ON A NON-DETERMINISTIC MIXED INTEGER PROBLEM FOR PRODUCTION CONTROL

A BRANCH-AND-BOUND ALGORITHM AND A NEURAL NETWORK APPROACH

Inauguraldissertation
zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Katrin Schlegel, M. Sc.
aus Tübingen

Mannheim, Juni 2021

Dekan: Dr. Bernd Lübcke, Universität Mannheim

Referent: Professor Dr. Simone Göttlich, Universität Mannheim

Korreferent: Professor Dr. Michael Herty, Universität Aachen

Tag der mündlichen Prüfung: 13. Oktober 2021

Abstract

Just in sequence (JIS) production of automotive manufacturers and numerous of their suppliers shift the focus to production sections that do not preserve the order sequence during the production process. The increasing variety enhances the importance of such sections. Resorting is necessary, which is cost, time and space consuming. The aim to satisfy the customers' demand, meet goals for productivity growth and be competitive through increase in efficiency motivates the controlling of such production sections.

This work focuses on use cases with a non-deterministic perturbation of the order sequence. The mathematical models are built and tested with the data provided by the paint shop of an automobile manufacturer. A mixed integer program and a neural network approach are presented.

The MIP is formulated based on a simulation of the paint shop in the objective function and an approximation of the statistics in the constraints. The structure of the model is exploited and dependencies are pointed out. It is shown to be stable, further experiment designs for use cases are proposed.
A specific constraint is identified as the main factor for the long and strongly varying computational times. On that basis a customized branch-and-bound algorithm is developed. The success of the algorithm in the reduction of the search tree size is shown and proposals for a performant implemetation are made.

Within the scope of a second approach a sequence-to-sequence neural network is presented. This approach is motivated by the strength of neural networks to recognize patterns in data without any preceeding interpretation or assumptions by the researcher. Different network types are discussed and the sequence-to-sequence network is embedded. Experiments with different parameter settings point out the non-deterministic and complex data structure. Further theoretical considerations on the network design are discussed.

Zusammenfassung

Die just-in-sequence-Produktion bei Automobilherstellern und einer Vielzahl ihrer Zulieferer lässt Produktionsabschnitte in den Fokus der Produktionssteuerung rücken, die die Reihenfolge der Aufträge nicht erhalten. Die wachsende Produktvielfalt verstärkt die Gewichtung ebensolcher Abschnitte. Die daraus resultierende notwendige Resortierung hat einen hohen finanziellen, zeitlichen und räumlichen Aufwand zur Folge. Das Bestreben die Kundennachfrage zu befriedigen, Wachstumsziele zu erreichen und konkurrenzfähig zu bleiben motiviert die Steuerung und Optimierung solcher Produktionsabschnitte.

Der Fokus dieser Arbeit liegt auf nicht-deterministischer Verwirbelung der Auftragsreihenfolge. Anhand der Daten aus dem Gewerk Oberfläche eines Automobilherstellers als Anwendungsfall werden mathematische Modelle zur Steuerung formuliert und getestet. In zwei seperaten Ansätzen wird das Potenzial eines MIP und eines neuronalen Netzes untersucht.

Das MIP basiert zum einen auf einer Simulation der Oberfläche in der Zielfunktion und zum anderen auf einer Annäherung der statistischen Verwirbelung in den Nebenbedingungen. Die Struktur des Models wird untersucht und es wird auf Abhängigkeiten zwischen Variablen hingewiesen. Wir zeigen, dass das Modell stabil ist und geben einen Ausblick auf Möglichkeiten zur Umsetzung weiterer Experimente.
Insbesondere eine Nebenbedingung wird als Hauptfaktor für lange und stark schwankende Rechenzeiten des MIPs identifiziert. Ausgegend davon wird ein branch-and-bound Verfahren mit problemspezifischem Pruning entwickelt. Die aus der Anwendung dieses Algorithmus resultierende Verkleinerung des Suchbaums wird evaluiert und es werden Vorschläge für eine performante Implementierung gemacht.

Im Rahmen eines zweiten Lösungsansatzes wird ein sequence-to-sequence neural network vorgestellt. Dieser Ansatz ist motiviert durch das Vermögen neuronaler Netze, losgelöst von menschlichen Interpretationen oder Annahmen, Muster in Daten erkennen zu können.
Unterschiedliche Arten von Netzen werden diskutiert und das sequence-to-sequence

neural networt wird eingebettet. Experimente mit unterschiedlichen Parametereinstellungen zeigen erneut die nicht-deterministische und äußerst komplexe Struktur der Daten auf. Weitere theoretische Überlegungen zur Architektur des neuronalen Netzes werden diskutiert.

Contents

List of Figures

List of Tables

1. Introduction

Any customer can have a car painted any color that he wants so long as it is black.

(Henry Ford)

Whereas the number of available colors for cars was limited to a handful of choices in the times of Henry Ford, one hundred years later automobile manufacturers provide a large variety to meet customer demands. The growing number of colors made it necessary to recognize the painting process in the context of production management. Today the painting process is a complex, multistage, extremely energy intensive and expensive operation [Bys+20]. While researchers have focused on optimizing processes in the assembly line so far, the paint shop has stayed untouched to a large extent. One reason might be, that high potential in cost and material reduction is facing a high process complexity.

In contrast to the final assembly, where the sequence of bodies gets physically fixed with putting them on the assembly line, a sequence of bodies gets perturbed while passing the paint shop. Currently, this perturbation is indirectly dealt with by operating a high rack warehouse for painted car bodies located between the paint shop and the assembly line. This so-called main buffer works as a storage and sorting place. Construction and operation of the buffer consumes space, that is often limited within a plant, as well as human resources. Additionally, it produces costs in a significant amount.
This gives rise to the question asking for a method to control the paint shop such that the buffer size can be reduced maintaining the supply of the assembly line.
The assembly line defines the sequence, in which the painted car bodies leave the paint

shop ideally. Hence, we look upon this shop against the production direction. Let a desired output sequence be given by the assembly line. Then, the method should determine an input sequence, whose resulting output sequence after passing the paint shop differs as little as possible from the given output sequence. A paint shop output sequence similar to the sequence demanded by the assembly line usually requires less buffer spaces than an arbitrary sequence.

Approaching the challenge of controlling the paint shop we use techniques from two different mathematical fields. On the one hand, we model the behaviour of the paint shop and formulate a mixed integer program. Optimization theory is known as a strong tool and its application in production planning and production control has rapidly increased in the past few decades.
On the other hand, we feed a neural network with the raw data. A powerful benefit of neural networks is the capability to identify patterns or trends in data, that have remained unrecognized by the human eye and data analysis. Recently, the potential of neural networks as an operations research tool has been discussed.

Structure of this work

Chapter 2 is dedicated to the structure of mixed integer programs as well as the performance of well-established algorithms. We give an introduction to the properties of polytopes. The performance of well-known optimization algorithms on polytopes is presented and reformulation techniques are outlined.
Chapter 3 starts with the description of the use case, originally provided by a member of the automotive industries, that we raise to a more general level. The problem presented stands out by its complexity and strong dependency on the input data. The terms of an extensive analysis of the available data from the Daimler AG are described and the results are presented. These results show, that certain adjustments in the operating principle of the paint shop are necessary in order to have starting points for the controlling thereof. We set a framework of assumptions and restrictions and develop an algorithm for the generation of artificial data fitting into this setting. We claim, that this algorithm is “the most deterministic way” of generating statistical data meeting the required specifications, which form the basis for best-case scenario analyses of the presented problem. Within the same framework we formulate a mixed integer program. Find the study of stability and the results of extensive test calculations presented

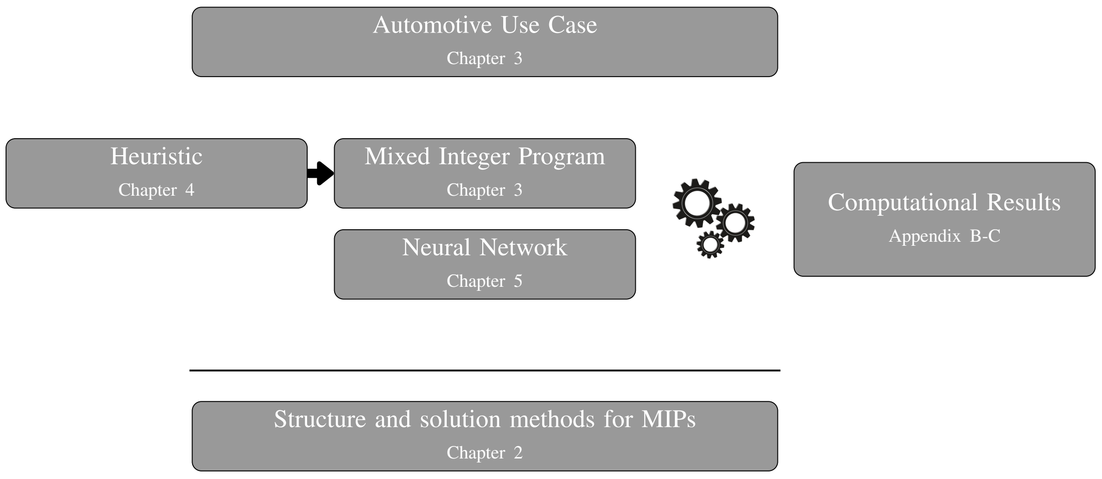


Figure 1.1.: Chapter overview

in Chapter 3. We conclude the chapter with a discussion of application possibilities in the automotive section and beyond.

In Chapter 4, a heuristic is presented in order to handle the strongly varying and partially extremely long computational times of the mixed integer program in Chapter 3. The chapter gives an overview of well-known branching and search strategies as well as pruning rules as being the core components of a branch-and-bound or branch-and-cut algorithms. The impact of the components on each other is roughly summarized. Subsequently, a branch-and-bound algorithm with a custom pruning rule is introduced and tested. We combine depth first search with a problem-specific pruning rule, that allows to reduce the search space by a significant part. The results are presented and further modifications and analyses are discussed.

Chapter 5 starts with an introduction to neural networks and their increasing popularity for numerous use cases in the industrial sector. We outline the differences of common neural networks and their application areas. The main design choices for networks are described. The remainder of this chapter is concerned with the architecture of a sequence-to-sequence neural network. Mapping one sequence into another does not belong to the inner circle of classic examples for the scope of neural networks. Classification and regression problems are application examples that come to mind first. In designing a reliable network we follow the architecture of translation networks with encoder and decoder. The network is tested with color sequences. Based on these re-

sults we present theoretical considerations to modify the network and the loss function, in particular.
See Figure 1.1 for a graphical chapter overview.

Application spectrum

The phenomenon described in this work can be found in numerous productive sectors. Due to our research background, the first industries suggesting themselves are automotive suppliers that provide their goods *just in sequence* (JIS).
For the performance of a mixed integer program as well as a neural network the data quality is essential. The application requires an extensive analysis of the relevant manufacturing data and -potentially- of production processes and structural conditions as well. There might be cases, where raw data is usable directly or after eliminating "non-representative" data caused by unregularities such as emergency stops. Yet we assume that it is not uncommon that the manufacturing process might require adjustments concerning noise making factors, such as wear and tear of machining tools, temperature of the material, air humidity, human factors of any kind, et cetera. This procedure of adjustment also has to be executed on the paint shops of the Daimler AG in order to apply the mixed integer program. Mainly depending on the exact structure and the number of processed Rohbaulack variants more or less groundwork is necessary in the paint shops of the individual plants to reach a certain quality of results. Similar efforts have been sought for the body shop in the plant in Sindelfingen since a few years within the scope of a separately set up project. This sumptuous analysis takes the cooperation of production managers, data analysts and mathematicians. By the time of completion of this work final results of the body shop project have not been available yet.
The complex structure of the paint shop is both, downside and opportunity. Despite all groundwork it might be necessary to combine several models in series in order to cope with the complexity. We see potential in dividing the paint shop in several sections, for each of which a modification of the presented mixed integer program or an individually trained neural network applies. However, the Daimler AG currently pursues a different strategy. Instead of understanding the shops as suppliers of the assembly line and as such optimizing them individually, the plant is aimed to be controlled as a whole. Results of this work will not be tested or applied in the running operation of a Daimler plant in the near future.

2. Structure and solution methods for mixed integer programs

2.1. Introduction to linear programming

Linear Programming is the problem of minimizing (maximizing) a linear objective function subject to a set of linear inequalities. We consider Linear Programs (LP) in the general form

$$\min c\,x \tag{2.1a}$$

$$\text{s.t.}\quad Ax \leq b \tag{2.1b}$$

$$x \geq 0 \quad , \tag{2.1c}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. We call $x_1, \ldots, x_n$ *decision variables*, (2.1a) the *objective function* and refer to (2.1b) and (2.1c) as the *constraints* of the optimization problem. We assume, that A has full rank m and there exists a feasible solution, then the feasible region of (2.1a)-(2.1c) is the intersection of a $d = n - m$ dimensional affine subspace and the nonnegative orthant in $\mathbb{R}^n$, a *polyhedron*. For a brief introduction and basic properties, see Section 2.2. An LP is called Mixed Integer Linear Program (MILP or MIP) if it adds the additional condition, that at least one of the decision variables $x_1, \ldots, x_n$ can only take integer values. Note, that minimization and maximation problems can easily be transformed into one another, since for the objective function holds

$$\min cx = \max -cx \quad . \tag{2.2}$$

An inequality constraint

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b \tag{2.3}$$

can be converted into an equality constraint by adding a nonnegative w, a so-called *slack variable*

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + w = b \quad . \tag{2.4}$$

An equality constraint

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b \tag{2.5}$$

can be expressed by two inequality constraints

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b \tag{2.6}$$

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b \quad . \tag{2.7}$$

Thus, an LP can always be written in the form (2.1a) - (2.1c) [Van14].

Remark. An LP with discrete - usually finite - but large search space is often referred to as *combinatorial optimization problem* in literature.

Some well-known examples for combinatorial optimization problems are [Law76]

(i) Traveling Sales Man Problem

(ii) Job Scheduling Problem

(iii) Knapsack Problem

(iv) Arc Coloring Problem.

2.2. Basic properties of polytopes

In the following we introduce the notion of bounded convex polytopes. For a detailed introduction we refer to [Zie12a]. Convex polytopes are fundamental geometric objects, whose discovery and study can be traced back to ancient Greece. Objects known to us as regular convex 3-dimensional polytopes are discusssed in Euclid's book XIII of the "Elements" [Zie01; Art99].

There are two alternative ways to specify convex polytopes. We have an interior description as convex hulls, in literature often referred to as $\mathcal{V}$-polytopes. And we have an exterior description as intersection of half spaces, known as $\mathcal{H}$-polytopes. The proof of the mathematical equivalence of these two formulations is nontrivial. For a detailed study see [Zie12a, Lecture 1].

Definition 2.1. Let $A \subset \mathbb{R}^d$. The subset A is called *convex* if the line segment for any two points in A lies in A, i.e. for all $x, \tilde{x} \in A$ we have $\lambda x + (1-\lambda)\tilde{x} \in A$ for all $\lambda \in [0,1]$ [Zie12a].

convex
nonconvex

The *convex hull conv*(A) of a set $A \subset \mathbb{R}^d$ is the smallest convex set that contains A

$$\operatorname{conv}(A) = \bigcap_{\substack{A \subseteq C \subseteq \mathbb{R}^d \\ C \text{ convex}}} C \quad . \tag{2.8}$$

Definition 2.2. Let $x_1, \dots, x_n \in \mathbb{R}^n$ and $\lambda_1, \dots, \lambda_n \in \mathbb{R}$. Then $\sum_{i=1}^n \lambda_i x_i$ is called a *linear combination* of the vectors $x_1, \dots, x_n$. It is futher a

(i) *conic combination*, if $\lambda_i \geq 0 \quad ,$

(ii) *affine combination*, if $\sum_{i=1}^n \lambda_i = 1 \quad ,$

(iii) *convex combination*, if it is conic and affine.

Definition 2.3. A *convex polyhedron* P is the intersection of a finite number of affine halfspaces, where an *affine halfspace* is a set

$$H^{\leq}(a,\beta) = \{x \in \mathbb{R}^d \mid \langle a,x \rangle \leq \beta\} \tag{2.9}$$

for some $a \in \mathbb{R}^d$, $\beta \in \mathbb{R}$. Thus every polyhedron is the set

$$P(A,b) = \{x \in \mathbb{R}^d \mid Ax \leq b\} \tag{2.10}$$

of feasible solutions to an LP $Ax \leq b$ for some matrix $A \in \mathbb{R}^{m \times d}$ and some vector $b \in \mathbb{R}^m$. A polyhedron is of dimension d if the points in the polyhedron affinely span $\mathbb{R}^d$. Moreover, a polyhedron is a topologically closed subset of $\mathbb{R}^d$ [Kai11a].

