ within seven seconds on average, we have found nearly equivalent solutions compared to the results of Vanderbeck with $\overline{z} = 7.31$ and $\overline{m} = 58$ resp. compared to Belov with $\overline{z} = 7.5$ and $\overline{m} = 58$, but in much shorter time (on a more powerful system) and - of greater importance - with much more simpler approaches. Though, take in mind again that the problem definitions differ as Vanderbeck and Belov search for a minimal pattern solution based on the minimal cuttoff solution.

	BLMPG-NR1			**BLM-NR1**			**Greedy3***			**DDA-NR1**		
Instance	z	m	**sec**	z	m	**sec**	z	m	**sec**	z	m	**sec**
d43p21	✗			**9**	**43**	124	11	45	3	11	43	21
d16p6	**8**	**40**	5	**8**	**40**	13	**8**	**40**	2	9	46	10
11p4	✗			—			5	153	15	5	105	7
7p18	**4**	**95**	632	**4**	**95**	92	5	158	3	6	97	2
d33p20	✗			**7**	**30**	82	8	36	3	**7**	31	8
12p19	**4**	**24**	46254	**4**	**24**	171	6	24	1	5	27	1
14p12	✗			**5**	**56**	377	6	67	6	6	58	3
30p0	✗			**7**	**91**	40201	10	95	10	9	94	12
kT01	**3**	**14**	10	**3**	**14**	2	**3**	16	1	**3**	21	1
kT02	**15**	**71**	3	**15**	**71**	14	18	85	3	17	70	29
kT03	**6**	**66**	2	**6**	**66**	9	7	68	1	7	71	1
kT04	**8**	**40**	5	**8**	**40**	10	**8**	**40**	2	9	46	9
kT05	**6**	**50**	2	**6**	**50**	5	**6**	55	1	7	55	1
kT06	✗			**3**	**67**	1267	5	57	4	4	57	2
kT07	✗			**4**	**67**	160	6	76	3	5	65	3
kT09	✗			—			6	120	9	5	114	5

Table 3.12.: Computational Results: `Vanderbeck` Class - Setup & Cutoff Minimization

Table 3.13 summarizes the results of the BLMPG-NR1, BLM-NR1, Greedy3* and DDA-NR1 approaches on the instances of the Fiber benchmark class. Note that BLMPG-NR1 is not applicable on 14 out of 25 instances, as the number of generated patterns is far too large. We found the solution time of two additional instances to high, about half a day each (estimated). Most of the remaining 23 instances can be solved to optimality in reasonable time, except for fiber19_5180. Based on these instances, the average number of master rolls can be reduced from 96 to 55 by approximately 43 %.

By applying the BLM-NR1 approach, we are able to solve the extended problem to optimality for all instances. However, the solution time increases by a factor of ten for some instances. The average number of master rolls being cut can be reduced from 98 to 63 by approximately 36 %.

	BLMPG-NR1			**BLM-NR1**			**Greedy3***			**DDA-NR1**		
Instance	z	m	**sec**	z	m	**sec**	z	m	**sec**	z	m	**sec**
fiber06_5180	**4**	**34**	5	**4**	**34**	7	**4**	47	1	5	37	1
fiber06_9080	**3**	**19**	63	**3**	**19**	4	4	20	2	4	24	1
fiber07_5180	**3**	**34**	3	**3**	**34**	3	**3**	34	1	4	34	1
fiber07_9080	**3**	**19**	12	**3**	**19**	5	**3**	39	1	4	23	1
fiber08_5180	**3**	**106**	8	**3**	**106**	8	4	137	1	4	385	1
fiber08_9080	**3**	**48**	30	**3**	**48**	7	**3**	48	2	4	52	2
fiber09_5180	**4**	**55**	5	**4**	**55**	10	5	56	2	5	55	1
fiber09_9080	**4**	**29**	34	**4**	**29**	8	5	29	2	**4**	33	1
fiber10_5180	**4**	**70**	10	**4**	**70**	14	5	84	2	5	75	1
fiber10_9080	**3**	**42**	74	**3**	**42**	16	4	49	2	4	43	1
fiber11_5180	**4**	**81**	21	**4**	**81**	21	**4**	81	1	5	69	1
fiber11_9080	**3**	**47**	667	**3**	**47**	13	4	73	2	4	45	1
fiber13a_5180	**5**	**56**	19	**5**	**56**	75	6	67	1	6	56	2
fiber13a_9080	**4**	**32**	747	**4**	**32**	17	6	37	3	**4**	34	1
fiber13b_5180	**4**	**28**	70	**4**	**28**	4	**4**	29	1	**4**	35	1
fiber13b_9080	✗			**3**	**16**	4	4	16	1	4	23	1
fiber14_5180	**4**	**49**	30	**4**	**49**	9	**4**	55	1	5	56	1
fiber14_9080	✗			**3**	**29**	10	4	29	1	4	33	1
fiber15_5180	**4**	**58**	11	**4**	**58**	14	5	59	2	5	76	2
fiber15_9080	**3**	**34**	160	**3**	**34**	12	4	35	2	4	34	2
fiber16_5180	**6**	**84**	2465	**6**	**84**	2603	8	85	3	7	91	2
fiber16_9080	✗			**5**	**47**	515	7	50	3	6	56	3
fiber17_5180	**5**	**84**	157	**5**	**84**	515	6	98	3	**5**	85	2
fiber17_9080	✗			**4**	**47**	61	5	58	4	5	53	2
fiber18_5180	**6**	**98**	2629	**6**	**98**	2325	**6**	104	3	**6**	185	2
fiber18_9080	✗			**4**	**56**	203	5	80	3	6	58	3
fiber19_5180	**6**	**135**	16264	**6**	**135**	929	8	135	4	7	141	2
fiber19_9080	✗			**4**	**90**	209	6	76	5	5	77	2
fiber20_5180	**6**	**33**	98	**6**	**33**	52	**6**	35	2	7	37	5
fiber20_9080	✗			**4**	**19**	14	**4**	24	1	5	19	5
fiber23_5180	—			**7**	**145**	28740	8	159	5	8	161	3
fiber23_9080	✗			**5**	**83**	1341	8	167	6	6	86	4
fiber26_5180	—			**7**	**192**	21835	8	217	9	9	195	6
fiber26_9080	✗			**5**	**107**	2226	6	155	10	6	109	6
fiber28a_5180	✗			**6**	**95**	1694	8	103	4	8	87	8
fiber28a_9080	✗			**5**	**48**	968	6	51	4	6	50	7
fiber28b_5180	✗			**8**	**119**	7718	9	257	7	9	128	7
fiber29_5180	✗			**7**	**62**	245	8	63	4	9	63	3
fiber29_9080	✗			**5**	**35**	815	6	36	6	6	39	4

Table 3.13.: Computational Results: Fiber Class - Setup & Cutoff Minimization

The heuristics `Greedy3*` and `DDA-NR1` still provide optimal or near optimal solutions in terms of z (the average pattern count increased slightly from 5.23 to 5.45 for `Greedy3*` and remains constant at 5.49 for `DDA-NR1`), while the number of master rolls being cut is reduced by approximately 66 % from $\overline{m} = 222$ to $\overline{m} = 76$ on average compared to `Greedy3` resp. by approximately 43 % from $\overline{m} = 131$ to $\overline{m} = 75$ on average compared to `DDA`.