Note, that polyhedra are of great importance for Operations Research since they are not only the set of feasible solutions to LPs but even the solution of MILPs that can be reduced to linear optimization problems over polyhydra [Kai11a].

Remark. $\mathbb{P} \subset \mathbb{R}^n$ is a polytope if and only if $\mathbb{P}$ is a bounded polyhedron.

Definition 2.4. Let $S = \{x_1, \ldots, x_n\}$ be a finite set of points in a real vector space $V = \mathbb{R}^d$. A *$\mathcal{V}$-polytope* $\mathbb{P}$ is the convex hull

$$\mathbb{P} = \text{conv}(S) = \left\{ \sum_{i=1}^{n} \lambda_i x_i \mid \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1 \right\} \tag{2.11}$$

of the set S. By this definition the polytope is defined by its vertices.

Definition 2.5. A *$\mathcal{H}$-polytope* $\mathbb{P}$ is a bounded solution set of a finite system of linear inequalities

$$\mathbb{P} = \mathbb{P}(A,b) = \left\{ x \in \mathbb{R}^d \mid Ax \leq b \right\} \quad , \tag{2.12}$$

where $A \in \text{Mat}_{m \times d}(\mathbb{R})$, $b \in \mathbb{R}^m$. $\mathbb{P}$ is bounded in the sense that there is a constant N such that $\|x\| \leq N$ for all $x \in \mathbb{P}$. By this definition the polytope is defined by its facets.

These two definitions are equivalent as the following main theorem for polytopes states. Originally the WEYL and MINKOWSKI theorems show, that any polyhedron, and thus every polytope in particular, that is represented in one form, can also be represented in the other. In literature one often finds the theorems stated for the case of convex cones and then the results extended to more general polyhedra. We chose to present the combined WEYL-MINKOWSKI theorem formulated for the case of polytopes.

Theorem 2.1. (WEYL-MINKOWSKI Theorem.) *A subset $\mathbb{P} \subseteq \mathbb{R}^d$ is the convex hull of a point set (a $\mathcal{V}$-polytope)*

$$\mathbb{P} = conv(S) \quad \textit{for some } S \in \mathbb{R}^{d \times n} \tag{2.13}$$

if and only if it is a bounded intersection of halfspaces (a $\mathcal{H}$-polytope)

$$\mathbb{P} = \mathbb{P}(A,b) \quad \textit{for some } A \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m \tag{2.14}$$

[Zie12a, Theorem 1.1].

Proof. This theorem actually consists of two theorems. The "$\Rightarrow$"-direction is the MINKOWSKI-Theorem, the reverse direction is WEYLs Theorem.
Besides the formulation above, a version for cones and one for polyhedra can be found in literature. Both of them can be found with proofs in [CCZ10], for example. □

Note, that although the geometric objects are the same, as the WEYL-MINKOWSKI Theorem states, from a computational point of view it makes a difference whether a certain polytope is represented as a convex hull or via linear inequalities: the size of one description cannot be bounded polynomially in the size of the other, if the dimension d is not fixed [KS09].
According to our definition a polytope is always convex, so we will not distinguish between polytopes and convex polytopes.

Remark. A d-polytope $\mathbb{P}$ with $\mathbb{P} = \text{conv}(\mathbb{P} \cap \mathbb{Z}^d)$ is in literature often referred to as *integral polytope*. Integral polytopes play a crucial role in integer programming.

Example. The empty set, any point, any bounded line segment, any convex polygon in $\mathbb{R}^n$ is a polytope.

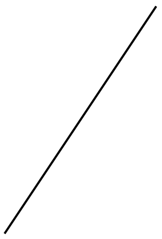


Figure 2.1.: 1-simplex, a 1-polytope

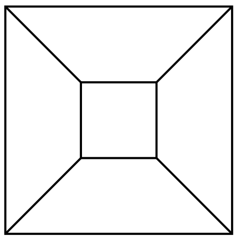


Figure 2.2.: SCHLEGEL diagram of a cube, a 3-polytope

By a classical theorem of MINKOWSKI any compact convex set can be written as a convex combination of its extreme points. In the context of polyhedral theory this theorem formulates as follows.

Theorem 2.2. (MINKOWSKI Theorem.) *Let $\mathbb{P}$ be a polytope in $\mathbb{R}^d$, let $\{x_1, \dots, x_n\}$ be a finite subset of $\mathbb{P}$. Let ext$(\mathbb{P})$ denote the set of vertices of $\mathbb{P}$.*
Then, the following two conditions are equivalent

1. $\mathbb{P} = conv\{x_1, \ldots, x_n\}$

2. $ext(\mathbb{P}) \subseteq \{x_1, \ldots, x_n\}$.

In particular, we have $\mathbb{P} = conv(ext(\mathbb{P}))$ *[Bro83, Theorem 7.2].*

Proof. This theorem applies for compact convex sets in $\mathbb{R}^d$ in general. Let $C \subset \mathbb{R}^d$ be a compact convex set and let $M \subset C$ be a subset. Suppose there is an extreme point x of C with $x \notin M$. Then, M is a subset of $C \backslash \{x\}$, and since $C \backslash \{x\}$ is convex by the definition of an extreme point, it follows, that $\text{conv}(M)$ is also a subset of $C \backslash \{x\}$. Set $\mathbb{P} = C$ and $\{x_1, \ldots, x_n\} = M$ and we have 1. $\Rightarrow$ 2.
For 2. $\Rightarrow$ 1. it suffices to show, that

$$C \subset \text{conv}(\text{ext}(C)) \quad . \tag{2.15}$$

Together with the opposite inclusion, that obviously holds, we have $C = \text{conv}(\text{ext}(C))$. And we also have $C = \text{conv}(M)$ for any subset M containing $\text{ext}(C)$. We prove (2.15) by induction on the dimension of C. For $\dim(C) = -1$ and $\dim(C) = 0$ there is nothing to be proven. For $\dim(C) = 1$ the statement is clearly valid. Suppose the statement is valid for all compact convex sets of dimension $< d$, and let C be a convex set of dimension d. Let $x \in C$, we have to show that x is a convex combination of extreme points of C. If x is an extreme point itself then there is nothing to be proven. So suppose x is not an extreme point, then there is a segment in C having x in its relative interior. Extending the segment, if necessary, we have two points y_0, y_1, such that $x \in]y_0, y_1[$. Let F_0 and F_1 be the smallest faces of C containing y_0 and y_1 respectively. Then, F_0 and F_1 are proper faces of C. They are, in particular, compact convex sets and they both have dimension $< d$. By induction hypothesis there are points $x_{01}, \ldots, x_{0p} \in \text{ext}(F_0)$ and $x_{11}, \ldots, x_{1p} \in \text{ext}(F_1)$ such that y_0 is a convex combination of the x_{0i}'s and y_1 is a convex combination of the x_{1j}'s. Since x is a convex combination of y_0 and y_1, it follows, that x is a convex combination of the x_{0i}'s and x_{1j}'s. In order to complete the proof, note, that the x_{0i}'s and x_{1j}'s are extreme points of C [Bro83, proof of Theorem 5.10]. □

Note, that a polytope has a finite number of faces.

The dimension $\dim(\mathbb{P})$ of a polytope $\mathbb{P}$ is the dimension of the affine hull $\operatorname{aff}(\mathbb{P})$

$$\operatorname{aff}(\mathbb{P}) = \left\{ y \in \mathbb{R}^d \,\middle|\, y = \sum_{i=1}^{n} a_i x_i, x_i \in \mathbb{P}, a_i \in \mathbb{R}, \sum_{i=1}^{n} a_i = 1 \right\} \quad . \tag{2.16}$$

A polytope $\mathbb{P}$ is denoted a *d-Polytope*, if $\dim(\mathbb{P}) = d$. By convention we have $\dim(\mathbb{P}) = -1$ if $\mathbb{P} = \emptyset$ [Bau+09].

Theorem 2.3. (CARATHÉODORY'S theorem.) *Let $\mathbb{P}$ denote a d-polytope. Any point in the polytope is a convex combination of at most $d+1$ extreme points of the polytope [Fad15].*

Definition 2.1. Let $\mathbb{P}$ be a polytope. A linear inequality $cx \leq c_0$ is valid for $\mathbb{P}$, if it is satisfied for all points $x \in \mathbb{P}$. A *face F* of $\mathbb{P}$ is any set of the form

$$F = \mathbb{P} \cap \{x \in \mathbb{R}^d \mid cx = c_0\} \tag{2.17}$$

The dimension of a face $\dim(F)$ is the dimension of its affine hull $\dim(\operatorname{aff}(F))$. A k-dimensional face is being called a *k-face*.

Remark. For the valid inquality $0x \leq 0$, we get that $\mathbb{P}$ itself is a face of $\mathbb{P}$.

Lemma 2.4. *Every face of a polytope is a polytope.*

Proof. Let $\mathbb{P} = \operatorname{conv}(S)$ be a polytope and let F be a face of $\mathbb{P}$ defined by $cx \leq c_0$. Let $S_0 = \{x \in S \mid cx = c_0\}$ and $\mathring{S} = \{x \in S \mid cx < c_0\}$. Then, $S = S_0 \cup \mathring{S}$. The following calculation shows, that $F = \operatorname{conv}(S_0)$. The convex combination $\lambda_1 x_1 + \cdots + \lambda_n x_n$, $x_i \in \mathring{S}$ satisfies $cx = c_0$ if and only if $\lambda_i = 0$ for all i. In order to see this let $\mathring{S} = \{x_1, \ldots, x_k\}$ and $S_0 = \{x'_1, \ldots, x'_l\}$. Let $x \in F$

$$\begin{aligned}
c_0 = cx &= c((\lambda_1 x_1 + \cdots + \lambda_k x_k) + (\lambda'_1 x'_1 + \cdots + \lambda'_l x'_l)) \\
&= (\lambda_1 c x_1 + \cdots + \lambda_k c x_k) + (\lambda'_1 c x'_1 + \cdots + \lambda'_l c x'_l) \\
&\leq (\lambda_1 c_0 + \cdots + \lambda_k c_0) + (\lambda'_1 c_0 + \cdots + \lambda'_l c_0) \\
&= c_0(\lambda_1 + \cdots + \lambda_k + \lambda'_1 + \cdots + \lambda'_l) \quad ,
\end{aligned}$$

where $\lambda_i c x_i \leq \lambda_i c_0$ for $1 \leq i \leq k$ and $\lambda'_j c x'_j = \lambda'_j c_0$ for $1 \leq j \leq l$. For this to hold, we must have $\lambda_i c x_i = \lambda_i c_0$, which only holds if $\lambda_i = 0$ for all i. Thus, we have
$x = \lambda'_1 x'_1 + \cdots + \lambda'_l x'_l$, so $x \in \operatorname{conv}(S_0)$. □

Definition 2.2. Let $\mathbb{P}$ be a polytope of dimension d. The 0-dimensional faces are called *vertices*. The 1-faces are called *edges*. The $(d-2)$-faces are called *ridges* and $(d-1)$-faces are called *facets*.

Note, that the facets of a polytope correspond to the exterior description of that polytope ($\mathcal{H}$-polytope). Exactly one inequality for each facet is needed.
Since some vertices are connected by edges, we can define an undirected graph
$\Gamma = (V(\Gamma), E(\Gamma))$, where $V(\Gamma) = \{v \big| v \in \text{vert}(\mathcal{P})\}$ and
$E(\Gamma) = \{(v,w) \in V(\Gamma) \big| \exists \text{ edge } E \text{ of } \mathcal{P} \text{ such that } v \in E, w \in E\}$.

Remark. In certain applications two *improper* faces are introduced: $\emptyset$ and $\mathbb{P}$ itself. Then, we have the following additional properties

i For every two faces $F_1, F_2 \subset \mathbb{F}$ we have $F_1 \cap F_2$ is also a face of $\mathbb{F}$. We write $F_1 \wedge F_2$.

ii For every two faces $F_1, F_2 \subset \mathbb{F}$ there exists a uniquely defined face $F_1 \vee F_2$, the "smallest" face that contains both, F_1 and F_2.

Proposition 2.5. *Let $\mathbb{P}$ be a d-polytope, let $f_k(\mathbb{P})$ denote the number of different k-faces of $\mathbb{P}$. Then, we have*

$$f_k(\mathbb{P}) \leq \binom{f_0(\mathbb{P})}{k+1} \quad , \tag{2.18}$$

with the convention $f_k(\mathbb{P}) = 0$ for $k > d$ or $k < -1$ [Gru13].

2.2.1. Representation and permutation polytopes

Let G be a finite group. Let $\rho : G \to \text{GL}(V)$ be a real representation of G. It induces an $\mathbb{R}$-algebra homomorphism from the group algebra $\mathbb{R}[G]$ to $\text{End}(V)$, which we also denote by ρ [Bau+09].

Definition 2.3. The *representation polytope* $\mathbb{P}(\rho)$ of the representation ρ is defined as the convex hull of $\rho(G)$ in the vector space $\text{End}(V)$ [Bau+09, Definition 1.1].

Let $G = \langle \pi_1, \ldots, \pi_m \rangle$ be a permutation group over a finite set X. Let n denote the number of elements of X. Consider the standard permutation representation χ with

$$\chi : G \to \mathbb{R}^{X^2} \quad , \tag{2.19}$$

$$\chi(\pi)_{ij} = \begin{cases} 1 & \text{if } \pi(i) = j, \\ 0 & \text{else} \quad . \end{cases} \tag{2.20}$$

This is the usual way of representing permutations of X by $n \times n$ permutation matrices.

Let S_n denote the symmetric group over $\{1,\ldots,n\}$. Let G be any finite group. Let $\chi : G \to S_n$ be the permutation representation, thus G can be identified with the permutation group $\chi(G) \leq S_n$. We write in short $G \leq S_n$ in both cases, whether G is a subgroup of S_n or identified as such by χ.

Definition 2.4. For $G \leq S_n$ the *permutation polytope* associated to G is defined as

$$\mathbb{P}(G) = \operatorname{conv}(\chi(G)) \subset \operatorname{Mat}_{n\times n}(\mathbb{R}) \cong \mathbb{R}^{n^2} \tag{2.21}$$

[Bau+09, Definition 1.2].

In particular, any permutation polytope is a representation polytope [Bau+09].

For an extensive investigation of general permutation polytopes, as well as their dimensions and graphs, we refer to the work by Guralnick and Perkinson [GP06].

The Birkhoff Polytope

We consider the special case that G is the symmetric group over X.

Definition 2.5. The nth *Birkhoff polytope B_n* also called *assignment polytope* is defined by

$$B_n = \operatorname{conv}(\chi(\pi) \,|\, \pi \in S_n) \quad . \tag{2.22}$$

The Birkhoff polytope is one of the most important polytopes. It arises in various fields of mathematics, in combinatorics ([Ath05]), statistics (see [Pak00]), optimization ([Pak00] et al.) or representation theory (e.g. [Bra+91], [Onn93]) [Bau+09].

The BIRKHOFF-VON NEUMANN Theorem characterizes the nth Birkhoff polytope B_n as the polytope of all doubly stochastic $n \times n$ matrices. This theorem was stated by George David Birkhoff in 1946 and independently proven by John von Neumann in 1953.

Theorem 2.6. *(*BIRKHOFF-VON NEUMANN *Theorem.) B_n is an $(n-1)^2$ dimensional polytope with $n!$ vertices having the following inequality description*

$$B_n = \left\{ (x_{ij}) \in \mathbb{R}^{n^2} \,\middle|\, x_{ij} \geq 0 \textit{ and } \sum_i x_{ij} = \sum_j x_{ij} = 1 \textit{ for } 1 \leq i,j \leq n \right\} \quad . \tag{2.23}$$

Proof. The original proof by Birkhoff can be found in [Bir46], an alternative version in [Hur]. □

The facets of $\mathbb{B}_n$ are defined by the inequalities $x_{ij} \geq 0$ for $1 \leq i,j \leq n$.

Remark. In particular, the Birkhoff polytope is a $0/1$-polytope, that is the convex hull of vectors in $\{0,1\}^{n^2}$. The probably most famous application of $0/1$-polytopes is the Travelling Salesman Problem [BS96]. The Birkhoff polytope is a so called *transportation polytope $T(a,b)$*

$$T(a,b) = \left\{ (x_{ij}) \in \mathbb{R}^{m\times n} \,\middle|\, x_{ij} \geq 0 \text{ and } \sum_i x_{ij} = b_j, \sum_j x_{ij} = a_i \text{ for } 1 \leq i \leq m, 1 \leq j \leq n \right\} \quad . \tag{2.24}$$

Remark. Two permutations $\sigma, \pi \in S_n$ correspond to an edge of B_n if and only if $\sigma^-1\pi$ is a cycle.

Although polytopes have been studied since ancient Greece, calculating the volume of a polytope still remains tough. Even for relatively small values of n, computing $\mathrm{vol}(\mathbb{B}_n)$ represents a significant challenge. The explicit volume is known up to $n = 10$ [BP03].

$$\mathrm{vol}(\mathbb{B}_{10}) = \frac{727291284016786420977508457990121862548823260052557333386607889}{82816086010676685512567631879687272934462246353308942267798072138805557399562702937508835048928208848640000000} \quad .$$

In [CM07] Canfield and McKay present an symptotic formula for the volume of the nth Birkhoff polytope $\mathbb{B}_n$

$$\mathrm{vol}(\mathbb{B}_n) = \frac{1}{(2\pi)^{n-1/2} n^{(n-1)^2}} \exp\left(\frac{1}{3} + n^2 + \mathcal{O}(n^{-1/2+\varepsilon})\right) \tag{2.25}$$

for any $\varepsilon > 0$ as $n \to \infty$.

De Loera et al. [LLY09] present an exact combinatorial formula

$$\operatorname{vol}(\mathbb{B}_n) = \frac{1}{((n-1)^2)!} \sum_{\sigma \in S_n} \sum_{T \in \operatorname{Arb}(l,n)} \frac{\langle c,\sigma \rangle^{(n-1)^2}}{\prod_{e \notin E(T)} \langle c, W^{T,e}\sigma \rangle} \quad . \tag{2.26}$$

For two vertices x,y of a polytope $\mathbb{P}$ recall, that the *distance* $\operatorname{dist}(x,y)$ is the minimal number of edges needed to go from x to y in the graph of $\mathbb{P}$. Let $\Delta(\mathbb{P})$ denote the *diameter* of $\mathbb{P}$, that is the maximum possible distance between two vertices in the graph of the polytope. In the context of algorithms in optimization theory we can characterize $\Delta(\mathbb{P})$ as the best-possible number of iterations initiated at the worst vertex [Loe13]. This relation to the Simplex Algorithm in Linear Programming was the origin of the following HIRSCH Conjecture. This conjecture was posed by Warren M. Hirsch in 1957 and published by George D. Dantzig in 1963 [Dan63, p.168]. It states, that any two vertices of a d-polytope with n facets can be connected by a graph of at most $n-d$ edges.

Conjecture 2.7. (HIRSCH *Conjecture). Let $n > g \geq 2$. Let $\mathbb{P}$ be a d-dimensional polytope with n facets and let $G(\mathbb{P})$ be its graph. Then, $\Delta(G(\mathbb{P})) \leq n-d$.*

As we know now the conjecture, as stated by Dantzig, is false [Zie12b]. Credits for this result go to Klee and Walkup, who gave a counterexample in [KW67]. For a nice overview giving correct kinds of polytopes for this conjecture see [KS09].

2.3. Performance of famous algorithms on the Birkhoff polytope

Over the years, tens of algorithms for linear programming have been suggested, however, most of them could not compete with the historically first algorithm, Dantzig's simplex method. Nevertheless, at least two methods are worth mentioning. In the following section we give a survey of these algorithms and their performance on polytopes and the Birkhoff polytope in particular. Therefore, we mainly follow [MG07].

Recall the general form of a mixed integer minimization problem

$$\min cx \tag{2.27a}$$
$$\text{s.t.} \quad Ax \leq b \tag{2.27b}$$
$$x \geq 0 \tag{2.27c}$$
$$x_j \text{ integer} \quad , \tag{2.27d}$$

The *LP relaxation* of a MIP is the continuous optimization problem that is obtained by dropping the integrality restrictions. For one, the LP relaxation provides an lower bound for the optimal value of the MIP. In addition, if an optimal solution to the LP relation is found, that satisfies the integrality constraints, it is also an optimal solution to the corresponding MIP [JNS00].
According to the rules of a branch-and-cut method, an enumeration tree is spanned, where every node is solved by a linear programming algorithm, commonly the simplex algorithm. By moving down the tree, more and more integer variables are fixed.
Branch-and-cut is a combination of branch-and-bound and a cutting plane method. In a cutting plane method, the LP relaxation of the integer program is solved. If the optimal solution is feasible in the integer program, it also solves the integer program. Otherwise, a constraint is added to the linear program that separates this solution from the set of feasible solutions to the integer program and the new program is solved again.
In a branch-and-bound method the first step also is to solve the LP relaxation. If the optimal solution is feasible in the integer program it also solves the integer program. Otherwise, the relaxation is split into two subproblems, usually by fixing a particular variable at zero or one. One subproblem is chosen and the LP relaxation of that subproblem is solved.
Depending on whether the subproblem is infeasible or the solution is also feasible to the integer program and whether objective value of the optimal solution is worse than the one of the known solution to the integer program the tree is pruned at this node or the node is split into two further subproblems [Mit96].
The computational efficiency in solving a mixed integer program depends on the number of iterations and the effort on each iteration. We first focus on the latter here. The significance and impact of the branching strategies will be revisited in Chapter 4.

2.3.1. Simplex Algorithm

As being one of the most important problems studied by researchers in mathematics, operations research und computer science, the understanding of linear programing has improved vastly in the last 70 years. The classical method for solving linear programs is still the well-known simplex algorithm, presented by George Dantzig in 1947. The simplex method considers the combinatorial structure of the faces of the polytopes.
In order to state it briefly, the algorithm proceeds by walking from one vertex to another. At each step, a vertex is chosen that is superior with respect to the objective function. The algorithm either determines an optimal solution, gets the indication of unboundedness or determines infeasibility. We follow the work of Adler et al. [APR14] as well as Friedmann et al. [FHZ11] and Friedmann [Fri11] and Pak [Pak00] for an overview of the performance of the simplex method in linear programs.
A *strongly polynomial algorithm* for a linear program is one whose total number of operations is bounded polynomially in the input length. Several variants of Dantzig's simplex method have been developed and many of them have been proven to have exponential worst-case performance. In order to specialize the simplex method to a concrete algorithm, a pivoting rule needs to be provided, determining the choice of the next vertex in case there is more than one candidate. It is still an open question if there exists a pivoting rule requiring a polynomial number of steps on any linear program. However, in practice the simplex method usually performs very well. The primal simplex method usually requires at most $2m$ to $3m$ pivots to obtain optimality. This was stated based on numerous experiments with thousands of practical problems by Dantzig very early [Dan63, p. 160].
In graph-theoretic terms, the simplex algorithm computes a path in the graph Γ defined by the vertices and edges of the underlying polytope. The efficiency of the method is determined by the length of that path. Therefore, the diameter of the polytope determines a lower bound for the number of pivot steps [FT94]. Recall the HIRSCH conjecture (2.7) stating, that a walk of length of at most $n-d$ should always exist. As a significant progress on this conjecture we mention the work of Kalai and Kleitman [KK92], who proved, that there always exists a walk of length at most $n^{\log_2 d+2}$. However, the existence of such a short walk, does not imply that the simplex algorithm finds it [ST04].
Recall, that the vertices of $\mathbb{B}_n$ are in a one-to-one correspondence of the elements of the symmetric group S_n, and the edges correspond to pairs of permutations $\sigma, \pi \in S_n$ such that $\sigma^{-1}\pi$ is a single cycle. Since every permutation can be represented by a product

of two cycles [HKL04, Proposition 5], conclude, that the diameter of the Graph Γ_n of vertices and edges of $\mathbb{B}_n$ is 2. Thus, we can reach the desired minimum in at most two steps [Pak00]. In order to determine the maximal number of steps, we consider the following "toy version" of the simplex algorithm presented by Pak [Pak00].

Theorem 2.8. *Let $\mathbb{P}_n$ be the* nth *Birkhoff polytope. Consider the linear functional ϕ*

$$\phi = c_\alpha x = x_{11} + \alpha x_{12} + \cdots + \alpha^{n-1} x_{1n} + \alpha^n x_{21} + \cdots + \alpha^{n^2-1} x_{nn} \quad . \tag{2.28}$$

We think of the graph Γ_n as a partially ordered set with $\sigma \geq \pi$ if there exists a sequence of edges between σ and π on which ϕ decreases. Then, for $0 < \alpha < 1/(n+1)$ there exists a decreasing sequence of vertices of $\mathbb{P}_n$ of length $> Cn!$, for a universal constant $C > 0$ [Pak00, Theorem 1.4].

Proof. We prove the theorem by constructing a desired sequence from the permutation $(1,2,\dots,n)$ to $(n,(n-1)\dots,1)$. For $n = 2$ the chain is trivial. We prove the statement by induction over n. Suppose we know the chain of permutations for $n \leq m-1$. Let the chain for $n = m$ be as follows: It starts with

$$(1,2,\dots,m-1,m) \to \cdots \to (1,m,m-1,\dots,2) \quad , \tag{2.29}$$

where the middle part is the sequence of permutations for $n = m-1$. We get the rest of the chain by going to an element like $(2,*,\cdots,*)$, then some chain in the middle, then $(2,n,n-1,\dots,3,1)$, to $(3,*,\cdots,*)$ and so on, until we reach $(n,n-1,\dots,1)$. All we need now is to describe the chain from $(i,n,n-1,\dots,i+1,i-1,\cdots,1)$ to $(i+1,n,n-1,\dots,i+2,i,\dots,1)$. Let therefore $k = \lfloor n/2 \rfloor$. Observe, that it is always possible to move from $\sigma_1 = (i,n,n-1,\cdots,1)$ to $\sigma_2 = (i+1,1,2,\dots,k,*,\dots,*)$ since $\sigma_2 \leq \sigma_1$ and there is a rearrangement of $*$ such that $\sigma_1^{-1}\sigma_2$ is a long cycle. Now get from σ_2 to $\sigma_3 = (i+1,1,2,\dots,k,m,m-1,\dots,k+1)$ by decomposing $\sigma_2^{-1}\sigma_3$ into a product of cycles and using them one by one. Use induction again for $n = m-k$. We reach $(i+1,n,n-1,\dots,1)$, which completes the construction. In order to compute the length of the whole chain of permutations, let L_n denote the length of the chain on S_n. The induction gives us

$$L_{n+1} = (n+1)(L_n - (\lfloor n/2 \rfloor + 1)!) \quad . \tag{2.30}$$

Dividing both sides by $(n+1)!$ gives us

$$\frac{L_{n+1}}{(n+1)!} > \frac{L_n}{n!} - \frac{1}{\lfloor n/2 \rfloor !} \quad . \tag{2.31}$$

Since $\sum_k \frac{2}{k!} \to \frac{2}{e} < 1$, we have $\frac{L_n}{n!} > 1 - \frac{2}{e}$ for all n. This implies the result [Pak00, proof of Theorem 1.4]. □

Thus, the maximum running time of the simplex algorithm on the Birkhoff polytope is exponential. Pak also showed that the expected average running time of the simplex method on the Birkhoff polytope with cost function c_α is $\mathcal{O}(n \log n)$ [Pak00].

Theorem 2.9. *Let ϕ be as above, let $0 \leq \alpha \leq \alpha_0$, then we have an expected running time of the algorithm of $\mathcal{O}(n \log n)$ [Pak00, Theorem 1.5].*

Proof. Let D_n be the total number of cycles in the symmetric group S_n, i.e. the degree of a graph Γ. We have

$$D_n = \sum_{l=2}^{n} \binom{n}{l} (l-1)! = n! \sum_{l=2}^{n} \frac{1}{l(n-l)!} < c(n-1)! \tag{2.32}$$

for some universal constant $c > 1$. Therefore, the probability of the randomly chosen cycle being a long cycle is $> 1/c$.
Now suppose the algorithm is at a permutation $\sigma = (n, n-1, \ldots, n-k, j, *, \cdots, *)$, with $1 \leq j < n-k-1$. Note, that elements $n, n-1, \ldots, n-k$ are fixed at this point, in the sense that no decreasing edge can ever change them. We compute the number of decreasing edges (σ,ω) leaving σ which do not change j. This number is not greater than D_{n-k-2} and at least $(n-k-1-j)(n-k-2)!$, which is the number of all cycles on the last $n-k-1$ elements and such that $\omega(k+2) > j$. Therefore, the probability, that in a decreasing edge (σ, ω), we have $\omega(k+2) > j$ is at least $c' = 1/(1+c)$. Conclude, that with probability $> c'$ the $(k+2)$th element j in a permutation changes to a uniform element $\in [j+1, n-k-1]$. Therefore, after $\mathcal{O}(\log n)$ moves the $(k+2)$nd element in a permutation will become $n-k-1$ and "stuck". From here it takes $\mathcal{O}(n \log n)$ steps for all elements to get fixed, which means we reach $(n, n-1, \ldots, 1)$ [Pak00, proof of Theorem 1.5]. □

One of the most important parameterizations of the simplex algorithm is the pivoting rule. It determines which nonbasic variable is to enter the basis at each step of the iteration. It still remains an open challenge to find a pivot rule that makes the simplex method run in polynomial time for all LPs or show that none exists. For discussions of this problem see [APR14; Tod02] and the references therein.

2.3.2. Ellipsoid Method

The ellipsoid method is designed to solve decision problems rather than optimization problems. It was originally invented by Shor [SZ71], Yudin and Nemirovski [YN76a; YN76b; YN77] to solve a certain kind of nonlinear optimization problems in the 1970s. Nine years later Khachyian [Kha79] showed how the ellipsoid method can be used to solve linear programs in provable polynomial time [MG07]. We consider the decision problem of finding a feasible point to a system of linear inequalities [Reb09]

$$Ax \leq b \quad . \tag{2.33}$$

Roughly, the ellipsoid method constructs a sequence of ellipsoids E_k, each of which contains a point satisfying the constraints (2.33), if one exists, starting with an initial ellipsoid containing the solution set of (2.33). On the $(k+1)$st iteration the algorithm checks whether the center x_k of the current ellipsoid E_k satisfies constraints (2.33). If so, it stops. If not, some constraints violated by x_k, say

$$a^T x \leq b_i \tag{2.34}$$

is chosen and the ellipsoid of minimum volume that contains the half-ellipsoid

$$E_{k+1} = \{x \in E_k \mid a^T x \leq a^T x_k\} \tag{2.35}$$

is constructed. Denote the center of E_{k+1} by x_{k+1}. The iterative step above is repeated. One can determine whether (2.27a) is feasible or not in a polynomial number of iterations by modifying the algorithm to account for finite precision arithmetic, applying it to a suitable perturbation of (2.27a) and choosing E_0 appropriately [BGT81]. This result of being “theoretically efficient” was first noted by L. G. Khachiyan in 1979, though the ellipsoid method did not prove to be “practically efficient”. In fact, it has been shown, that the expected running time of the simplex algorithm is polynomial and much better than the expected running time of the ellipsoid method [GLS88]. The decision problem (2.33) is closely related to the linear program

$$\min c\,x \tag{2.36a}$$
$$\text{s.t.} \quad Ax \leq b \tag{2.36b}$$
$$x \geq 0 \quad . \tag{2.36c}$$

Due to duality, theory solving problem (2.36a)-(2.36c) is equivalent to finding a feasible point of the following system of linear inqualities:

$$Ax \leq \quad b \tag{2.37a}$$
$$-x \leq \quad 0 \tag{2.37b}$$
$$-Ay \leq -c \tag{2.37c}$$
$$-y \leq \quad 0 \tag{2.37d}$$
$$-cx + bx \leq \quad 0 \tag{2.37e}$$

where the third and fourth inequality come from the dual problem of (2.36a)-(2.36c). The last inequality comes from the Strong Duality Theorem. The equivalence of the two problems means that vector x of each solution (x,y) of (2.37a)-(2.37e) is an optimal solution of (2.36a)-(2.36c), and y is an optimal solution for the dual problem. In addition, for every solution to the optimization problem there exists a vector such that this pair is feasible for (2.37a)-(2.37e) [Reb09]. From this equivalence Gács and Lovász [GL81] concluded, that the linear programming problem (2.36a)-(2.36c) can be solved in polynomial time. It has to be noted, that the problem size increases from n to $n+m$. Since a polytope is bounded, the ellipsoid method is a polynomial-time algorithm. Usually the simplex method performs better in practice [BGT81].