By the Table 3.10 we have discussed our results on pure setup minimization compared to the results of Cui (see [19] for reference). Considering the results of `Greedy3*` with $\overline{z} = 5.45$ and $\overline{m} = 76$ resp. of `DDA-NR1` with $\overline{z} = 5.49$ and $\overline{m} = 75$ compared to the average values of Cui with $\overline{z} = 7.9$ and $\overline{m} = 61$ (which is close to the optimal value of the pure cutoff reduction problem on average), we have found solutions with significant less different patterns on an acceptable cost of 15 additional master rolls to be cut on average. The average computation time is 3 seconds on both heuristics compared to 1.5 seconds of Cui (on a similar system).

Note the following two important remarks, as they will also apply for the evaluation with respect to the Tables 3.14, 3.15 and 3.16 based on all benchmark instances:

1. A different weighting of setup vs. material cost in the approach of Cui (see [19]) may lead to solutions closer to our solutions in terms of z while the number of master rolls being cut increases.

2. As Cui allows for overproduction while we enforce exact demand fulfilment, the number of different patterns and the number of master rolls being cut on an instance reported by Cui can both be better (lower) than even the optimal solution with respect to our problem definition of exact demand fulfilment with pure setup minimization (and cutoff reduction as secondary objective).

Setup Minimization - Computational Results for all Classes
Table 3.14 summarizes the results of the BLMPG, BLM, Greedy3 and DDA approaches on all benchmark classes. The average pattern count $\overline{z}$, the average number $\overline{m}$ of master rolls being cut and the average computation time per instance in seconds are reported.

The results for the Kallrath, Vanderbeck and Fiber classes have been already discussed and compared to results publishes in literature, see the notes in relation with the Tables 3.8, 3.9 and 3.10. Note that the average value $\overline{z}$ is slightly worse for the BLM formulation on the Kallrath and Vanderbeck class as reported before, since the execution is terminated after 3600 seconds, while there are two instances in both classes which are solved to optimality only within several hours.

	BLMPG			**BLM**			**Greedy3**			**DDA**		
Class	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$
Kallrath	—			***7.60***	104	306	8.12	128	2	8.00	99	4
V.beck	—			*6.44*	70	815	7.19	153	4	7.25	79	5
Fiber	—			**4.51**	98	315	5.23	222	3	5.49	131	2
bel20_1	—			—			13.85	507	7	16.80	404	13
bel50_2	✗			—			15.50	374	50	13.20	291	340
bel50_3	✗			—			22.20	696	40	22.50	547	359
bel50_4	✗			—			30.50	1179	29	37.70	958	669
bel50_5	—			—			27.95	1036	19	32.20	813	805
bel50_6	33.10	1256	3327	—			36.35	1541	16	48.20	1242	706
bel50_7	✗			***20.85***	95	868	25.25	116	7	20.95	97	553
bel50_8	✗			—			30.25	1772	38	39.25	1440	726
bel150_9	✗			✗			80.30	3451	259	✗		
type01	—			**3.11**	15	3	3.80	21	1	3.64	17	2
type02	✗			5.60[a]	195	3326	5.48	281	7	5.65	146	4
type03	✗			**4.56**	25	58	5.74	33	2	5.61	30	6
type04	✗			—			8.47	418	28	8.09	252	68
type05	✗			*7.02*	48	2316	9.08	60	5	8.45	53	53
type06	✗			—			13.85	705	114	12.16	478	2282
type07	**6.77**	55	2	**6.77**	57	2	7.37	64	1	8.29	57	1
type08	***7.80***	560	272	—			9.00	736	4	11.25	569	3
type09	**11.72**	102	35	**11.72**	103	71	13.16	118	3	14.98	103	7
type10	—			—			15.95	1317	12	20.08	1041	60
type11	—			*21.15*[a]	199	2602	23.74	221	6	27.33	191	103
type12	—			—			27.82	2289	44	36.55	1884	1886
type13	**8.04**	70	1	**8.04**	72	1	8.46	75	2	9.31	71	1
type14	**8.89**	702	4	—			9.55	809	3	11.74	717	2
type15	**14.31**	130	2	**14.31**	131	41	15.27	144	3	17.54	129	5
type16	***16.06***	1310	226	—			17.88	1534	7	22.67	1328	39
type17	**25.55**	241	4	***26.00***[a]	246	1753	28.06	272	6	32.35	239	93
type18	30.45[a]	2433	3215	—			32.15	2817	22	43.53[a]	2378	1395

Table 3.14.: Computational Results: Summary - Setup Minimization

As can be seen, six classes can be solved to optimality using the BLMPG formulation in very short time on average. In addition, over 90 % of the instances of the classes type08 and type16 can be solved to optimality in reasonable time. The relative gap is mostly under 15 % for the instances of the bel50_6 and type18 class, some of them are solved to optimality using the BLMPG formulation. Most of the instances in class type01 can be solved to optimality within minutes, however the solution time comes to several hours for

some instances due to a huge amount of generated patterns.

Several classes like `Fiber`, `type01` and `type03` can be solved to optimality in short time on average by using the `BLM` formulation, while the relative gap is small on average for the `Vanderbeck`, `bel50_7`, `type05`, `type11` and `type17` classes (at which some instances are solved to optimality). As can be seen, the `BLM` formulations performs bad on most `Belov` classes due to a large amount of order widths and high demand levels and on the `Foerster & Wäscher` sub-classes ending by an even number, as the demand levels are very high, too. The `Greedy3` heuristic is applicable on all instances/classes, providing solutions in very short time on most classes on average. As far as we can compare the results with the optimal or near optimal solutions in the first two columns, the heuristic provides in general solution with small absolute gap (often less than one or two pattern worse except for `bel50_7`) resp. with small relative gap (at around 13 % on average) on all classes. One drawback is the number of master rolls being cut, which is by far the highest on most classes compared to the other approaches.

The `DDA` heuristic is applicable on most classes, although the solution times per instance are significantly higher on several classes compared to `Greedy3`, as we are basically solving the formulation `BLM` twice but on much smaller sub-instances. Note that the solution quality strongly depends on the problem data characteristics. In particular, `DDA` performs very well/best, if the widths ordered are small on average (like for the classes `type01` - `type06`, `bel50_2` and `bel50_7`), while the results get slightly worse on instances with medium sized widths on average like `type07`, `bel50_3` or on the real-world instances in terms of $\overline{z}$. If the widths are broad on average, `DDA` should not be applied. As can be seen, the solutions returned by `DDA` are significant worse compared to `Greedy3`, *e. g.*, seven pattern worse on `bel50_4` and twelve patterns worse on `bel50_6` (while there is nearly no improvement compared to the upper bound of 50 for this instances). In case of `type18`, `DDA` provides solutions with $\overline{z} = 43.53$, *i. e.*, the solution quality is even worse than the upper bound of 40. Note that `DDA` provides preferable solutions with significant lower values of $\overline{m}$ compared to `Greedy3`.