2.3.3. Interior Point Method

In contrast to the simplex algorithm, that explores the extremal points of the feasible region of a linear program, the interior point method walks its way through the interior of the polytope. In 1984, the publication of a paper by Karmarkar [Kar84] built the foundation of the research on this new method. It was originally invented in the context of linear programming. See also [Wri97] for more discussion on different variations of the interior point method. Due to its ability of scaling to large problem sizes, the interior point method has attracted a lot of attention in recent years, since the demand

for large-scale optimization in industries and engineering has increased. Another advantage of the interior point method is, that it is applicable to semidefinite, quadratic and other nonlinear programs. Like the ellipsoid method, most variants of the interior point method have polynomial time complexity [MPR98].
Consider a linear program in the standard form

$$\min cx \tag{2.38a}$$
$$\text{s.t.} \quad x \in X \tag{2.38b}$$
$$x \geq 0 \quad , \tag{2.38c}$$

where X is a closed convex domain. The basic idea of the interior point method is to equip the linear program with a barrier function F, a smooth and strongly convex function defined on $int(X)$, such that $F(x_k) \longrightarrow \infty$ along every sequence $(x_k) \in int(X)$ with $x_k \longrightarrow \bar{x} \in \partial X$. A possible barrier function subproblem is

$$\min c\,x - \mu \sum_{i=1}^{n} \log(x_i) \tag{2.39a}$$
$$\text{s.t.} \quad Ax = b \tag{2.39b}$$
$$x \geq 0 \quad , \tag{2.39c}$$

where μ denotes the so-called barrier parameter, a positive constant [NT09]. The interior point method appears to be superior to the simplex algorithm when it comes to problems with a large number of constraints and variables (say, more than one thousand) [MPR98]. One reason may be the possibility of parallelization where the interior point method often is preferred to the simplex method due to matrix sparsity and communication during the solving process [DSA98]. For the application on MIPs, the integer point method is usually combined with a branch-and-cut algorithm. The challenge thereby is that a solution of the relaxed program might in general not be a good starting point for an interior point method, since it is close to the boundary of the feasible region. In particular, this method tends to have very poor performance: when the starting point is a nonoptimal extreme point, then several iterations are needed to walk towards the center of the feasible region.
This is usually overcome by stop working on the current relaxation before it is solved completely: the earlier we can find good cutting planes, the better, that means further in the interior, the initial solution to the next relaxation [Mit96]. This proceeding obviously saves iterations spent on the current relaxation and also on the next relaxation, because the starting point for the next relaxation is more centered. The advantages and

diasadvantages of this approach of early termination can be found in [Mit96]. In the work of Mitchell, a branch-and-bound interior point algorithm can be found [Mit96, Section 2.2].
We summarize, that an interior point method can be useful in solving large integer programs, especially when a feasible solution with good objective value is sufficient and optimality is not necessarily required. However, it depends on the problem specifications if an interior point method, despite its difficulties, outperforms a simplex algorithm in a branch-and-bound algorithm [BM98].

2.4. Reformulation techniques for MIPs

In this section we examine ways to reformulate mixed integer programs. Given an initial formulation of an LP, we are interested in different ways of reformulating the problem in order to obtain improved problem formulations by using the problem structure. Improvement may be obtained by reduction of the number of variables so that calculations are typically faster, treatment or elimination of symmetry among solutions, as well as identification of variables that are more effective as branching variables [VW10].

2.4.1. Reformulation based on decomposition

Consider an integer program of the form

$$\begin{aligned} \min\ & cx \\ \text{s.t.}\ & Ax \leq b \\ & x \in \mathbb{Z}^n_+ \quad . \end{aligned}$$

Let $X = \{x \in \mathbb{Z}^n_+ | \ Ax \leq b\}$. We are interested in finding alternative problem descriptions, that are more effective in any kind [Bad+18]. Both classical decomposition methods, *Dantzig-Wolfe decomposition* and *Benders' decomposition*, are concerned with decomposing the original problem in a master problem and one or several subproblems.
The Dantzig-Wolfe decomposition exploits the linear programming formulation of the

Lagrangean dual (see Section 2.4.3) as master problem by using dynamic column generation [VS06].
The key point of Bender's decomposition is to derive a master problem, that provides a lower bound for the original problem. The problem is either solved, if possible, or, if not, a cut is generated and added to the master problem, it is called a Bender's cut [CJ05]. A Bender's cut is inferred by the dual of the subproblem and thus valid for all variables of the master problem and rather effective.

Remark. Anticipating the results of Chapter 3 we note, that due to the strong relation of all of the decision variables, a decomposition approach is not pursued.

2.4.2. Projection of polyhedrons

The structure of a linear program or mixed integer program might admit the focus on a subset of the more important variables (e.g the integer variables in a MIP).

Definition 2.6. Let $U \subset \mathbb{R}^n \times \mathbb{R}^p$ denote a subset, the projection of U on the first n variables $x = (x_1, \ldots, x_n)$ is the set

$$\operatorname{proj}_x(U) = \{x \in \mathbb{R}^n | \; \exists w \in \mathbb{R}^p with (x,w) \in U\} \tag{2.41}$$

[VW10, Definition 5].

The FARKAS' Lemma characterizes the optimality condition for several problems by stating, that either the primal problem or the dual problem of a MIP has a solution [Dax97].

Lemma 2.10. (FARKAS' Lemma.) *Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then, exactly one of the following two conditions holds*

i there exists $x \in \mathbb{R}^n$ such that $Ax = b$, $x \geq 0$,

ii there exists $y \in \mathbb{R}^m$ such that $A^T y \geq 0$, $y^T b < 0$.

A proof of this lemma can be found in numerous optimization textbooks, inter alia [CCZ14, Theorem 3.4].
Furthermore, the FARKAS' Lemma gives a characterization of the projection of a polyhedron.

Theorem 2.11. *Let $Q = \{(x,w) \in \mathbb{R}^n \times \mathbb{R}^p_+ | \; Gx + Hw \geq d\}$. Then,*

$$\begin{aligned} proj_x(Q) &= \{x \in \mathbb{R}^n | \; v(d - Gx) \leq 0 \; \forall v \in V\} \\ &= \{x \in \mathbb{R}^n | \; v^j(d - Gx) \leq 0 \, for \; j = 1, \ldots, J\} \quad , \end{aligned} \tag{2.42}$$

where $V = \{v \in \mathbb{R}^m_+ | \; vH \leq 0\}$ and $\{v^j\}_{j=1,\ldots,J}$ are the extreme rays of V [VW10, Theorem 4].

Proposition 2.12. *If $P \subset \mathbb{R}^n \times \mathbb{R}^p$ is a polytope, then the projection of P onto $\mathbb{R}^n$ is a polytope [JKM04, Proposition 15].*

2.4.3. Relaxation techniques in MIP solving

Since a matrix is a permutation matrix if - and only if - it is both orthogonal and doubly stochastic, much work focused on finding semidefinite relaxations on orthogonality constraints. These relaxations are convex and hence tractable, but usually they become quite large and scale poorly [Fog+13]. Lim and Wright [LW14] present a typical way of converting a discrete optimization problem over a set of permutations into a continuous relaxation. It follows three steps

(i) represent the permutations by permutation matrices,

(ii) relax to the convex hull of the set of permutation matrices, which is the Birkhoff polytope,

(iii) relax other constraints to ensure convexity/continuity.

This results in a significant rise in the number of variables, from n variables that are needed to represent the permutation directly to $\mathcal{O}(n^2)$, that are required to represent the Birkhoff polytope.

Lagrangean relaxation

Lagrangean relaxation is a technique based on the observation that many computationally hard linear programming problems can be viewed as easy problems complicated by a relatively small set of constraints. Using the Lagrangean relaxation produces a linear program that is easier to be solved and whose optimal value is a lower bound on the optimal value of the original problem. Thus, this technique can be used to provide bounds in a branch-and-bound algorithm [Fis04]. The Lagrangean relaxation was first applied to the Travelling Sales Man Problem (TSP) in 1970/71 by Held and Karp [HK70]. Since then the number of problems, whose computational time can be speeded up by the use of this technique, has grown. Among them are - in addition to the TSP - Knapsack problem generalized assignment problem, set covering, scheduling problems to name the most famous ones. Several modified versions of the Lagrangean method have been developed, we follow the version presented by [VW10]. Consider an optimization problem of the general form

$$\begin{aligned} &\min cx \\ \text{s.t.}\quad & Bx \leq b && \text{(2.43a)} \\ & Dx \leq d && \text{(2.43b)} \\ & x \in \mathbb{Z}^n_+ \quad , \end{aligned}$$

where (2.43b) represent rather "complicated constraints" and (2.43a) are "more tractable" and can be solved quickly in practice. The Lagrangean approach for this problem consists of turning (2.43b) into constraints that can be violated at a price π. Let $Z = \{x \in \mathbb{Z}^n_+ | Bx \leq b\}$.
This gives rise to the so-called *Lagrangean subproblem*

$$L(\pi) = \min_x \{cx + \pi(Dx - d) | Bx \leq b, x \in \mathbb{Z}^n_+\} \quad . \tag{2.44}$$

For any $\pi > 0$ the dual function $L(\pi)$ defines a lower dual bound on the optimal value x^* of the original problem, since we have

$$cx^* \geq cx^* + \pi(Dx^* - d) \geq L(\pi) \quad . \tag{2.45}$$

Maximizing this bound over the set of admissible penalty vectors is known as the *Lagrangean dual*

$$z_{LD} = \max_{\pi > 0} L(\pi) = \max_{\pi > 0} \min_{x \in Z} \{cx + \pi(Dx - d)\} \quad . \tag{2.46}$$

Let $\{x^t\}_{t=1,\dots,T}$ denote the set of extreme points of $\mathrm{conv}(Z)$. Since the Lagrangean subproblem achieves its optimum at x^t for a $t \in \{1,\dots,T\}$, one can write

$$z_{LD} = \max_{\pi>0} \min_{t=1,\dots,T} \{cx^t + \pi(Dx^t - d)\} \quad . \tag{2.47}$$

Let σ denote a lower bound on $(c+\pi D)x^t$, then the Lagrangean dual can be rewritten

$$\begin{aligned} \max\ & -\pi d + \sigma \\ \text{s.t}\ & \sigma - \pi D x^t \le cx^t, \quad t = 1,\dots,T \\ & \pi \ge 0 \\ & \sigma \in \mathbb{R} \end{aligned}$$

Taking its linear programming dual we obtain

$$\begin{aligned} \min\ & \sum_{t=1}^{T} (cx^t)\lambda_t \\ \text{s.t.}\ & \sum_{t=1}^{T} (Dx^t)\lambda_t \le d \\ & \sum_{t=1}^{T} \lambda_t = 1 \\ & \lambda_t \ge 0 \end{aligned}$$

The following theorem summarizes the results.

Theorem 2.13. (Lagrangean Duality.)

$$z_{LD} = \min\{cx \mid Dx \le d, x \in conv(Z)\} \quad . \tag{2.50}$$

By definition of the set $\{x^t\}_{t=1,\dots,N}$ *we have*

$$conv(Z) = \{x = \sum_{t=1}^{T} x^t \lambda_t \mid \sum_{t=1}^{T} \lambda_t = 1, \lambda_t \ge 0, t = 1,\dots,T\} \quad . \tag{2.51}$$

Thus, the value of the Lagrangean dual is equal to the value of the original LP obtained by minimizing cx over the intersection of the complicated "constraints" with the convex hull over the "tractable" set [VW10, Theorem 5].

Remark. Anticipating the results of Section 4.2.1 we note that the structure of MIP

(3.29a)-(3.29m) appears suitable for Langrangean relaxation.

2.4.4. On the extended formulation of a polytope

In the following section we give a brief overview of the concept of representing a polytope associated with an optimization problem as a linear projection of a higher dimensional polyhedron. We follow mainly Kaibel [Kai11b] and Goemans [Goe15].

Definition 2.7. The *nth permutahedron* Π_n is the convex hull of all permutations of the vector $v = \{1,\dots,n\}$

$$\Pi_n = \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = \frac{n(n+1)}{2}, \sum_{i \in S} x_i \leq \sum_{i=1}^{|S|} (n+1-i) \text{ for all } S \subset \{1,\cdots,n\} \right\} .$$

[LW14]

The permutahedron is an $(n-1)$-dimensional polytope with $n!$ vertices. In order to see this, consider the following: for each permutation σ the vector σv lies on the sphere with radius $r = \sum_{i=1}^{n} i^2$, so σv can't be a convex combination of other permutations of v. The number of $(n-k)$-dimensional faces is $k! \cdot S(n,k)$, where $S(n,k)$ is the Stirling number of the second type.
Let $\pi : \mathrm{Mat}_{n\times n}(\mathbb{R}) \to \mathbb{R}^n, \quad A \mapsto Av$. Then, π projects the set of permutation matrices P_n to the vertices of Π_n. Since π is linear we have $\pi(\mathrm{conv}(P_n)) = \mathrm{conv}(\pi(P_n))$.
The permutahedron is a linear projection of the Birkhoff polytope via the map
$p(A)_i = \sum_{j=1}^{n} ja_{ij}$.
Since for every linear objective function vector $c \in \mathbb{R}^n$, we have

$$\max\{\langle c,x\rangle \,|\, x \in \Pi_n\} = \max\{\sum_{i=1}^{n}\sum_{j=1}^{n} jc_i a_{ij} \,|\, A \in \mathbb{P}_n\} \quad , \tag{2.52}$$

one can use the Birkhoff polytope $\mathbb{P}_n$ instead of the permutahedron Π_n in linear programming related issues [Kai11b]. Using a relaxation based on the permutahedron (description requires $2^n - 2$ inequalities) instead of the Birkhoff polytope (prescribed by n^2 nonnegativity inequalities) is computationally infeasible, because of the exponentially many facets (whereas the Birkhoff polytope has n^2 facets). For more information on extended formulations of polytopes in general see [Kai11b] as well as [CCZ10].

3. The Mixed Integer Model

3.1. Motivation

In recent years, the consumer demand for customized products and individual customer requests has been growing extensively. This results in a high product diversity in many areas of economy. The automotive industry is widely affected by this development and provides a good example of high complexity products manufactured in mass customization [LRZ06].

Automobile manufacturers face challenges of several kinds in many areas, such as development, sales, production, logistics. Production related challenges tend to be quite complex in the number of involved parties and restrictions, that have to be met. Simultaneously, handling those challenges well is essential in order to satisfy the demand, meet goals for productivity growth and be competitive through increase in efficiency. The manufacturing process in the automotive industry contains three shops, the body shop, paint shop and the assembly line. Controlling of the assembly line has been in the focus of researchers in operations research and optimization theory. A binding specification of the production sequence is supposed to stabilize production and logistic processes, which ensures predictable costs and production time. This issue called the *Car Sequencing Problem* (CSP) has first been described by Parello et al. in [PKW86]. Since then, extensive research has been done in that field and results can be found in a great amount of publications, see [DKM06; Sol+08; Kis04; FB08; PR08] and references therein.