Results for the `Belov` classes have been published in [8] and [9], both by G. Belov and G. Scheithauer. As already noted before, the comparability is limited (although exact demand fulfilment is considered), as Belov and Scheithauer search for a minimal pattern solution based on the optimal solution of the cutoff minimization problem. We obtain the most meaningful comparison, if we concentrate on the best results in terms of $\overline{z}$, either published in [8] or [9], while observing especially those results where the authors allow some increased material input (*i. e.*, 5 % more master rolls may be used compared to m^{opt}), as this leads to better solutions in terms of $\overline{z}$, since the problem definition gets closer to our pure pattern reduction problem with exact demand fulfilment while not observing the number of master rolls being cut. In particular:

- **bel20_1**: The best average solution found in [8] is $\overline{z} = 14.5$ bounded by $\overline{m} = 366$, compared to $\overline{z} = 13.85$ and $\overline{m} = 507$ provided by `Greedy3`.

- **bel50_2**: The best average solution found in [9] is $\overline{z} = 20.65$ bounded by $\overline{m} = 259$, compared to $\overline{z} = 13.20$ and $\overline{m} = 291$ provided by `DDA`.

- **bel50_3**: The best average solution found in [9] is $\overline{z} = 27.60$ bounded by $\overline{m} = 528$, compared to $\overline{z} = 22.20$ and $\overline{m} = 696$ provided by `Greedy3` resp. $\overline{z} = 22.50$ and $\overline{m} = 547$ provided by `DDA`.

- **bel50_4**: The best average solution found in [8] is $\overline{z} = 34.15$ bounded by $\overline{m} = 922$,

compared to $\overline{z} = 30.50$ and $\overline{m} = 1179$ provided by `Greedy3`.

- **bel50_5**: The best average solution found in [9] is $\overline{z} = 34.70$ bounded by $\overline{m} = 745$, compared to $\overline{z} = 27.95$ and $\overline{m} = 1036$ provided by `Greedy3` resp. $\overline{z} = 32.20$ and $\overline{m} = 813$ provided by `DDA`.

- **bel50_6**: The best average solution found in [9] is $\overline{z} = 41.70$ bounded by $\overline{m} = 1149$, compared to $\overline{z} = 36.35$ and $\overline{m} = 1541$ provided by `Greedy3`. The best results in terms of $\overline{z}$ is computed by `BLMPG` with $\overline{z} = 33.10$ and $\overline{m} = 1256$.

- **bel50_7**: The best average solution found in [9] is $\overline{z} = 22.10$ bounded by $\overline{m} = 93$, compared to $\overline{z} = 25.25$ and $\overline{m} = 116$ provided by `Greedy3` resp. $\overline{z} = 20.95$ and $\overline{m} = 97$ provided by `DDA`. The optimal solution in terms of $\overline{z}$ computed with `BLM` is close to $\overline{z} = 20.85$ and $\overline{m} = 95$.

- **bel50_8**: The best average solution found in [9] is $\overline{z} = 35.55$ bounded by $\overline{m} = 1388$, compared to $\overline{z} = 30.25$ and $\overline{m} = 1772$ provided by `Greedy3`.

- **bel150_9**: The best average solution found in [8] is $\overline{z} = 95.30$ bounded by $\overline{m} = 2790$, compared to $\overline{z} = 80.30$ and $\overline{m} = 3451$ provided by `Greedy3`.

The computational time of Belov's and Scheithauer's approach does not exceed 300 seconds, just like `Greedy3` (where the computational time for most instances is less than 60 seconds). In summary, our approaches provide significantly better solutions in terms of $\overline{z}$, at the cost of a higher number of master rolls being cut. However, on some classes like `bel50_2`, `bel50_3` or `bel50_7`, the increase in terms of $\overline{m}$ is within an acceptable range.

Results for the `Foerster & Wäscher` instances (`type01 - type18`) have been published among others in [14], [15], [18], [19] and [20]. As Cui Et. al (see [19], published in 2015) provides the most recent and - as far as we know - best results, we will focus on this paper for comparison basically. In addition, the paper summarizes the results of [14] and [20]. However, as already stated within the remarks by Table 3.13, the comparison suffers from different objectives and problem statements, as the authors allow for overproduction while observing both, the number of master rolls (material cost) and the number of setup (setup cost) within the objective. As a results, the solutions in terms of $\overline{z}$ (and $\overline{m}$) might be better than even the optimal solutions for our problem definition. For instance, Cui reports an average value of $\overline{z} = 6.40$ compared to the optimal solution of our problem definition with exact demand fulfilment of $\overline{z} = 6.77$ for the class `type07` (further: $\overline{z} = 7.52$ on `type13` compared to our optimal solution of $\overline{z} = 8.04$ or Cui on `type14` with $\overline{z} = 8.10$ compared to our optimal solution of $\overline{z} = 8.89$). Also note that our optimal solutions for $\overline{m}$ based on the optimal solutions of the pattern minimization problem are worse compared to the results by *e. g.*, Cui, who reports $\overline{m} = 50.24$ on `type07` compared to our optimal solution of $\overline{m} = 52.25$ (further: $\overline{m} = 63.47$ on `type13` compared to our optimal solution of $\overline{m} = 65.84$).

In order to shorten the comparison for the 18 classes (with 100 instances each), we calculate the mean of means $\overline{\overline{m}}$ resp. $\overline{\overline{z}}$ for `Greedy3` (abbreviation `Gr3`) and compare the results with the mean of means of the approaches provided in literature (see Table 3.15, first row corresponds to $\overline{\overline{m}}$, second row to $\overline{\overline{z}}$). Note that the results of `Greedy3`* (abbreviation `Gr3`*) are added for comparison (see Table 3.16 for a more detailed solution report).

Gr3	Gr3*	YL[14]	LY1[15]	LY2[15]	LY3[15]	CZ[18]	CZ[19]	CZ[20]
661.89	582.39	500.24	510.04	525.74	525.50	–	489.91	490.72
14.16	14.65	15.37	14.84	14.56	14.68	15.82	13.34	15.57

Table 3.15.: Comparison of the Results on the `Foerster & Wäscher` Instances

The initials of the authors in addition to the reference are used to identify the results of the approaches in literature. Note that there are three algorithms provided in [15]. We could not identify the number of master rolls being cut for Cui[18].

As can be seen, `Greedy3` provides the second best mean of means with respect to the number of patterns, while the number of master rolls being cut is by far the worst (highest). At the cost of an increase in terms of $\overline{\overline{z}}$, the number of master rolls can be reduced by applying `Greedy3`*. The average computation time per instance is within seconds for all approaches.

Despite the limited comparability due to the varying problem statements, especially if we consider that Cui provides solutions better than our optimal solutions (see notes above), we found the results of the `Greedy3` heuristic to be remarkable (also take in mind again the simplicity of our approach).

Note that `DDA` also provides better solutions in terms of $\overline{z}$ in general compared to the results in literature based on the classes `type01 - type06`, but with significantly lower values of $\overline{m}$ compared to `Greedy3`, though still higher compared to the results in literature and in case of class `type06` with far higher solution times.

Setup Minimization & Cutoff Reduction - Computational Results for all Classes
Table 3.16 summarizes the results of the `BLMPG-NR1`, `BLM-NR1`, `Greedy3`* and `DDA-NR1` approaches on all benchmark classes. The average pattern count $\overline{z}$, the average number $\overline{m}$ of master rolls being cut and the average computation time per instance in seconds are reported.

The results for the `Kallrath`, `Vanderbeck` and `Fiber` classes have been already discussed and compared to results publishes in literature, see the notes in relation with the Tables 3.11, 3.12 and 3.13. Note that we do not provide the average values for these classes with respect to the exact model formulations `BLMPG-NR1` resp. `BLM-NR1`, as not all instances can be solved within 3600 seconds (but most of them). For the same reason, no results of *e. g.*, `BLMPG-NR1` on `type18` are reported, as the average computation time for determining the minimal pattern solution is already close to 3600 seconds for `BLMPG`. Thus we cannot expect a noticeable reduction in terms of $\overline{m}$ within the short remaining time.

By applying the `BLMPG-NR1` formulation, all instances of four classes can be solved to optimality, *i. e.*, at the cost of higher computation times (but still acceptable), the number of master rolls can be reduced significantly. A great percentage of the classes `type09`, `type14` and `type16` can be solved to optimality, too. The classes `type01` and `type03` can be solved to optimality by using the `BLM-NR1` formulation. However, the amount of instances solvable by a monolithic formulation within 3600 seconds decreases observable.

Only the `Greedy3`* heuristic is applicable on all instances. The average pattern count (mean of means) on the nine `Belov` instances increases from $\overline{\overline{z}} = 31.35$ to $\overline{\overline{z}} = 32.42$ in comparison with `Greedy3`, *i. e.*, by approximately 3.4 %, while the number of master rolls being cut decreases from $\overline{\overline{m}} = 1186$ to $\overline{\overline{m}} = 1102$, *i. e.*, by approximately 7 %. The average

pattern count (mean of means) computed by Belov and Scheithauer (note that we only observed the best results spread over [8] and [9] for comparison) is $\overline{\overline{z}} = 36.25$ bounded by $\overline{\overline{m}} = 916$. The results of `Greedy3*` (*i. e.*, the mean of means) for the 18 `Foerster & Wäscher` classes have already been discussed by table 3.15. While the average number of patterns for these classes increases from $\overline{\overline{z}} = 14.16$ to $\overline{\overline{z}} = 14.65$ using `Greedy3*`, *i. e.*, by approximately 3.5 %, the number of master rolls being cut decreases on average from $\overline{\overline{m}} = 662$ to $\overline{\overline{m}} = 582$, *i. e.*, by approximately 12 %.

Despite from marginal differences, the `DDA-NR1` heuristics provides constant solutions in terms of $\overline{z}$, while the number of master rolls being cut is reduced observable on most classes, at the cost of higher computation times compared to `DDA`. For this reason, `DDA-NR1` provides better results on the classes `type01 - type06`, `Kallrath`, `Vanderbeck`, `bel50_2`, `bel50_3` and `bel50_7` compared to `Greedy3*`, while there are in total fewer results reported for `DDA-NR1` compared to Table 3.14 for `DDA`, as exceeding the time limit of 3600 seconds gets more likely.

	BLMPG-NR1			**BLM-NR1**			**Greedy3***			**DDA-NR1**		
Class	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$
Kallrath	—			—			8.24	91	3	8.04	85	6
V.beck	—			—			7.38	71	4	7.19	63	7
Fiber	—			—			5.45	76	3	5.49	75	3
bel20_1	—			—			14.45	453	6	16.70	373	22
bel50_2	✗			—			16.70	297	55	13.20	268	389
bel50_3	✗			—			23.30	586	36	22.50	513	911
bel50_4	✗			—			30.85	1110	27	—		
bel50_5	—			—			29.35	864	19	—		
bel50_6	—			—			37.25	1429	15	—		
bel50_7	✗						25.70	114	6	20.95	93	1680
bel50_8	✗			—			32.55	1702	35	—		
bel150_9	✗			✗			81.60	3367	227	✗		
type01	—			**3.11**	**12**	4	3.98	15	1	3.64	15	2
type02	✗			—			6.04	143	7	5.65	117	5
type03	✗			**4.56**	**22**	57	6.18	27	2	5.61	28	7
type04	✗			—			9.93	258	29	8.09	232	71
type05	✗			—			9.48	50	5	8.45	48	55
type06	✗			—			16.16	498	170	12.10[a]	432	2521
type07	**6.77**	**52**	3	**6.77**	**52**	5	7.45	59	2	8.26	55	2
type08			—				8.93	593	3	11.17	520	6
type09	**11.72**	***98***	193	**11.72**	**98**	215	13.30	111	3	15.08	100	12
type10	—			—			16.39	1114	11	20.12	977	177
type11	—			—			24.07	214	6	27.40	185	461
type12	—			—			29.10	2152	40	—		
type13	**8.04**	**66**	2	**8.04**	**66**	4	8.43	72	2	9.39	68	2
type14	**8.89**	***654***	66	—			9.58	722	3	11.67	660	4
type15	**14.31**	**124**	3	**14.31**	**124**	181	15.29	138	4	17.54	125	10
type16	**16.45**[a]	*1317*	1600	—			18.13	1379	7	22.71	1245	138
type17	**25.55**	**234**	14	—			28.21	266	6	—		
type18	—			—			33.11	2672	22	—		

Table 3.16.: Computational Results: Summary - Setup & Cutoff Minimization

Comparison of the G3-BLM Approach with BLM and Greedy3

In this section, we are going to compare the results of the exact approach `G3-BLM` (reminder: the solution returned by `Greedy3` is used to warm start the exact formulation `BLM`) to the results of the exact formulation `BLM` (see Table 3.14) in order to evaluate the performance improvement in terms of $\overline{z}$ and solution time by exploiting a good initial solution. In addition, the results of `Greedy3` itself are observed in order to evaluate the additional performance improvement in terms of $\overline{z}$ by solving `BLM` subsequently. The solution process is terminated after 3600 seconds. Table 3.17 summarizes the results of `G3-BLM` and the improvements compared to `BLM` and `Greedy3`. Note that several classes are missing, as the solution times of `BLM` are to high (*i. e.*, there will be no significant improvement of the initial heuristic solution), while we could report the initial solution found by `Greedy3` of course. The absolute deviations of `BLM` and `Greedy3` compared to the results computed by `G3-BLM` are denoted by $\overline{z}'$, $\overline{m}'$ and $\overline{\text{sec}}'$. A negative value indicates an improvement.