We consider body shop and paint shop as second and first tier supplier of the final assembly. Currently, both shops neither preserve the body sequence during the passage nor are able to provide the sequence required by the final assembly. In order to secure the supply of the final assembly with the required car body at any time, the paint shop

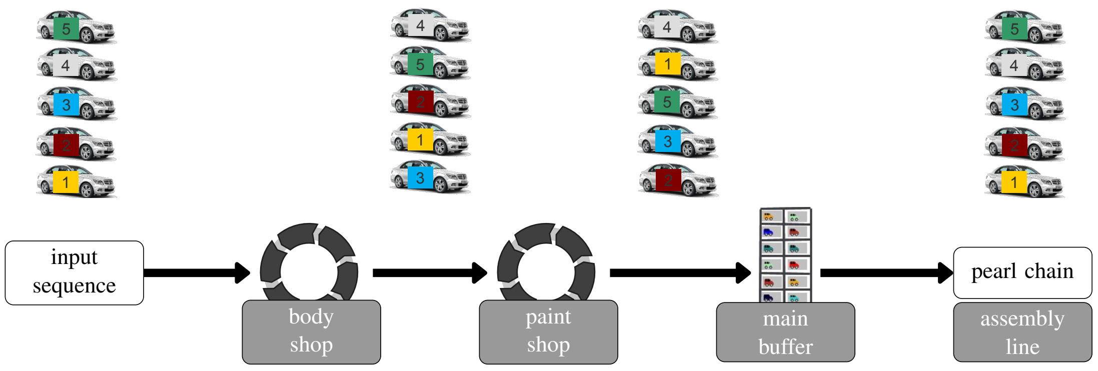


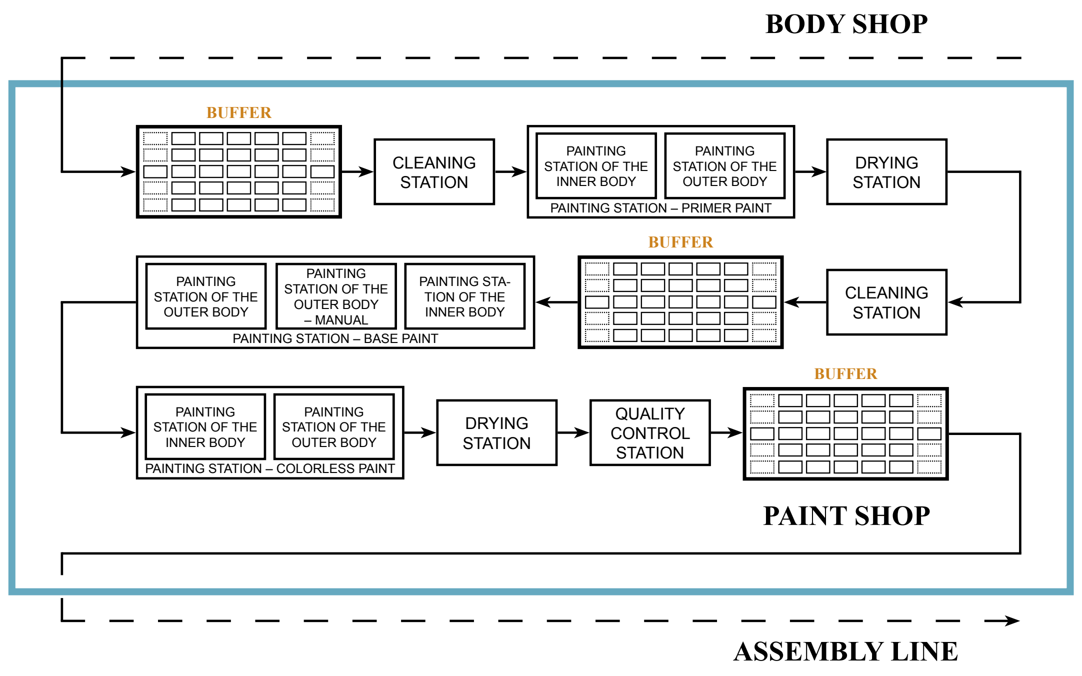
Figure 3.1.: Production chart

charges a high rack warehouse with painted bodies, which functions as storage and sorting place. This so-called *main buffer* assembles the pearl chain sequence for the final assembly. The filling of the main buffer has to be monitored at all times and has to be manually adjusted if necessary. The buffer size has to be checked for sufficiency with every launch of a new model. Costs for construction, support, monitoring and adjustments sum up to a significant amount. In the production line, the paint shop is located between body shop and final assembly (see Figure (3.1)). A car body is assigned to a specific customer order right from the start of manufacturing. Therefore, it holds several order defining attributes, of which the following are relevant in our work: body ID, body type, paint code, model, production number and vehicle type. When entering the paint shop, the car bodies physically just differ in body type, whereas they differ in body type and color when leaving. A combination of body type and color code is called a *Rohbaulack variant*. Car bodies can switch their assigned orders at several stations in the paint shop, if they coincide in body type and color code. We call the possibility to exchange orders *swap option*. Exchanges like that are performed for reasons of different kind such as manufacturing control, supply shortfalls or modification of customer request. In the design of the MIP we respect only the main buffer as a spot with swap options and consider the pairing of car bodies and customer order fixed from manufacturing start to main buffer.

Depending on the question one is interested in, a sequence in the paint shop can be considered a sequence of specified virtual orders or of physical Rohbaulack variants. We justify the latter one by the fact that bodies of the same type and color code can get the assigned customer order exchanged. Unless noted explicitly, a sequence of bodies is usually considered a sequence of orders in the following. A third way is to

understand a sequence as a sequence of orders that get shifted.
We assume that the bodies traverse the paint shop within one day, the set of orders that are processed in the same day is being called a *daily order set*. This definition does not hold in a factory, since there are bodies that remain in the shop beyond the day limit.

For a better understanding of the process and the resulting challenges we take a peek at the interior of the paint shop. A rough draft of the organization and the workflow with focus on relevant details should suffice.
The paint consists of three coats for mat lacquers and four coats for metallic lacquers, that are applied in dipping baths or spraying booths. Since purging painting tools is time and money comsuming, bodies are usually stored in a smaller buffer until a certain amount of one color is accumulated. Besides that, the paint shop might include some other smaller buffers to assure continuity in the production process. See Figure 3.2 for an exemplary structure of a paint shop.


BODY SHOP
BUFFER
CLEANING STATION
PAINTING STATION OF THE INNER BODY
PAINTING STATION OF THE OUTER BODY
PAINTING STATION – PRIMER PAINT
DRYING STATION
BUFFER
CLEANING STATION
PAINTING STATION OF THE OUTER BODY
PAINTING STATION OF THE OUTER BODY – MANUAL
PAINTING STATION OF THE INNER BODY
PAINTING STATION – BASE PAINT
BUFFER
PAINTING STATION OF THE INNER BODY
PAINTING STATION OF THE OUTER BODY
PAINTING STATION – COLORLESS PAINT
DRYING STATION
QUALITY CONTROL STATION
PAINT SHOP
ASSEMBLY LINE

Quelle: [Bys+20, Fig. 3.]

Figure 3.2.: Structure of paint shop.

Different buffer structures can be found among European automotive manufacturers. For an interesting overview and discussion on this issue see [Bys+20].
The permanent need to reduce production costs while meeting manufacturing and consumer related due dates gives raise to the question, if there is a way to control both, body shop and paint shop, in order to minimize the size of the main buffer.

This research focuses on the paint shop, however, the results are adjustable for the body shop. See section (3.7) for a further discussion on necessary adjustments and the possibility to combine two models in order to control both, the body shop and the paint shop.
Between the painting processes there are several stations for quality control, reworking or manual operations. Besides, we have ovens, cooling sections and polishing stations. Moreover, there are other work stations that are not directly painting related, e.g. sealing. Due to the architecture of the paint shop as well as the body characteristics, a sequence of bodies is not preserved during the throughput. The factor that two or more longitudinal conveyors meet one transverse conveyor and the accumulation of bodies in a sorter inside the paint shop are examples of the first type. Regarding the second type we name the fact that not every body requires the same amount of time for reworking and that rarely requested color codes may have more time spent in the buffer in the paint shop.
Many of those factors being non-deterministic challenges the control of the paint shop. Assuming the main impact on the sequence perturbation is carried by properties of the bodies, essentially the paint, we decided to consider the paint shop a black box, that transforms an input sequence into a perturbed output sequence.

3.2. Notation and setup

Originating from the automotive use case decribed above (see Section 3.1) we set the problem in a more general mathematical context. Therefore, it is required to fix some notation. Some of the terms might have been mentioned in the work so far. For the sake of completeness we give rigorous definitions as well as a concrete example in terms of a use case in the following.

Let $\mathcal{A}_1, \dots, \mathcal{A}_m$ denote *properties* and let $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_m\}$ denote the set of those properties. For each property $\mathcal{A}_l$ let there be given a finite number of *characteristics* $\mathcal{A}_l = \{a_l^1, a_l^2, a_l^3, \dots\}$. In the use case of this research we focus on six properties

1. body ID

2. body type

3. color code
4. model
5. production number
6. vehicle type.

The characteristics of every property are given in the form of numerical codes of a specific length.

Let

$$\Gamma : \mathcal{A}_1 \longrightarrow \mathcal{A}_2 \times \cdots \times \mathcal{A}_m \tag{3.1}$$

be a map. We call an element $A = (a_1, \dots, a_m) \in \mathrm{Graph}(\Gamma) \subset \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$ an *order*. In our use case an order is a full specification of a vehicle corresponding to a customer request [1]. An order is characterized by its characteristic in property $\mathcal{A}_1$, respectively the body ID.

Let

$$\tau_l : \mathrm{Graph}(\Gamma) \longrightarrow \mathcal{A}_l \quad , \tag{3.2}$$

$$(a_1, \dots, a_m) \mapsto a_l \tag{3.3}$$

denote the projection on the lth component. On a set of orders we define equivalence classes by the quotient

$$\mathcal{A} \Big/ \tau_{l_1} \times \cdots \times \tau_{l_L} \quad , \tag{3.4}$$

where $\{\tau_{l_1}, \dots \tau_{l_L}\} \subset \{\tau_1, \dots, \tau_m\}$ with $L \in \mathbb{N}$, $L < m$ and

$$\tau_{l_1} \times \cdots \times \tau_{l_L} : \mathrm{Graph}(\Gamma) \longrightarrow \mathcal{A}_{l_1} \times, \dots, \times \mathcal{A}_{l_L} \quad , \tag{3.5}$$

$$(a_1, \dots, a_m) \mapsto (a_{l_1}, \dots, a_{l_L}) \quad . \tag{3.6}$$

Hence, we have

$$(a_1, \dots, a_m) \sim (a'_1, \dots, a'_m) \Longleftrightarrow \tau_{l_i}(a_1, \dots, a_m) = \tau_{l_i}(a'_1, \dots, a'_m) \quad . \tag{3.7}$$

We denote the equivalence class of an order $(a_1, \dots, a_m) = A$ by $[A]$. The projections

[1] At the time of our tracking, e.g entrance and exit of the paint shop, the physical order carries the six properties mentioned above. A completely manufactured one carries far more.

defining the quotient space (3.4) will always be clear from context, so we omit them in the notation of the equivalence classes.

Furthermore, for each fixed set of orders $\{A_1, A_2, \dots\}$ there exists a family of sequences $j_t : \mathbb{N} \longrightarrow \{A_1, A_2, \dots\}$. We denote a sequence by (x^i_j), where i denotes the order A_i and j denotes its position within the sequence.
Let $\left(S, (x^i_j), (y^i_{\tilde{j}})\right)$ be a tuple of a finite set of orders $S = \{A_1, \dots, A_n\}$ and two finite sequences $(x^i_j), (y^i_{\tilde{j}})$ of orders $A_i \in S$. For the length n of a sequence we have $n((x^i_j)) = |S|$. Since the elements are pairwise different a sequence (x^i_j) can also be identified with a permutation in S_n or a permutation matrix $X = (x_{ij}) \in \mathrm{Mat}_{n\times n}(\{0,1\})$ in the natural way.

$$x_{ij} = \begin{cases} 1 \text{ if order } x^i \text{ is at position } j \text{ within the sequence} & , \\ 0 \text{ else.} \end{cases} \tag{3.8}$$

Remark. The identification of a sequence with the permutation or the permutation matrix are 1-to-1 maps

$$\begin{tikzcd} & \text{permutation matrix} \arrow[dl, leftrightarrow, "1\text{-}1"'] \arrow[dr, leftrightarrow, "1\text{-}1"] & \\ \text{permutation} \arrow[rr, leftrightarrow, "1\text{-}1"'] & & \text{sequence} \end{tikzcd}$$

Depending on the mathematical context we prefer one interpretation or the other. For the sake of readability we omit the brackets in the notation from now on and write x^j_i and x_{ij} for a sequence and permutation or matrix, respectively.

We extend the definition of equivalence classes 3.4 on an order set to sequences and analogously to permutations and permutation matrices, respectively. For two sequences x^j_i, y^k_i we define

$$x^j_i \sim y^k_i \Longleftrightarrow x_i \sim y_i, \qquad \text{for all } i = 1, \dots, n \quad . \tag{3.9}$$

The equivalence class of x^j_i is denoted by $[x^j_i]$. Again, we omit the notion of the projections defining the relation, since they are usually clear from context.

In the presented use case let S be a *daily order set*, that is the set of orders which are manufactured in the paint shop on a given day. This number may vary daily, so we consider n arbitrary but fixed during the optimization.[2]
Every order carries two time stamps. Ordering by one provides the *input sequence*, that is the sequence in which the orders enter the paint shop. Ordering by the other time stamp gives us the *output sequence*, in which the orders leave the shop. Table 3.1 shows the format of the use case data.

body ID	**body type**	**paint code**	**model**	**production number**	**vehicle type**	**time In**	**time Out**
1700159	7034	197	22215912	9398284	V222	01.07.2019 06 : 38	01.07.2019 21 : 38
1700164	7036	992	22215612	9398217	W222	01.07.2019 06 : 40	01.07.2019 22 : 03
1700188	7067	197	22216612	9398205	V222	01.07.2019 07 : 49	01.07.2019 20 : 39
1700281	7034	998	22215612	9398090	V222	01.07.2019 06 : 52	01.07.2019 20 : 43

Table 3.1.: Data format of orders

Since every order is assigned to a body from production start, the number of orders equals the number of car bodies.
We differ three kinds of sequences

(i) sequence of physical bodies,

(ii) sequence of virtual customer orders,

(iii) sequence of Rohbaulack variants.

Recall, that an order is assigned to a body at any time but may get interchanged. After distinguishing them we now keep those three perspectives in mind. Where necessary we will explicitly mention the perspective to choose.

Considering the paint shop a black box, we can think of the process therein as a permu-

[2]In production orders not rarely remain in the paint shop beyond one or more daily order sets due to manufacturing reasons such as reworking. We neglect this in our model and assume that all orders which enter the paint shop one day, leave it the same day. In Section 3.7 we discuss the possibilites of adjusting the model for application.

tation of order sequences or as shifting of single orders within position 1 to n, where the corresponding permutation matrix carries stochastic entries. We denote a *shift* forwards within the sequence with negative sign and backwards with positive sign. Let

$$d_i = -(j - \bar{j}) \tag{3.10}$$

be the shift that x^i received during the throughput, where j is its input position and $\bar{j}$ its output position.
The shift of any order passing the paintshop depends on two factors. It is the result of the interactions of the orders nearby as well as the intrinsic behavior of that order itself. Assume, that this behavior can be tightened[3] to a fixed subset of the set of properties. We call this subset *set of defining properties* $\mathcal{A}_{dp}$. Note, that we have $\mathcal{A}_1 \notin \mathcal{A}_{dp}$.

At some point of the thesis we refer to the data from production as *historic input/output sequence* in order to distinguish them from calculated or artificial orders. We call the default output sequence, which is aimed to be reached with high probability and low deviation, *optimal output sequence*. Note, that the term "optimal" refers to production planning in the assembly line, not to the quality of MIP solutions. This optimal output sequence is dictated by the final assembly. In the factory, single orders of the optimal output sequence may be rearranged or put on hold. We neglect this and consider the optimal output sequence to be fixed.

In the use case we set $L = 2$ and $\tau_{l_1} = \tau_2, \tau_{l_2} = \tau_3$. For two orders $a = (a_1, \ldots, a_6)$ and $a' = (a'_1, \ldots, a'_6)$ of a daily order set, we have

$$\begin{aligned} (a_1, \ldots, a_6) \sim (a'_1, \ldots, a'_6) \iff \\ \tau_2(a_1, \ldots, a_6) = \tau_2(a'_1, \ldots, a'_6) \text{ and } \tau_3(a_1, \ldots, a_6) = \tau_3(a'_1, \ldots, a'_6) \quad . \end{aligned} \tag{3.11}$$

Two orders are equivalent, if - and only if - they coincide in body type and color code. We say the orders are of the same Rohbaulack variant.

Determining probability functions

For a fixed set of input sequences $\{x^i_j\}$ let Ω denote the set of possible output sequences, that is the set of permutations of each input sequence. For each order in the

[3]In the sense that probability functions (discussed in Section 3.2) with respect to that set are time-independent and of high regularity.

input sequences let $X : \Omega \longrightarrow \mathbb{Z}$ denote the random variable that describes the shift $X(\omega)$ experienced by the order regarding the input sequence and the event of output sequence ω[4].

Let $\mathcal{A}_{dp} = \{\mathcal{A}_{m_1}, \ldots, \mathcal{A}_{m_m}\} \subset \{\mathcal{A}_2, \ldots, \mathcal{A}_m\}$ be the set of defining properties. For each element $\alpha = (\alpha_{m_1}, \ldots, \alpha_{m_m}) \in \mathcal{A}_{m_1} \times \cdots \times \mathcal{A}_{m_m}$ let P_α denote the probability distribution of shifts for orders of characteristics α

$$P_\alpha : [-n+1, n-1] \cap \mathbb{Z} \longrightarrow [0,1] \tag{3.12}$$
$$k \mapsto P_\alpha(k) = \sum \tilde{P_X}(X = k) \tag{3.13}$$

where the sum over all random variables is corresponding to orders of specification α.