	G3-BLM			**BLM**			**Greedy3**		
Class	$\overline{z}$	$\overline{m}$	$\overline{\text{sec}}$	$\overline{z}'$	$\overline{m}'$	$\overline{\text{sec}}'$	$\overline{z}'$	$\overline{m}'$	$\overline{\text{sec}}'$
Kallrath	***7.64***	127	308	+0.04	+23	+2	-0.48	-1	+306
V.beck	*6.44*	94	709	0.00	+21	-18	-0.75	-60	+705
Fiber	**4.51**	100	228	0.00	+2	-87	-0.72	-122	+225
bel50_7	***20.85***	96	379	0.00	+1	-489	-4.40	-20	+372
type01	**3.11**	15	3	0.00	0	0	-0.69	-6	+2
type02	4.90	194	2667	-0.70	-1	-659	-0.58	-87	+2660
type03	**4.56**	26	12	0.00	+1	-46	-1.18	-7	+10
type05	**6.80**	47	230	-0.22	-1	-2086	-2.28	-13	+225
type07	**6.77**	58	3	0.00	+1	+1	-0.60	-6	+2
type09	**11.72**	104	42	0.00	+1	-29	-1.44	-14	+39
type11	***20.35***[a]	188	1437	-0.80	-11	-1165	-3.39	-33	+1431
type13	**8.04**	72	3	0.00	0	+2	-0.42	-3	+1
type14	*8.99*	755	862	–	–	–	-0.56	-54	+859
type15	**14.31**	132	27	0.00	+1	-14	-0.96	-12	+24
type17	**25.3**[a]	239	1088	-0.70	-7	-665	-2.76	-33	+1082

Table 3.17.: Computational Results of `G3-BLM` - Setup Minimization

There is a slightly improvement in terms of $\overline{z}$ compared to `BLM` on some benchmark classes. Certainly, the reduction in terms of runtime compared to `BLM` is more significant and of greater importance, *e. g.*, the average runtime can be reduced by approximately 90 % for instances of the class `type05` while most of the instances of class `type14` can be solved to optimality by applying `G3-BLM` in contrast to `BLM`. However, note that there is no observable impact on some classes like `type01` or `type13` compared to `BLM`. In conclusion, one should apply `G3-BLML` instead of `BLM` to get an optimal solution, as there is a slightly improvement in terms of $\overline{z}$ being computed in much shorter time in general.

At the cost of a significantly increase in terms of solution time compared to `Greedy3`, the number of patterns and master rolls being cut can be reduced clearly on all investigated benchmark classes, *e. g.*, in case of `bel50_7`, the average pattern count is reduced by 4.40.

Comparison of the BLM and BLMFV Formulations

First numerical experiments indicate that the `BLMFV` formulation provides superior results on a (small) subset of instances, *e. g.*, the solution time for `C4` of the `Kallrath` class is reduced from 28 seconds to 5 seconds, for `11p4` of the `Vanderbeck` class from 34634 seconds

to 2733 seconds (enormous reduction) and a significant higher number of instances of the `type02` class can be solved to optimality within 3600 seconds compared to `BLM`. These instances are characterized by high demand levels and small widths for all orders (possible explanation: the large number of variables due to the high demand levels is reduced in the `BLMFV` formulation, while the solver has no troubles in finding feasible solutions due to the small widths). However, further computational experiments must be performed in order to verify this finding. In general, the results are significantly worse in terms of z and solution time compared to the `BLM` formulation.

4. Conclusions and Further Research

The cutting stock problem with setup minimization and exact demand fulfilment has been investigated from a different angle, *i. e.*, instead of designing increasingly sophisticated solution techniques like Branch & Cut & Price involving decompositions methods and applying branching- and cutting strategies, we have exploited progresses in hard- and software technology and problem specific properties in order to develop several performant top-level and easy to implement approaches suitable to either solve the problem to optimality or to provide good heuristic solution.

In particular, novel exact linear model formulations (`BLMPG`, `BLM` and `BLMFV`) have been presented appropriate to solve problem instances of low and medium complexity to optimality in reasonable time. Additionally, we have presented novel exact linear model formulations (`BLMPG-MMR` and `BLM-MMR`) for the setup minimization problem with multiple types of master rolls.

The numerical results on instances for the single type of master roll problem indicate that three classes of real-world problems, consisting of 80 instances in total, can be solved to optimality using the `BLM` formulation within a time limit of 3600 seconds on each instance, except four instances with high complexity for which the solution time rises up to ten hours. Further numerical experiments have been executed on 27 classes frequently used in literature, consisting of 1980 instances in total. For eight classes, we can provide optimal solution for each single instance using one of the model formulations within the time limit of 3600 seconds on each instance. The vast majority of five additional classes can be solved to optimality, too. On three classes, the average computational time rises up to approximately 3600 seconds, while we cannot provide optimal solutions in general, but often with small relative gap. The eleven classes remaining are far to complex to be solvable by a monolithic model formulation.

To the best of our knowledge, we can provide optimal solutions for the pure setup minimization problem with exact demand fulfilment for the first time in literature on the instances mentioned above, except for some problems of the real-world classes, where optimal solutions have been known before.

For more complex instances, we have developed the heuristic approaches referred as `Greedy3` and `DDA`. While the idea of the `Greedy3` heuristics is quite simple, the numerical results, *i. e.*, the number of patterns and the solution time, are very convincing on nearly all benchmark classes. In general, `Greedy3` provides solutions with small absolute gap compared to the optimal solutions we have calculated (one or at most two patterns worse) resp. with small relative gap (approximately 13 % on average), while the approach is applicable on all instances with short solution times as well (at most 259 seconds on extremely large instances, but mostly far under ten seconds per instance). However, `Greedy3` tends to calculate solutions with a high consumption of master rolls being cut.

The solution quality of `DDA` depends much more on the problem instance characteristics. On the subset of instances with small widths on average, the solutions returned by `DDA` are superior compared to `Greedy3`, while the solution quality decreases drastically, if the widths ordered are broad on average. A further disadvantage besides the limited range of usability on some classes is the long average running time on several classes compared

to Greedy3. However, the DDA solutions are throughout preferable in terms of material consumption, *i. e.*, the number of master rolls being cut is lower compared to the Greedy3 solution.

The comparability with our results to those in literature is limited, as most authors investigate a deviating problem definition *e. g.*, by observing both, the number of patterns and master rolls within the objective, while allowing overproduction. Take in mind that the problem becomes easier when allowing for overproduction, *i. e.*, some authors provide solutions even better than our optimal solutions. Nevertheless, as we focus on setup minimization primarily, the results of Greedy3 and DDA (on instances with small widths on average) are in general significantly better than the results in literature with respect to the number of patterns, while the solutions provided in literature are mostly superior with respect to the number of master rolls being cut (which is not considered in the pure setup minimization problem with exact demand fulfilment).

In order to reduce the number of master rolls as secondary objective, we have added some modifications to our approaches for pure setup minimization. By solving the model formulations BLMPG-NR1 resp. BLM-NR1, we are able to calculate optimal solutions for this extended problem on 76 out of 80 real-world instances, leading to a significant reduce of material usage. However, while the solution time is still acceptable (less than one hour) for the majority, a greater amount of instances is solvable only within several hours compared to BLMPG resp. BLM. The estimated computational time rises up to one day for solving each of the remaining four instances. In addition, we can provide optimal solutions for this extended problem on the vast majority of all instances of nine benchmark classes, using either the BLMPG-NR1 or BLM-NR1 formulation, with average solution times far under 3600 seconds. However, the monolithic formulations are inappropriate in general to solve this more challenging problem on classes with higher complexity. Nevertheless, we can provide optimal solutions for the minimal number of master rolls problem constrained by the minimal number of patterns for the first time in literature on a significant amount of instances.

At the cost of solution time, the DDA-NR1 extension is suitable to reduce the number of master rolls being cut compared to DDA, while the number of patterns remains constant in general. However, significant less instances can be solved within 3600 seconds using DDA-NR1 compared to DDA due to the increase in terms of solution time. Also note that the DDA-NR1 approach should only be applied on instances with small widths on average, just like DDA. By applying the Greedy3* heuristic, the number of master rolls being cut can be reduced, while the number of patterns increases moderately. There is no significant difference in terms of solution time compared to Greedy3.

Although the adapted approaches are suitable to reduce the number of master rolls, the results provided in literature are in general clearly superior in terms of material usage, while we still provide preferable solutions in terms of setup cost (taking in mind again the differing problem definitions and - most of all - the simplicity of our approaches).

In conclusion, an instance should be solved based on the BLMPG formulation, if all patterns can be generated in advance. The BLM (or BLMFV) formulation should be used, if the instance is in general of small or medium complexity, but it is not possible to generate all patterns. In case the monolithic formulation should not be applied, use the DDA heuristic if the widths are small on average (and the demand levels are high), otherwise use the Greedy3 heuristic, since this approach provides solutions in very short time with small gap in general.

Further Research
There are several promising points and extensions which could not be (completely) investigated resp. integrated. Among other things, it is worth to tighten the upper and lower bounds on the pattern multiplicities in order to reduce the solution time for models like `BLM`. Also one should perform additional numerical experiments with the `DDA` approach for different values of the divisor v and for the monolithic model formulation `BLMFV`, see Note 8 for some more detailed remarks. A repair heuristic being applied on worse solutions calculated by `DDA` for the sub-instances might be beneficial, as one can easily imagine scenarios in which the number of patterns can be reduced by a simple reorganisation. On the implementation site, it would be worth to add a preprocessing step to the code in order to decide automatically which approach should be used based on the problem instance characteristics.

Additional subjects of much greater scope are the development of top-level heuristics like `Greedy3` but for the one-dimensional cutting stock problem with pure setup minimization and exact demand fulfilment in case of multiple types of master rolls and an approach based on the `Greedy3` heuristic, but with major modifications in order to turn the heuristic into an exact approach (already generated patterns may change completely). With regard to the last point, we suggest to observe only a subset of all order widths in each iteration (which is determined by solving an auxiliary problem) in order to shorten the solution time in each iteration by reducing the number of variables and by applying tighter bounds.

List of Figures

List of Tables

Acronyms

BLM Binary Linear Model. 12

BLM-MMR Binary Linear Model with Multiple Types of Master Rolls. 19

BLM-NR1 Solving `BMP` initially while reducing the Number of Master Rolls being cut subsequently. 41

BLM-NR2 Binary Linear Model based on `BLM` observing both, the Number of Patterns and the Number of Master Rolls being cut (whihin the Objective). 42

BLMEPG Combining the Generation of efficient Patterns with free Patterns, formulated as Binary Linear Model. 21

BLMFV Binary Linear Model with Fewer Variables. 39

BLMPG Binary Linear Model based on complete Pattern Generation. 11

BLMPG-MMR Binary Linear Model based on complete Pattern Generation and Multiple Types of Master Rolls. 18

BLMPG-NR1 Solving `BMPLG` initially while reducing the Number of Master Rolls being cut subsequently. 41

BLMPG-NR2 Binary Linear Model based on `BLMPG` observing both, the Number of Patterns and the Number of Master Rolls being cut (whihin the Objective). 42

DDA Demand Dividing Algorithm. 33

DDA-NR1 Adaption of the `DDA` Algorithm in order to reduce the Number of Master Rolls being cut. 43

G1A Binary Linear Model being solved repeatedly by Algorithm `Greedy1`. 24

G2A Binary Linear Model being solved repeatedly by Algorithm `Greedy2`. 27

GAMS General Algebraic Modeling System. 53

Greedy3* Adaption of the `Greedy3` Algorithm in order to reduce the Number of Master Rolls being cut. 42

MILP Mixed Integer Non Linear Programming. 2

MINLP Mixed Integer Non Linear Programming. 2

PG Model for Pattern Generation (solved with `CPLEX Solution Pool`). 10

Bibliography

[1] Gilmore, P. and Gomory, R.: *A Linear Programming Approach to the Cutting Stock Problem*, Operations Research 9, 849-859, 1961.

[2] Belov, G. and Scheithauer, G.: *A branch-and-cut-and-price algorithm for onedimensional stock cutting and two-dimensional two-stage cutting*, European Journal of Operational Research, 171:85-106, 2006.

[3] Vanderbeck, F.: *Exact Algorithm for Minimising the Number of Setups in the one dimensional Cutting Stock Problem*, Operations Research, 48(5):915-926, 2000.

[4] Johnston, R. and Sadinlija, E.: *A New Model for Complete Solutions to One-Dimensional Cutting Stock Problems*, European Journal of Operational Research 153:176?183, 2004.

[5] Kallrath, J., Rebennack, S., Kallrath, J. und Kusche, R.: *Solving Real-World Cutting Stock-Problems*, European Journal of Operational Research, 238:374-389, 2014.

[6] Banbal, E. and Kallrath, J.: *Exact Optimization of real-world cutting stock problems with GAMS*, Advances in Decision Technology and Intelligent Information Systems, Vol. XVI (Ed. K. J. Engemann, G. E. Lasker): IIAS, pp. 21-25, 2015.

[7] S. Umetani, M. Yagiura and T. Ibaraki: *One dimensional cutting stock problem to minimize the number of different patterns*, European Journal of Operational Research, 146, 388-402, 2003.

[8] Belov, G., Scheithauer, G.: *The number of setups (different patterns) in one-dimensional stock cutting (Technical Report)*, Institute of Numerical Mathematics, Technische Universität Dresden, 2003.

[9] Belov, G., Scheithauer, G.: *Setup and open-stacks minimization in one-dimensional stock cutting*, INFORMS Journal on Computing, 19, 27–35, 2007.

[10] T. Gau and G. Wäscher: *CUTGEN1: A Problem Generator for the Standard One-dimensional Cutting Stock Problem*, European Journal of Operational Research, 84, 572-579, 1995.

[11] S. Umetani, M. Yagiura and T. Ibaraki: *One-dimensional cutting stock problem with a given number of setups: A hybrid approach of metaheuristics and linear programming*, Journal of Mathematical Modelling and Algorithms, 5, 43-64, 2006.

[12] McDiarmid, C.: *Pattern Minimisation in Cutting Stock Problems.* Discrete Applied Mathematics 98, 121–130, 1999.

[13] Haessler, R. W., Sweeney, P. E.: *Cutting Stock Problems and Solution Procedures*, European Journal of Operational, Research 54, 141–150, 1991.

[14] Yanasse, H. H., Limeira, M. S.: *A hybrid heuristic to reduce the number of different patterns in cutting stock problems*, Computers & Operations Research, 33: 2744–2756, 2006.

[15] Lima Cerqueira, G. R., Yanasse, H. H.: *A pattern reduction procedure in a one-dimensional cutting stock problem by grouping items according to their demands*, Journal of Computational Interdisciplinary Sciences 1(2): 159-164, 2009.