For every pair of input and output sequences, for every order in those sequences determine the shift. Then, for each element $\alpha \in \mathcal{A}_{m_1} \times \cdots \times \mathcal{A}_{m_m}$ the relative frequency distribution can be understood as approximation of the corresponding probability distribution P and set

$$P_\alpha(k) := \frac{|\{\text{orders of characteristics } \alpha \text{ with shift } k\}|}{|\{\text{orders of characteristics } \alpha\}|} \quad . \tag{3.14}$$

Let f_α denote the probability function

$$f_\alpha(x) = \begin{cases} P_\alpha(k) = p_k & \text{if } x = k \in \{-n+1, \ldots, n-1\} \\ 0 & \text{else.} \end{cases} \quad , \tag{3.15}$$

A set of pairs of input and output sequences, for which probability distributions are determined, is being called *profile data set.*

For the sake of readability we now drop the reference to the characteristics in the notation of probability density functions, but refer to the order instead. Write f_i for f_α, if the characteristics of x_i equal $\alpha = \alpha_{m_1}, \ldots \alpha_{m_m}$.
Note, that we then have $f_i = f_j$, if $x^i = x^j$ respective the defining properties, that is $x^j \in [x^i]$.

In the following we consider the color code the only defining property. For the sake of readability and understanding we pass on the general formulation from here on and

[4]In this section we adapt the notation familiar from the field of stochastic theory.

continue in terms of the use case.
Let $\mathcal{C} = \{c_1, \ldots, c_C\}$ denote the set of different colors, that appear in the paint shop[5]. Fix a daily order set $\{x^1, \ldots, x^n\}$ and the associated optimal output sequence $\tilde{y} = \tilde{y}^i_{\tilde{j}}$, set by the assembly line. Let x, y be two sequences consisting of the orders in that daily order set. Consider the quotient $\{x^1, \ldots, x^n\} \big/ \pi_3$. Let l_h denote the size of the equivalence class given by $\pi_3^{-1}(h)$ for $h = 1, \ldots, C$. Let $s_{h,k}(x,y)$ be the number of orders of color h and shift k from sequence x to sequence y.
Since it will either be clear from the context, to which two sequences the shift refers, or mentioned explicitly, we just write $s_{h,k}$. With this notation, the quotient $\frac{s_{h,k}}{l_h}$ denotes the relative frequency of a shift by k positions among orders of color h, for $k = -n, \ldots, n$ and $h = 1, \ldots, C$.

Combinatorics

With regard to the design of the MIP (see Section 3.5) and the evaluation of the results, some combinatoric observations are required. For a set of n orders there are

$$n! \tag{3.16}$$

possible sequences. On the set of equivalence classes $\mathcal{A} \big/ \tau_{l_1} \times \cdots \times \tau_{l_L}$ we have

$$\frac{n!}{\prod_i |[A]_i|!} \tag{3.17}$$

sequences, where the product $\prod_i |[A]_i|$ is over all equivalent classes of orders. For the size of an equivalence class $[x]$ of sequences we have

$$|[x]| = \prod_i |[A]_i| \quad . \tag{3.18}$$

Terms for best-case scenario studies

The structure of the problem described above stands out by its high complexity and the strong dependency of the behavior of the orders within a sequence. Thus, it suggests itself for a best-case scenario analysis, where the data is of very high regularity that

[5]In the plant considered in this work we have $|\mathcal{C}| \approx 40$.

permits estimations on lower bounds of the computational time and upper bound of the quality of results. For both of the approaches, the MIP and the neural network, we specify the following terms.
We set $\mathcal{A}_{dp} = \{\mathcal{A}_3\} = \{\text{paint code}\}$ for the set of defining properties, and consider data sets of sequences satisfying $|\{\mathcal{A}_3\}| = 2$, $|\{\mathcal{A}_3\}| = 4$ and $|\{\mathcal{A}_3\}| = 8$ respectively. Let n_C denote the size of a data set consisting of pairs of input and output sequences, where n denotes the sequence length and $C = |\{\mathcal{A}_3\}|$. We set

$$n_C = 10_2, 15_2, 20_2, 30_2, 10_4, 15_4, 20_4, 30_4, 40_4, 50_4, 30_8, 40_8, 50_8 \quad . \tag{3.19}$$

The following terms hold.

(i) For each data set there exists one distribution for the set of defining properties $\mathcal{A}_3$ such that for each sequence in the data set the orders of the sequence fulfill the distribution[6].

(ii) The probability density functions are time-independent.

(iii) Calculations run on sequences of exactly one data set.

In a second step, in order to investigate the behavior of calculation time and the quality of results, we run further calculations on the terms

(i) Bullet points (i) and (ii) from above

(ii) Profile data are taken from more than one data set, all of the same size.

For details of the neural network data see Section 5.3.2.

3.3. Analysis of real and artificial data

The use case in this work is provided by the Daimler AG. The description of the setup and the processes as well as the sets of real data relate to a particular plant in Sindelfingen. The remaining part of this chapter gives an analysis of real data and the reasons for using artificial data instead. An algorithm for data generation is presented and the

[6]For further explanation see Table 3.3 as well as the whole section about the generation of artificial data (Section 3.3.2)

results are discussed.
The assumption that the setup and conditions, under which the paint shop is run, don't change over time provides us with a significant number of pairs of input and output sequences. However, note, that we are not able to manipulate the input sequence and, particularly, have exactly one realization of the experiment in form of the historic output sequence.

3.3.1. Analysis of real data

In order to justify the interpretation of the throughput of a sequence of car bodies through the paint shop as an experiment in probability theory, it is required to have a recognizable regularity for each color code or Rohbaulack variant, that are assumed to be the only elements of the set of defining properties, $\mathcal{A}_{dp} = \{\text{paint code}\}$ or $\mathcal{A}_{dp} = \{\text{body type, paint code}\}$. For a probability distribution P_α we analyze statistic values up to the third moment. Mean, variance and skewness are determined for several subsets of the data to be analyzed according to the method of moving values. Therefore, we fix two natural numbers *shift* σ and *range* ρ, with $\sigma < \rho$. For each element $\alpha \in \mathcal{A}_{m_1} \times \cdots \times \mathcal{A}_{m_m}$ get the subsequence of orders of these characteristics. For each of those subsequences we iteratively determine subsequences of length ρ, where the σth order of the νth subsequence is set to be the first order of the $(\nu+1)$th subsequence until the end of the sequence. Mean, variance and skewness are calculated for each of these subsequences. Note, that for each order set ρ and σ are fixed and independent of the number of orders for each color code. This leads to different numbers of subsequences but offers potential for comparison.
The paint shop is run in a three-shift operation from 0:00 to 24:00. This leads to the following approach in the data analysis. We fix a continuous period of time and get the historic output sequence for each day in this time period as the set of orders, that left the paint shop on that day. Ordering those orders by the date of entry is considered the corresponding input sequence. Note, that this procedure ignores orders of the same entry day but an exit day beyond the fixed time period.

For the analysis of real data we request order sets of cardinality 10037, 29994, 100305[7]. In terms of cardinality those sets approximately meet the artificial data sets of sequence length $n = 10, 30, 100$, where in each case the data analysis is based on sets of 1000

[7]On a sidenote: These data sets correspond to the workload of about $7, 20$ and 70 working days, respectively.

sequences. The following table (Table 3.2) shows the values for shift and range.

order set	size	range ρ	shift σ
HD-1	10037	200	50
HD-2	29994	300	200
HD-3	100305	500	400

Table 3.2.: Real data analysis, range and shift values

In choosing those values we balanced between having a statistically significant number of orders per subset and a representative number of subsets for each color code. Despite the fact that further analysis[8] did not show any indication of relations between shift and position of the order within the sequence, the orders are shuffled for the computation of the statistical values to ensure that even hidden effects may not fall into account.

The diagrams in Section A.1 show mean and variance values[9] for four of the most common color codes. The values are rounded up to two decimal points.
Since the skewness values are in the range of 10^{-7} to 10^{-9} and thus the distributions can be considered symmetric, we forego the presentation of those.
Focussing on the second data set HD-2 consisting of 29994 orders we determine great variations in the mean values up to 277 positions over all subsequences (see Figure A.2a). For a reference, this corresponds to 0.16 to 0.21 of the total number of orders per sequence. All of the variance values consist of five or six figures (Figure A.2b). By taking into account the results of both of the other data sets, we conclude that the greater the data set the greater the variations in mean and variance values and thus the less stable the behavior of the observed system.
The results of color codes, that are requested more rarely[10], are even worse regarding stability. The main reasons for this include the fact that these car bodies tend to remain

[8]In the scope of an extensive analysis the data was investigated for the indication of several dependencies, such as 'position within sequence - shift' and 'color code of predecessor/successor - shift' and 'Rohbaulack variant of predecessor/successor - shift'. None of these analyses showed any reliable results. The main reasons may be found in the great number of different factors that cause the shift of orders and the round-the-clock operation of the paint shop. In a manufactoring view the latter impedes a division of the flow of car bodies into the individual sequences. We leave it at this footnote and forgo further specifications.

[9]Due to the different numbers of orders for each color code we get different numbers of subsequences in the procedure of calculating moving statistical values. Furthermore we bounded the number of displayed subsequences to 22 for real data and to 30 for artificial data

[10]We omit a detailed presentation in Appendix A.1

longer in the small buffer, where they are gathered before the topcoat is applied. And secondly, bodies of rare color codes are rather affected by manually control since they have less swap options in the main buffer.

We summarize that the analysis has shown that the shift of the car bodies passing the paint shop can not be attributed to one property or a combination of several properties. There might be an underlying regularity under a huge noise. On the one hand, manual operations might have a great influence, on the other hand, there are factors such as temperature, humidity, air pressure, substrate composition, that can have an impact on the painting quality and that can not be tightened to specific properties of the bodies. This lack of regularity is one reason why we felt the need to generate artificial data.

3.3.2. Algorithm for generating artificial data

In order to perform best-case scenario studies on the Mixed Integer Program (Section 3.5) as well as on the neural network (Section 5.3.1), we designed an algorithm to generate artificial data. We are interested in the dependence of the computational time and the quality of results from

(i) the overall length of the sequence

(ii) the number of orders of specific properties (e.g. different paints)

(iii) the support sizes of the probability functions.

Since the shift of any individual order is highly dependent on the behavior of the orders in a neighborhood, we cannot get sequences meeting our desired properties by cutting off daily sequences of real data. Also, it is not possible to design input sequences that run through production. In addition, the cardinal number of the daily order set varies by around 7 to 9%, fluctuations up to 20% are possible. For computational studies this is an improper situation. On this account we generated artificial input and output sequences of any desired kind.

We claim, that this algorithm (Algorithm 1) presented in detail in the following is the *most deterministic way* of generating data that meet the requirements. Since determinism is measured by the Dirac measure, the phrasing of this claim calls for further explanation. For each tuple of characteristics α, in this case $\alpha \in \mathcal{A}_3 = \{\text{color codes}\}$, we fix an integer m_α. For different data sets let $|\{\text{color codes}\}| = 2, 4, 8$. The color-specific probability functions describe the behavior of the orders which solely results from the shift by m_α positions as well as the interaction of the orders. In calculating the sequences, the algorithm does not take into account other factors. In this sense we consider this method the "most deterministic way" of creating pairs of sequences, where the shifts of the orders follow probability distributions.

Let n_C be a sequence of n orders and C different color codes among these orders. We take a suitable set of real orders[11]. Generate a number $N \leq n!$ of different input sequences by arranging the orders randomly and sorting out duplicate sequences. The following table (Table 3.3) shows the color codes and the corresponding shifts m_α as well as the distributions of colors for the first data set of each specification. In total, we generate sixteen data sets constisting of 1000 sequences for each assignment of n and C (see (3.19))[12]. All of these data sets obey different distributions of colors. The data analysis as well as calculations for the MIP and the neural network are perfomed first on data sets, where the orders of each sequence follow one paint code distribution, namely those shown in Table 3.3, column four to six. We call data sets of this kind *mono-distributed* and denote them by n_C-1. The $\imath$th sequence in this set is denoted by n_C-1-$\imath$. In a second step, the same computations are performed on data sets where the sequences are taken out of the sixteen sets mentioned above in equal parts. We call these data sets *poly-distributed*. We write n_C-2 and n_C-2-$\imath$ for the data set and the $\imath$th sequence, respectively.

[11] We remark that the body types and thus Rohbaulack variants - in contrast to number and distribution of color codes - are not specified here. Rohbaulack variants are not relevant for the data analysis but in the evaluation of MIP results. Therefore, we postpone further comments on number and distribution of Rohbaulack variants to Section 3.7.

[12] Except for $n_C = 10_2$ and $n_C = 15_2$ where it is not possible to fix sixteen different color code distributions for combinatorial reasons.

length n	**paint code**	**shift m_α**	**distribution in n_2**	**distribution in n_4**	**distribution in n_8**
$0 \leq n \leq 30$	197	-3	0.8	0.35	0.25
	775	2	0.2	0.30	0.22
	149	-4		0.20	0.10
	897	1		0.15	0.05
	799	6			0.12
	183	4			0.08
	831	-4			0.07
	992	2			0.11
$30 < n \leq 50$	197	-8	0.8	0.35	0.25
	775	2	0.2	0.30	0.22
	149	-4		0.20	0.10
	897	10		0.15	0.05
	799	6			0.12
	183	4			0.08
	831	-4			0.07
	992	2			0.11
$50 < n \leq 100$	197	-15	0.8	0.35	0.25
	775	11	0.2	0.30	0.22
	149	-9		0.20	0.10
	897	21		0.15	0.05
	799	6			0.12
	183	4			0.08
	831	-4			0.07
	992	2			0.11

Table 3.3.: Color-specific shift and distribution of colors

We summarize that the input sequences in each data set are permutations of each other[13]. For each input sequence, for each order of the sequence, calculate the *fictional* output position $j + m_\alpha$ by adding the specific shift m_α to the input position j of the order. In doing so, there might occur positions < 1 or $> n$ and some positions might be occupied by more than one order whereas others might not be occupied at all. Sorting the orders by that fictional position in ascending order gives us the actual

[13]We keep this fact in mind, it will also be vitally important in the evaluation of the MIP results.

artificial output sequence (see Algorithm 1).

Algorithm 1: Artificial data generation

```
Input: order pool;
$n$ - sequence length;
$C$ - number of colors;
$N$ - data set size;
$m_\alpha$ - shift;
distribution for $n_C$;
initialize data set $d$;
initialize $\hat{x}$ order sequence;
Function GenerateBasicSequence($n$, $C$, order pool):
    fill $\hat{x}$ with orders of the order pool such that $\hat{x}$ satisfies $n_C$;
return
for $1 \leq v \leq N$ do
    $x \leftarrow$ randomPermutation($\hat{x}$);
    for $1 \leq j \leq n$ do
        set fictional output position of $x_j$ as $j + m_\alpha$;
    end
    $y \leftarrow x$ ordered by fictional output position in ascending order;
    d.Add($x$,$y$);
end
return $d$;
```

See Table 3.4 for an exemplary realization on a set of ten orders.

3.3.3. Analysis of artificial data

We perform the same analysis on the artificial data as on the historic sequences, ignoring the fact that we actually know the underlying deterministic shifts m_α. Analogous to the analysis of the real data, statistical values up to the third moment as well as probability functions (for a reminder see Section (3.2)) are determined. Table 3.5 shows the values for shift σ and range ρ.

paint code	775	197	197	149	197	775	897	775	197	149
input position	1	2	3	4	5	6	7	8	9	10
fictional output position	$1+2=3$	$2-3=-1$	$3-3=0$	$4-4=0$	$5-3=2$	$6+2=8$	$7+1=8$	$8+2=10$	$9-3=6$	$10-4=6$
output position	5	1	2	3	4	9	8	10	6	7

Table 3.4.: Example data generation

order set	**range ρ**	**shift σ**
10_4	300	200
50_4	400	300
100_8	400	300
200_8	400	300

Table 3.5.: Artificial data analysis, range and shift values

In order to perform a best-case scenario analysis, we strive for data of stability and regularity as high as possible. Section A.2.1 displays the moving statistical values for the mono-distributed sets $n_C - i = 10_4$-1, 50_4-1, 100_8-1, 200_8-1 rounded up to two decimal points. Across all data sets and all color codes we have very stable mean values. The statistical values of the second moment increase with growing sequence length in a moderate way. Comparing 10_4-1 to 50_4-1 and 100_8-1 to 200_8-1 we note, that the variance values decrease. However, those are of very little significance due to the overall order of magnitude.
Section A.2.2 shows the results on poly-distributed sets 10_4-2, 50_4-2, 100_8-2, 200_8-2. In comparison to values for mono-distributed sets, we obtain greater variations for the mean values and thus greater variance values. These analyses give an insight into the strong dependencies of the orders amongst each other and the effect on each others shift. We take both of them being in a range, so that the data can still be considered sufficiently good. The skewness values are again insignificantly small, so that the distributions can be considered symmetric.
Probability distributions are determined for the same eight data sets. Find them attached in Section A.2.3.
As the statistical values let us expect we obtain time-independent probability distributions. They consistently can be approximated by a Gaussian distribution. We note, that for $n \geq 50$ the graphs show a significant difference for mono-distributed and poly-distributed data sets. The presented probability functions referring to the latter sets show, what we might call in this context, a softening. Different color distributions for the sequences lead to greater support for the probability distribution of each color. Despite the fact that the underlying shifts are deterministic, the interaction of the orders is affected just by varying the distributions of colors.