[16] Kallrath, J.: *Polylithic modeling and solution approaches using algebraic modeling systems*, Optimization Letters. 5. 453-466. 10.1007/s11590-011-0320-4, 2011.

[17] Adjiman, C. S. J.: *Global Optimization Techniques for Process Systems Engineering*, PhD Dissertation, Dept. of Chemical Engineering, Princeton University, Princeton, NJ, 1999.

[18] Cui, Y., Zhao, X., Yang, Y., Yu, P.: *A heuristic for the one dimensional cutting stock problem with pattern reduction*, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 222, 677–685, 2008.

[19] Cui, Y., Zhong, C., Yao, Y.: *Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost*, European Journal of Operational Research 243: 540–546, 2015

[20] Cui, Y., Liu, Z.: *C-Sets-based sequential heuristic procedure for the one-dimensional cutting stock problem with pattern reduction*, Optimization Methods and Software, 26, 55–167, 2011.

[21] Foerster, H., Wäscher, G.: *Pattern reduction in one-dimensional cutting stock problems*, International Journal of Production Research, 38: 1657–1676, 2000.

[22] Verband Deutscher Papierfabriken: *Papier Kompass*, `https://www.vdp-online.de/test-and-review/internal-demo/pages/showroom-advance/news/read/article/papier-kompass.html`.

A. Notation

Appendix A summarizes the notation and their definition of all symbols being used. In particular, the general notation - as for the basic problem input data (*e. g.*, width W of the master rolls, number K of knifes) - is presented in Section A.1, the sets and the indices being used are summarized in Section A.2. The syntax being used for the parameters and decision variables in the model formulations and heuristic approaches can be seen in Section A.3 and Section A.4, resp.

A.1. General

$W \in \mathbb{R}^+$	Width of the master rolls.
$K \in \mathbb{N}$	Number of knifes.
$z \in \mathbb{N}$	Number of different setups/patterns.
$m \in \mathbb{N}$	Number of master rolls being cut.
$m^{lo} \in \mathbb{N}$	Lower bound on the number m of master rolls being cut.
$m^{opt} \in \mathbb{N}$	Optimal (minimal) number of master rolls being cut.
$z^{lo} \in \mathbb{N}$	Lower bound on the number z of patterns.
$z^{up} \in \mathbb{N}$	Upper bound on the number z of patterns.
$z^{opt} \in \mathbb{N}$	Optimal (minimal) number of patterns.
$v \in \mathbb{N}$	Divisor used in the `DDA`, `DDA-NR1` and `BLMFV` approaches.
$e \in (0; 1]$	Efficiency rate.
$I^P \in \mathbb{N}$	Number of order widths which can be fulfilled exactly by the current pattern.
$\overline{z} \in \mathbb{R}^+$	Average pattern count of a specific class (arithmetic mean).
$\overline{m} \in \mathbb{R}^+$	Average number of master rolls being cut of a specific class (arithmetic mean).
$\overline{\overline{z}} \in \mathbb{R}^+$	Mean of means for the pattern count.
$\overline{\overline{m}} \in \mathbb{R}^+$	Mean of means for the number of master rolls.

A.2. Sets and Indices

$i \in \mathcal{I}$	Set of orders: $\mathcal{I} = \{i_1,\ \ldots,\ i_{\lvert\mathcal{I}\rvert}\}$.
$i \in \mathcal{I}'$	Set of orders which are not fulfilled exactly yet: $\mathcal{I}' \subseteq \mathcal{I}$.
$p \in \mathcal{P}$	Set of patterns: $\mathcal{P} = \{p_1,\ \ldots,\ p_{\lvert\mathcal{P}\rvert}\}$. Sometimes we also use the index $s \in \mathcal{P}$.
$j \in \mathcal{J}$	Set of different types of master rolls: $\mathcal{J} = \{j_1,\ \ldots,\ j_{\lvert\mathcal{J}\rvert}\}$.
$k \in \mathcal{K}$	Set of pattern multiplicities: $\mathcal{K} = \{1,\ \ldots,\ \max_{i\in\mathcal{I}} D_i\}$.
$n \in \mathcal{N}$	Set of item multiplicities: $\mathcal{N} = \{1,\ \ldots,\ \min[\max_{i\in\mathcal{I}} \lfloor W/w_i \rfloor ; K]\}$.

$p \in \mathcal{P}^g$	Set of generated patterns (only efficient patterns): $\mathcal{P}^g = \{p_1, \ \ldots, \ p_{\lvert\mathcal{P}^g\rvert}\}$
$p \in \mathcal{P}^f$	Set of free patterns: $\mathcal{P}^f = \{p_1, \ \ldots, \ p_{\lvert\mathcal{P}^f\rvert}\}$
$k \in \mathcal{K}_0^f$	Set of pattern multiplicities for the quotients: $\mathcal{K}_0^f := \{0, 1, ..., \max_{i \in \mathcal{I}^f} f_i\}$.
$k \in \mathcal{K}_0^r$	Set of pattern multiplicities for the remainders: $\mathcal{K}_0^r := \{0, 1, ..., \max_{i \in \mathcal{I}^r} r_i\}$.

A.3. Parameter

$w_i \in \mathbb{R}^+$	Width of order i.
$D_i \in \mathbb{N}$	Demand level of order i.
$W_j \in \mathbb{R}^+$	Width of master roll type j.
$l_s \in \mathbb{Z}$	Lower bound on the multiplicity of pattern s.
$u_p \in \mathbb{N}_0$	Multiplicity of pattern p.
$a_{ip} \in \mathbb{N}_0$	Item multiplicity of order i in pattern p.
$C_j \in \mathbb{N}_0$	Capacity (stock level) of master roll type j.
$e_{pj} \in \{0, 1\}$	Indicates whether pattern p can be applied on master roll type j.
$I_p^{max} \in \mathbb{N}$	Maximum number of different widths in pattern p.
$P_i^{max} \in \mathbb{N}$	Maximum number of patterns containing order width i.
$\gamma_{ip}^g \in \mathbb{N}_0$	Item multiplicity of order i in pattern p (generated).
$W^r \in \mathbb{R}_0^+$	Remaining length of the recently generated pattern.
$K^r \in \mathbb{N}_0$	Remaining number of knifes of the recently generated pattern.
$W_p^r \in \mathbb{R}_0^+$	Remaining length of pattern p.
$K_p^r \in \mathbb{N}_0$	Remaining number of knifes of pattern p.
$f_i \in \mathbb{N}_0$	Quotients when using the $D_i = f_i v + r_i$ demand representation.
$r_i \in \{0, \ldots, v-1\}$	Remainders when using the $D_i = f_i v + r_i$ demand representation.

A.4. Decision Variables

$\mu_p \in \mathbb{Z}_0^+$	Pattern multiplicity of p.
$\alpha_{ip} \in \mathbb{Z}_0^+$	Item multiplicity of order i in p.
$\delta_p \in \{0, 1\}$	Indicates whether p is used.
$\gamma_i \in \mathbb{Z}_0^+$	Item multiplicity of order i in the currently generated pattern.
$\delta_{kp} \in \{0, 1\}$	Indicates whether p is used exactly k times.
$\mu_{pjk} \in \{0, 1\}$	Indicates whether p is applied exactly k times on j.
$\gamma_{iknp} \in \{0, 1\}$	Indicates whether p contains i exactly n times and is used exactly k times.
$\upsilon_{pj} \in \{0, 1\}$	Indicates whether p can be applied on j.
$\eta_p \in \{0, 1\}$	Indicates whether p is used.
$\chi_i \in \{0, 1\}$	Indicates whether the order i is satisfied exactly.

$\gamma_{inp} \in \{0,1\}$	Indicates whether the order i is added with item multiplicity n to p.
$\delta^f_{kp} \in \{0,1\}$	Indicates whether p is used exactly kv times.
$\gamma^f_{iknp} \in \{0,1\}$	Indicates whether p contains i exactly n times and is used exactly kv times.
$\delta^r_{kp} \in \{0,1\}$	Indicates whether p is used exactly k times.
$\gamma^r_{iknp} \in \{0,1\}$	Indicates whether p contains i exactly n times.

B. Selected Optimal Patterns

Appendix B contains four optimal solutions on the setup minimization problem with a single type of master roll (for demonstration purposes only). The solution method is reported next to the instance name. Note that we have already explained how to interpret the solution reports appropriately within Section 3.2. However, the solution report for the fifth instance (`C12MMR`) is explained in place, as the instance refers to the setup minimization problem with multiple types of master rolls (being solved by `BLM-MMR`).

Instance C4 solved by BLM-NR1:

Solution summary	Total	1	2	3	4	
delta (p used):	4	1	1	1	1	
Number Rolls :	457	211	188	39	19	
Widhts	Width	aip				Out(i) D(i)
A1	36	2	1	0	0	610 610
A2	31	0	2	0	1	395 395
A3	14	1	0	0	0	211 211
A4	45	0	0	2	1	97 97
Max oo Cuts:		3	3	2	2	
Used:	41524	86	98	90	76	
Offcut:	4176	14	2	10	24	
Total:	45700	100	100	100	100	

Instance 7p18 solved by BLMPG:

Solution summary	Total	808	1434	1790	1921	
delta (p used):	4	1	1	1	1	
Number Rolls :	141	1	24	112	4	
Widhts	Width	aip				Out(i) D(i)
A1	542	1	0	3	0	337 337
A2	720	0	1	1	0	136 136
A3	494	0	2	0	2	56 56
A4	593	2	1	0	2	34 34
A5	710	0	1	0	1	28 28
A6	718	1	1	0	0	25 25
A7	536	1	0	0	1	5 5
Max oo Cuts:		5	6	4	6	
Used:	368910	2982	3729	2346	3420	
Offcut:	208626	1114	367	1750	676	
Total:	577536	4096	4096	4096	4096	

Instance Fiber17_5180 solved by BLMPG:

Solution summary	Total	143	917	4487	6609	6976		
delta (p used):	5	1	1	1	1	1		
Number Rolls :	84	10	20	4	44	6		
Widhts	Width	aip					Out(i)	D(i)
A1	500	2	0	1	4	1	206	206
A2	1000	0	1	0	3	0	152	152
A3	1040	2	1	0	0	0	40	40
A4	915	0	1	1	0	1	30	30
A5	950	0	1	0	0	0	20	20
A6	1200	0	1	0	0	0	20	20
A7	900	0	0	0	0	3	18	18
A8	930	0	0	3	0	0	12	12
A9	1020	1	0	0	0	0	10	10
A10	1050	1	0	0	0	0	10	10
A11	600	0	0	0	0	1	6	6
A12	923	0	0	1	0	0	4	4
Max oo Cuts:		6	5	6	7	6		
Used:	422402	5150	5105	5128	5000	4715		
Offcut:	12718	30	75	52	180	465		
Total:	435120	5180	5180	5180	5180	5180		

Instance type07_4 solved by BLMPG:

Solution summary	Total	1	5	111	172	1065	1274		
delta (p used):	6	1	1	1	1	1	1		
Number Rolls :	48	20	10	8	1	3	6		
Widhts	Width	aip						Out(i)	D(i)
A1	738	1	0	0	0	0	0	20	20
A2	129	0	1	1	0	0	0	18	18
A3	24	0	0	0	0	0	3	18	18
A4	704	0	1	0	0	0	0	10	10
A5	54	0	0	0	0	1	1	9	9
A6	792	0	0	1	0	0	0	8	8
A7	557	0	0	0	0	0	1	6	6
A8	298	0	0	0	0	0	1	6	6
A9	675	0	0	0	0	1	0	3	3
A10	394	0	0	0	2	0	0	2	2
Max oo Cuts:		1	2	2	2	2	6		
Used:	39319	738	833	921	788	729	981		
Offcut:	8681	262	167	79	212	271	19		
Total:	48000	1000	1000	1000	1000	1000	1000		

Instance C12MMR solved by BLM-MMR:
The notation `C12MMR` indicates that this instance corresponds to the instance `C12` of the `Kallrath` benchmark set, but that we have just added two types of master rolls with width $W_1 = 2835$ resp. $W_3 = 5265$ to the already existing type with width $W_2 = 4050$ (*i. e.*, $|\mathcal{J}| = 3$) and some random stock levels C_j for testing purposes on our models `BLMPG-MMR` resp. `BLM-MMR` only.

```
1  Solution Summary    Total
2  ---------------------------------------------------------------------
3            p used:       5      1      1      1      1      1  Out(j)  C(j)
4  ---------------------------------------------------------------------
5                MR1    2835      0      0      2      0      0       2    25
6                MR2    4050      0      0      4      5      0       9    35
7                MR3    5265      7      6      0      0      4      17    20
8  ---------------------------------------------------------------------
9           #p used:              7      6      6      5      4
10 ---------------------------------------------------------------------
11 Widhts      Width      aip                                        Out(i)  D(i)
12 A1           1620                2      0      0      0      1      18    18
13 A2           1700                1      0      0      0      0       7     7
14 A3            800                0      0      1      0      0       6     6
15 A4            850                0      1      0      0      0       6     6
16 A5            900                0      0      0      1      0       5     5
17 A6           1050                0      0      0      1      0       5     5
18 A7           1100                0      0      0      1      0       5     5
19 A8           1150                0      0      0      0      1       4     4
20 A9           1720                0      1      0      0      1      10    10
21 A10          1800                0      0      1      0      0       6     6
22 A11          1860                0      1      0      0      0       6     6
23 ---------------------------------------------------------------------
24      Max  3 cuts:                3      3      2      3      3
25               Used:           4940   4430   2600   3050   4490
26
27               MR1    2835                    X
28               MR2    4050                    X      X
29               MR3    5265      X      X      X      X      X
30
31    Total  cutoff:   21655
```

The minimal number of patterns is equal to five. Note that the optimal solution of the single type of master roll problem is one pattern worse. Lines 5-7 indicate how often and on which master roll type the patterns are applied. In addition, the total consumption and the available stock levels for each master roll type is reported. Line 9 summarizes the total pattern multiplicity, *e. g.*, p_3 will be used 6 times in total. The large `X` (added for control purposes, see lines 27-29) indicates, that the pattern can be applied on this master roll type, *e. g.*, the pattern p_3 can be applied on all master roll types, while p_1 can be applied on the widest type only.