3.3.4. Discussion of the artificial data

The great variations in mean and variance values even for frequent color codes indicate that the perturbation of order sequences passing the paint shop cannot be tightened to the paint of the car bodies. Further analyses did not yield a better choice of the set of defining properties. Perturbation factors depending on the order position within a sequence or on properties of predecessor/successor can not be identified as well. Thus, the real data leaves us with two conclusions: Firstly, they do not meet our requirements on regularity and thus are improper for extensive testing of the Mixed Integer Model, that is based on this very regularity. Secondly, on the application side there is the concern that the sheer number of perturbating factors does not allow controlling the paint shop as a whole. Additionally, the inflexibility - as far as in particular the sequence length is concerned - made it necessary to discard the real data.

The algorithm (Algorithm 1) provides us with some comfortable degrees of freedom, such as the choice of sequence length, number of color codes, et cetera. By taking deterministic shifts as a basis, the dispersion of shifts is the result of the interaction of the orders solely. This leads to eminently stable moving mean and variance values and time-independent color-specific probability functions.
As a downside of this approach we only have one realization for each experiment, i.e. one output sequence for each input sequence. However, we assume that this is the case in many applications in the industries, since the orders may often correspond to customer demands and cannot be exchanged or arranged arbitrarily. Thus, with use cases in mind, the choice of this approach seems legitimate.
While the statistical values and the probability distributions show the desired extent of stability, poly-distributed data sets make the dependencies apparent. The effect of varying color code distributions within a data set on mean and variance values justify the term of best case studies for mono-distributed data sets. They may indeed be considered the best conditions the Mixed Integer Model can be tested for. We conclude that Algorithm 1 generates artificial data satisfying the requirements on stability.

3.4. Measure on the set of sequences

There are two candidates suggesting themselves for the choice of a measure, depending which identification we use. On the set of sequences of the same length we have the

Hamming distance as a natural measure.

Definition 3.1. The *Hamming distance d_H* measures the number of substitutions that are required to transform one string into the other [Mac03, page 206].

Example. The Hamming distance d_H between the two strings

$$\begin{gathered}11\mathbf{0}1011\mathbf{1}0\\ 11\mathbf{1}1011\mathbf{0}0\end{gathered}$$

is 2.

On the symmetric group S_n we have the Cayley distance.

Definition 3.2. The *Cayley distance d_C* is the minimum number of transpositions needed to transform one permutation into another

$$d_C(\sigma,\pi) = \min\{n \big| \sigma\tau_1 \cdots \tau_n = \pi,\ \tau_i \text{ transpositions}\} \quad . \tag{3.20}$$

It is well-known that the Cayley distance is a metric in S_n [LAR12].

Both these measures do not meet all of our requirements. We need a measure that respects the number of permutations and the shift length of the orders.

Definition 3.3. Let x_j^i and x_k^i denote two sequences of one daily order set. We define the distance d between two sequences

$$\delta(x_j^i, x_k^i) = \sum_{i=1}^{n} |j-k| \quad . \tag{3.21}$$

We define a second measure on the equivalence classes of Rohbaulack variants

$$d(x_j^i, x_k^i) = \sum_{i=1}^{n} |j-\tilde{k}| \quad , \tag{3.22}$$

where

$$\begin{aligned}\tilde{k} = \min\{m \mid \; & \tau_2(x_j^i) = \tau_2(x_m^k) \text{ and } \tau_3(x_j^i) = \tau_3(x_m^k) \\ & \text{and } x_m^k \text{ is not assigned to an order yet}\} \\ = \min\{m \mid \; & x_j^i = x_m^k \text{ in color and body type} \\ & \text{and } x_m^k \text{ is not assigned to an order yet}\} \quad .\end{aligned} \tag{3.23}$$

Note, that we can identify the following relationship between δ and d by using the equivalence relation (3.4)

$$d([x_j^i],[x_k^i]) = \min\{\delta(x_j^i,\tilde{x}_k^i) \,|\, \tilde{x}_k^i \in [x_k^i]\} \quad . \tag{3.24}$$

Recall, that every sequence x_j^i can be identified with a permutation matrix A in $\mathrm{Mat}_{n\times n}(\{0,1\})$. Let v denote the vector $(1,\cdots,n) \in \mathbb{R}^n$. Then, for δ and d we have

$$\delta(A,B) = \sum_{i=1}^{n} -(A_i v - B_i v) \tag{3.25}$$

and

$$d(A,B) = \min\{\delta(A,\tilde{B}) \,|\, \tilde{B} \in [B]\} \quad . \tag{3.26}$$

For a fixed sequence length n, the distances δ and d are metrics. Identity of indiscernibles and symmetry are obviously fulfilled. In order to prove the triangle inequality let x_j^i, x_k^i, x_l^i be sequences. We note, that for each summand we have for $\tilde{l} \in \mathbb{R}_+$

$$|j-k| = |j-\tilde{l}+\tilde{l}-k| \leq |j-\tilde{l}|+|\tilde{l}-k| \quad . \tag{3.27}$$

This holds in particular for $\tilde{l}=l$. Summing over all orders in the sequence proves the triangle inequality. With identity (3.24) it follows that d also fulfills the triangle inequality.

Remark. Both of the Definitions 3.21 and 3.22 are motivated by the operating principle of the main buffer. In a simplified scheme, each painted car body is stored in the main buffer until the retrieval by the assembly line. Definition 3.22 respects the possibility of order swaps. We assume that a buffer works with a first-in-first-out principle within each equivalence class of Rohbaulack variants. Thus, an order, that is placed on the correct position considering its color and body type still makes a positive contribution to the distance, if there is an order of the same Rockbaulack variant in the buffer already.

3.5. Mixed Integer Model

3.5.1. Idea

Information about the behavior of the paint shop can be found accumulated in the color-specific probability functions as well as in the mean values of shifts. The information contained in the color-specific probability functions may be considered more detailed, whereas the mean values offer the possibility to respect the order sequence in a specific input sequence.

We consider the following procedure a simulation of the paint shop: for a given profile data set determine the mean value of shifts for each color code. Let x be a suitable input sequence, for each order in x let the *fictional output position* be the sum of the input position and the mean value corresponding to the color code of the order. The *simulated output sequence* y is determined by sorting the orders by their fictional output positions in ascending order[14].

We combine the power of the probability functions with the flexibility of the mean values of shifts in the design of the Mixed Integer Model determining an optimal input sequence.

The objective function minimizes the distance (see (3.26)) of a corresponding simulated output sequence y to the given optimal output sequence $\tilde{y}$ over all possible input sequences under the condition that the relative shifts $\frac{s_{h,k}}{l_h}$, regarding an input sequence x and the simulated output sequence y, minimize the distance to the corresponding probability function pointwise

$$\min \sum_h \sum_k \left| \frac{s_{h,k}}{l_h} - f_h(k) \right| \quad . \tag{3.28}$$

3.5.2. Mathematic formulation of the MIP

This section is dedicated to the formulation of a MIP to determine an optimal input sequence for the use case presented in Section 3.1. Table 3.6 gives an overview of the MIP variables.

[14]This procedure is quite similar to the one in the artificial data algorithm with a key factor that the orders are shifted by the mean value in which the interaction of orders is accumulated.

input		
$\tilde{y}_{ig}$	optimal output sequence	$i,g=1,\dots,n$
decision variables		
x_{ij}	input sequence	$i,j=1,\dots,n$
binary variables		
${}^{x_i}b_l{}^m$	auxiliary variable	$l,m=1,\dots,\vert[x_i]\vert$
r_{ig}	auxiliary variable	$i,g=1,\dots,n$
integer variables		
z_g	number of orders whose position is smaller than p_g	$g=1,\dots,n$
p_i	position of order x_i after shifting by mean	$i=1,\dots,n$
notation		
τ	$\tau=\tau_2\times\tau_3$ projection on Rohbaulack variant	
$\vert[x_i]\vert$	cardinal number of $[x_i]\in\tilde{y}/\tau$	$i=1,\dots,n$
$t_l^{\vert[x_i]\vert}$	position of the lth order of Rohbaulack variant $\tau(x_i)$ in $\tilde{y}$	$l=1,\dots,\vert[x_i]\vert,\ i=1,\dots,n$
constraint $\mathbf{F}_{\tilde{\mathbf{y}}}\leq\mathbf{F}^*+\varepsilon$		
$s_{h,k}$	number of orders of color h and shift k[15]	$h=1,\dots,C,\ k=-n+1,\dots,n-1$
l_h	number of orders of color h	$h=1,\dots,C$
L_{hi}	slack variable	$h=1,\dots,C,\ i=1,\dots,n$

Table 3.6.: Variables in MIP (3.29a)-(3.29m)

[15] regarding optimal output sequence $\tilde{y}_{ig}$ and x_{ij}

The MIP is given by

$$\min_{x} \omega(x) = \min_{x_{ij}} \sum_{i=1}^{n} \left| z_i - \left(\sum_{l=1}^{|[x_i]|} {}^{x_i}b_l{}^m * t_l^{|[x_i]|} \right) \right| \tag{3.29a}$$

$$\text{s.t} \quad \sum_{l=1}^{|[x_i]|} {}^{x_i}b_l{}^m = 1 \qquad \text{for all } m = 1,\dots,|[x_i]| \tag{3.29b}$$

for all equivalence classes in $\tilde{y}/\tau$

$$\sum_{m=1}^{|[x_i]|} {}^{x_i}b_l{}^m = 1 \qquad \text{for all } l = 1,\dots,|[x_i]|, \tag{3.29c}$$

for all equivalence classes in $\tilde{y}/\tau$

$${}^{x_i}b_l{}^m \in \{0,1\} \tag{3.29d}$$

$$z_g = \sum_{i=1}^{n} r_{ig} - 1 \qquad \text{for all } g = 1,\dots,n \tag{3.29e}$$

$$p_g - p_i \leq r_{ig}M \qquad \text{for all } i,g = 1,\dots,n \tag{3.29f}$$

$$-(p_g - p_i) \leq (1 - r_{ig})M \qquad \text{for all } i,g = 1,\dots,n \tag{3.29g}$$

$$r_{ig} \in \{0,1\} \qquad \text{for all } i,g = 1,\dots,n \tag{3.29h}$$

$$p_i = \sum_{j=1}^{n} x_{ij} * j + \mu_i \qquad \text{for all } i = 1,\dots,n \tag{3.29i}$$

$$F_{\tilde{y}}(x) \leq F^* + \varepsilon \tag{3.29j}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for all } j = 1,\dots,n \tag{3.29k}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for all } i = 1,\dots,n \tag{3.29l}$$

$$x_{ij} \in \{0,1\} \tag{3.29m}$$

where F^* denotes the objective value of an optimal solution of the mixed integer program

$$\min_x F_{\tilde{y}}(x) = \min_{x_{ij}} \sum_{h=1}^{C} \sum_{\substack{k=-n+1 \\ 1 \le i+k \le n}}^{n-1} \left| \frac{s_{h,k}}{l_h} - f_h(k) \right| \tag{3.30a}$$

$$\text{s.t} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for all } j = 1, \dots, n \tag{3.30b}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for all } i = 1, \dots, n \tag{3.30c}$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } i, j = 1, \dots, n \tag{3.30d}$$

Number of variables for MIP (3.29a)-(3.29m)[16]:

$$\begin{array}{l} 2n^2 + 2n + \mathcal{O}(n^2) \quad , \\ \quad \text{of which } 2n^2 + \mathcal{O}(n^2) \text{ binary and } 2n \text{ integer.} \end{array} \tag{3.31}$$

Number of variables for MIP (3.30a)-(3.30d)[17]:

$$\begin{array}{l} n^2 + \mathcal{O}(2n-1) \quad , \\ \quad \text{of which } n^2 \text{ binary and } \mathcal{O}(2n-1) \text{ continuous.} \end{array} \tag{3.32}$$

Remark. Without loss of generality assume that the optimal output sequence $\tilde{y}_{ij}$ provided by the final assembly can be identified with the identity matrix $E(n)$. This can always be achieved by a suitable choice of the is. The programmatic formulation dif-

[16]Let $C \in \mathbb{N}$, $C := \max_i \{ |[x]_i| \mid [x]_i \in {\{x_i\}}/_{\tau} \} \le n$ and let $\tilde{C} \in \mathbb{N}$, $\tilde{C} \le n$ denote the number of equivalence classes in ${\{x_i\}}/_{\tau}$. Then, we have $2n^2 + 2n + \tilde{C}C^2$ as upper bound on the number of variables.

[17]Consider formulation (3.33a)-(3.33f) in order to see that. Let D denote the number of equivalence classes in ${\{x_i\}}/_{\tau_3}$, the number of shifts within a sequence is $2n-1$.

fers from the mathematic formulation mainly for problem (3.30a)-(3.30d).

$$\min_{x} F_{\tilde{y}}(x) = \min_{x_{ij}} \sum_{h=1}^{C} \sum_{\substack{-n+1\le k\le n-1\\ 1\le i+k\le n}} L_{hk} \tag{3.33a}$$

$$\text{s.t} \quad L_{hk} \le \left(\frac{\sum_{\substack{\tau_3(x_i)=h\\ \tau_3(y_i)=h}} x_{i(i+k)}}{\tilde{y}_{ii}} - f_h(k) \right) \qquad \text{for all } h=1,\dots,C, i=1,\dots,n \tag{3.33b}$$

$$L_{hk} \ge -\left(\frac{\sum_{\substack{\tau_3(x_i)=h\\ \tau_3(y_i)=h}} x_{i(i+k)}}{\tilde{y}_{ii}} - f_h(k) \right) \qquad \text{for all } h=1,\dots,C, i=1,\dots,n \tag{3.33c}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for all } j=1,\dots,n \tag{3.33d}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for all } i=1,\dots,n \tag{3.33e}$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } i,j=1,\dots,n \tag{3.33f}$$

Remark. Forgetting about constraint (3.29j) we have

$$\omega : S_n \longrightarrow {S_n}\big/{\tau} \longrightarrow \mathbb{N} \tag{3.34}$$

and thus the search space is the Birkhoff polytope $\mathbb{B}_n$.

The summand ε in constraint (3.29j) can be interpreted as weight. The smaller the value of ε, the more weight on meeting the color-specific probability distributions (3.29j) compared to minimizing the distance between the simulated output sequence and the optimal output sequence (3.29a).

We usually refer to MIP (3.29a)- (3.29m) as *the MIP* in this work.

Remark. Note, that the fictional output positions do not necessarily coincide with the positions within a corresponding unique output sequence (where every position is occupied by exactly one order). For reasons of saving on variables as well as the boundedness of the optimization problem ($\le$-constraints), we choose to work with output positions determined as number of orders that have a lower value for $p_i = \sum_{j=0}^{n} x_{ij} j + \mu_i$

(see constraint (3.29i)). This may lead to $p_i = p_{\tilde{i}}$ for $i \neq \tilde{i}$. The calculation of the input sequence is not affected by this nonunique positions. However, note, that it may lead to objective values of the MIP that can not be interpreted as the distance between calculated output sequence and optimal output sequence readily, such as an objective value equal to 1.

3.5.3. Stability analysis of the MIP

It is not trivial to check the results of the MIP for quality due to its non-deterministic character. Since we have just one realization for each input sequence we are not able to check for distribution of output sequences. However, there are required conditions that can be investigated, such as stability of the results.
Roughly spoken for two "similar" optimal output sequences we expect the input sequences calculated by the model also to be "similar". For an optimal output sequence y^1 let x^1 denote the corresponding optimal input sequence. Then, we swap the positions of M orders and get the perturbed optimal output sequence y^2. Let x^2 denote the corresponding input sequence, respectively. Let δ be the measure defined in (3.21) and let $\zeta_{x^1x^2}$ denote the number of orders that got swapped permuting x^1 into x^2. The dependence of the probability function on the color of the orders motivates the investigation of color-internal and cross-color swaps. We examine sequences of up to 50 orders for reasons of calculation time.

Instance	**M**	**Swap color-internal**	$(\delta(\mathbf{y^1},\mathbf{y^2}),\zeta_{\mathbf{y^1y^2}})$	$(\delta(\mathbf{x^1},\mathbf{x^2}),\zeta_{\mathbf{x^1x^2}})$	$\Delta = \frac{\delta(\mathbf{x^1},\mathbf{x^2})}{\zeta_{\mathbf{x^1x^2}}}$
10_2	2	yes	(2,2)	(2,2)	1
	2	no	(4,2)	(4,2)	2
	3	yes	(4,3)	(4,3)	$\frac{4}{3} = 1.\overline{3}$
	3	no	(4,3)	(4,3)	$\frac{4}{3} = 1.\overline{3}$
10_4	2	yes	(2,2)	(4,2)	2
	2	no	(6,2)	(10,4)	$\frac{5}{2} = 2.5$
	3	no	(6,3)	(10,8)	$\frac{5}{4} = 1.25$
20_4	2	yes	(6,2)	(10,2)	5
	2	no	(6,2)	(16,12)	$\frac{4}{3} = 1.\overline{3}$
	3	no	(6,3)	(6,3)	2
20_8	2	yes	(4,2)	(6,2)	3
	2	no	(4,2)	(24,14)	$\frac{12}{7} \approx 1.71$
	3	no	(6,3)	(12,9)	$\frac{4}{3} = 1.\overline{3}$
50_4	2	yes	(4,2)	(12,9)	$\frac{4}{3} = 1.\overline{3}$
	2	no	(6,2)	(28,21)	$\frac{4}{3} = 1.\overline{3}$
	4	no	(12,4)	(30,17)	$\frac{30}{17} \approx 1.73$
50_8	2	yes	(4,2)	(2,2)	1
	2	no	(6,2)	(28,15)	$\frac{28}{15} = 1.8\overline{6}$
	4	no	(10,4)	(42,28)	$\frac{3}{2} = 1.5$

Table 3.7.: Results of the stability analysis for MIP (3.29a)-(3.29m)

Table 3.7 shows that the MIP is stable in the sense that small changes in the optimal output sequences cause small changes in the calculated input sequences, where we understand small changes in the optimal output sequence as a relative shift per order $\Delta < 3$ independently of the sequence length n. In the generation of perturbed optimal output sequences we achieved the property $\Delta < 3$ by shifting few orders by few positions (by an absolute value of 2 to 4 positions).

We have $\delta(x^1,x^2) \geq \delta(y^1,y^2)$ in most and $\zeta_{x^1x^2} \geq \zeta_{y^1y^2}$ in all of the cases. The instances 50_4 and 50_8 show higher values for $\delta(x^1,x^2)$ with no color-internal swap. However, the relative shift per order Δ is < 2 for all calculations, so that we conclude, that the MIP is stable in the sense defined above. There is no detectable difference between color-internal and cross-color swaps.

3.6. Computational results of the MIP

Find extensive computational results of the MIP attached in Appendix B. We run the MIP on mono-distributed and poly-distributed data sets

$$10_2\text{-}1,\ 10_2\text{-}2,\ 10_4\text{-}1,\ 10_4\text{-}2,\ 15_2\text{-}1,\ 15_2\text{-}2,\ 15_4\text{-}1,\ 15_4\text{-}2,$$
$$20_2\text{-}1,\ 20_2\text{-}2,\ 20_4\text{-}1,\ 20_4\text{-}2,\ 30_2\text{-}1,\ 30_2\text{-}2,\ 30_4\text{-}1,\ 30_4\text{-}2,\ 30_8\text{-}1,\ 30_8\text{-}2,$$
$$40_4\text{-}1,\ 40_4\text{-}2,\ 40_8\text{-}1,\ 40_8\text{-}2, 50_4\text{-}1,\ 50_4\text{-}2,\ 50_8\text{-}1,\ 50_8\text{-}2$$

For each instance the profile data consists of 1000 pairs of input and output sequences. Due to the great variation in factors such as explored nodes and computational time the optimization is performed for 100 sequences for each instance. Dependencies and correlations among results for computation time, number of explored nodes, number of orders and number of colors are determined on the basis of mean and variance values.

Table B.2 displays mean μ and standard deviation σ values for the number of explored nodes rounded up to integers as well as for computational time rounded up to four decimal points and the mean value of objective values rounded up to one decimal point. We note, that for the instances 30_2-2, 40_4-1, 40_4-2, 40_8-1, 40_8-2, 50_4-1, 50_4-2, 50_8-1, 50_8-2, calculations got interrupted by reaching the time limit. The statistic values in Table B.2 are determined from the results when each calculation terminates, whether an optimal solution is found or the time limit is reached. The share Δ of interrupted calculations is specified in the last column of the table, if it differs from the whole set.
The standard deviation of number of explored nodes as well as of the computational time summarizes the great variations in both of these parameters. We observe that μ(obj val) is always smaller for the mono-distributed set than for the corresponding poly-distributed set. With the exception of set 30_2-1 and 20_2-2, the average objective value μ(obj val) increases with growing n for fixed number of color codes and mode

of color distributions.
Comparing the results for mono-distributed profile data to those of the corresponding poly-distributed sets, the former one exceeds the latter one by at least approximately 34% in number of explored nodes (see pair of values 10_4-1 and 10_4-2) and about 31% in computational time[18] (see 15_4-1 and 15_4-2).
This survey also shows the impact of the number of defining properties and of their characteristics on the complexity of the problem, the size of the search space in particular. The smaller the number of different color codes in the data set, the greater the number of explored nodes and the longer the computational time. Potential reasons for this are reviewed in Section 3.7.

For each instance, find 10 of the 100 calculations explicitly presented in Table B.1. We consider the number of explored nodes, number of simplex iterations, model status, objective value, MIP gap in % and computational time in s. Over all instances there are great variations in number of explored nodes, and thus in number of simplex iterations, and the computational time.[19]

In analogy to Table B.2 Table B.4 shows the statistic values for the identical 100 sequences per instance with setting the MIPStart attribute in the Gurobi^{TM}Optimizer. For each calculation the result of the mixed integer program (3.30a)-(3.30d) is passed to the solver before the optimization begins. We observe the same correlations between the number of color codes and the calculation time or the number of explored nodes as well as between the calculation time or the number of nodes for mono-distributed as well as for poly-distributed data sets.

The comparison of the results with and without MIPStart shows the range of the impact MIPStart has on the number of explored nodes and the computational time. There are several gradations from performance improvement by around 87.5% in computational time[20] to a negative effect in form of an extension by around 276%[21]. This appears to be independent from the MIPStart objective value. The Gurobi^{TM}Optimizer logs for sequence 30_2-4-96 (see Figure B.3 and Figure B.4) are an example where the use of MIPStart extends the computational time by the factor 3.7 although the objective value of the start solution[22] is significantly smaller than the objective value of the first few

[18]Both with regard to the mean values.
[19]We remind that sequences of mono-distributed data sets are permutations.
[20]for data set 30_4-1
[21]for data set 20_2-1
[22]objective value MIPStart: 9

incumbents[23] Gurobi^{TM}finds by itself. In this case the time consuming part is proving optimality. Passing a start solution can lead to an adverse path through the search space. There are other sequences where closing the gap between lower and upper objective bound is the major portion of the computing time, with or without MIPStart. For further examples see Table B.3. It itemizes the objective value of the MIPStart solution, the number of explored nodes, model status, objective value of the solution, gap in % as well as computational time for the exact same sequences as those presented in Table B.1. We observe similar variations in the values as in Table B.1 and additionally in the objective value of the MIPStart. All instances are effected by the phenomenon that MIPStart reduces the computing time for some sequences, while it keeps the time uninfluenced or even extends it for others. We examine the impact exemplarily with four sequences (see Table 3.8).

		difference obj values of MIPStart and solution	
		small	great
MIPStart effect on comp time	positive	20_4-1-8	20_2-2-10
	negative	30_8-1-4	15_2-2-8

Table 3.8.: Examples overview

Problem 20_4-1-8 shows the MIPStart effect we hoped to see for all the problems. The difference between the objective value of the start solution and of the solution is 2 and thus quite small (see Table B.3). The use of MIPStart reduces the computational time by the factor 0.5.
The second example for a desired effect is problem 20_2-2-10. The difference of both objective values is rather big (9). MIPStart brings the computational time down from 68.89s to 21.80s.
For problem 15_2-2-8 the difference of the objective values also equals 9. However,

[23]objective value first three incumbents $92, 23, 21$

MIPStart leads to an extension of the computing time by the factor ≈ 4.
Problem 30_8-1-4 is an example for the fourth case. Despite a very small difference of 1, the use of MIPStarts extends the computing time from 8.42s to 33.28s.

The average growth of computational time for n_4 data sets is given by Figure B.1 for mono-distributed sets and Figure B.2 for poly-distributed sets. Recall that the number of variables of MIP (3.29a)-(3.29m) is given by $2n^2+2n+\mathcal{O}(n^2)$. The figures show the computational time subject to the leading term $2n^2$. In each case the values are plotted with a linear axis and a logarithmic axis. For sequence length $n \geq 30$ there are sequences in the data set, where the optimization does not terminate with an optimal solution but by reaching the time limit. We include these sequences in the calculation by taking the time limit of $50000s$ as an approximation of the computational time.
Each diagram shows six graphs. They refer to calculations with and without MIPStart in equal parts. Data points derived from the results in Table B.2 and B.4 are connected with a dotted line for $n \geq 30$. The exponential fits with the continuous line are determined on the basis of computational times for sequence length $n < 30$, where each problem is solved until optimality. The dotted lines represent the exponential fit calculated by taking all sequence lengths into account. The graphs can be understood as upper and lower bounds. The use of MIPStart has a significant effect on the average computational time for $n > 20$. By setting a time limit, this effect has less impact for $n = 50$ in proportion. We have similar developments for mono- and poly-distributed sets with stronger growth over all graphs.

The Gurobi$^{\text{TM}}$parameters TimeLimit and MIPGap are set to 50000s and $3e-2$, respectively. The weight in constraint (3.29j) is set to $\varepsilon = 0.02$.

The results have been generated with an Intel-i7 CPU, Gurobi$^{\text{TM}}$Optimizer version 8.0.1 on a Lenovo notebook, code in Visual-C# 2017.

3.7. Discussion of the MIP results

The computational time decreasing with growing number of color codes for a fixed n, arises from combinatorics. The objective function (3.29a) measures the distance between two sequences on the level of Rohbaulack variants. Let r be the number of Rohbaulack variants in a daily order set and let b_i, $i = 1,\ldots,r$ be the number of orders of Rohbaulack variant i. Then, the number of sequences modulo Rohbaulack variants is given by (see Section 3.2)

$$\frac{n!}{\prod_{i=1}^{r} b_i} \quad . \tag{3.35}$$

The number and distribution of body types and thus of Rohbaulack variants is not fixed in the artificial data algorithm (Section 3.3.2). However, the random selection of order configurations from real data secures that the artificial data show the correlation: the smaller the number of color codes the greater the size of equivalence classes for each Rohbaulack variant. Thus, less color codes presume less different sequences modulo Rohbaulack variants and more equivalent sequences. We interpret the phenomenon of increasing computational time with decreasing number of color codes as a result of Gurobi$^{\text{TM}}$struggling with constraint (3.29j).

Sequences equivalent to the optimal solution modulo Rohbaulack variant do not necessarily satisfy constraint (3.29j). The greater the number of equivalent sequences, the greater the risk for the solver to follow a path into depth and to explore nodes that can be pruned by checking constraint (3.29j) for integer solutions. In the progress of the branch-and-cut algorithm the solver follows tree paths into a depth, where constraint (3.29j) can only be satisfied by the partly relaxed solution of the node, but not by an integer solution of this path.

This effect is enhanced by the following, also concerning constraint (3.29j) and MIP (3.30a)-(3.30d). We explain it by the example of 30_4-1 and 30_2-1. Let $x_j^i \in 30_4$-1 and let $\tilde{x}_l^k \in 30_2$-1 be sequences. Then, there exists a color code c in the orders of $\tilde{x}_l^k$ such that there are more orders of color c in sequence $\tilde{x}_l^k$ than there are in sequence x_j^i. Suppose the number of orders are two and five, respectively. We consider orders of one color but we keep the strong dependencies of all of the sequence orders in mind. Suppose, there are two possibilities for the orders of color c taking shifts, that are both optimal. Figure 3.3 shows the probability distribution of color code c (data points on solid line) as well as the relative frequencies for two possibilities of the orders taking shifts. Let both possibilities correspond to nodes in the search tree of an optimization call for MIP (3.29a)-(3.29m). Suppose the filled bars are part of a feasible solution and the lined bars are part of an infeasible solution due to violation of constraint (3.29j). Comparing both cases (Figure (3.3) and Figure (3.4)), we can imagine that infeasibility can be detected earlier in optimization process. However, in the five-orders-case there is the possibility to follow the corresponding tree path, since the partly relaxed solution is still feasible for some of the five orders fixed to a position. Checking constraint

(3.29j) for an integer solution regarding variables x_{ij} (corresponding orders of color c as well as others colors), that are already fixed to 0 or 1, may detect infeasibility at an earlier node.

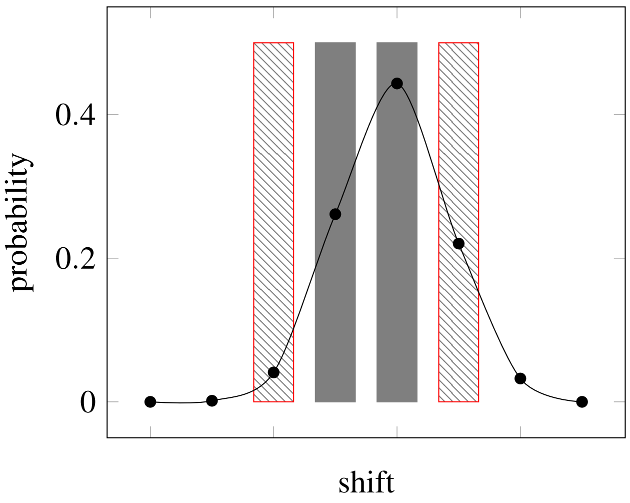


Figure 3.3.: Probability distribution and relative frequency for two orders

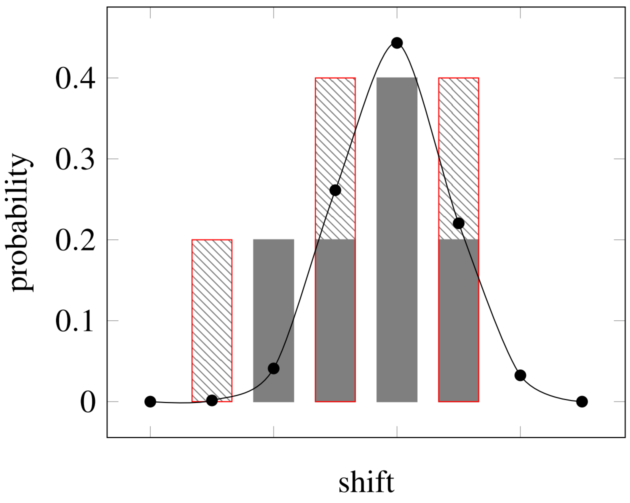


Figure 3.4.: Probability distribution and relative frequency for five orders

Setting the Start attribute and passing a feasible solution before the optimization call can help Gurobi^{TM}in cases where it struggles to find an initial feasible solution. It is possible to specify only some of the variables. The Gurobi^{TM}solver tries to construct a complete solution from the provided information [Gurb]. Examples of this effect are observed in the data. It is rarely the matter to overcome the struggle to find an initial feasible solution. In fact, Gurobi^{TM}manages to find a feasible solution for many of the

problems very quickly. In the case presented above, MIPStart provides a quite good solution regarding the objective value. For a part of the performed calculations we observe the improvement of the calculation time intended by the use of MIPStart.
However, there is a significant part where MIPStart negatively affects the computing time. In numerous cases, passing a feasible solution makes it more difficult for the solver to prove optimality, in others, the solver quickly fails to close the gap between upper and lower bound. Figure B.1 and Figure B.2 show that the computational time scales exponentially on average.

The design of the spanning tree as well as the path through the tree in the process of the branch-and-cut method have a major impact on the performance. The search tree, that is spanned originating from the start solution, can lead to an improper tree design and a worse starting position. We observe this phenomenon not only in comparing calculations with and without MIPStart but also among problems with a fixed attribute setting. In particular we identify constraint (3.29j) as a main factor for the performance issues and turn this into an opportunity by customizing a heuristic (see Chapter 4).

Potential adjustments for the use case

We perform the following considerations neglecting the computational times that do not allow any applications so far. Extensive testing and venier adjustments on the model are only possible by several experiments in the factory.
Assuming that there is a uniquely identified sufficiently small set of defining properties the quality of the results determined by the MIP is to be tested and evaluated in the ongoing operation in the plant. This enables us to weight the shifting of the orders by the color specific mean (simulation performed in the objective function) against the constraint of minimizing the distance to the probability functions by choosing a proper ε in the constraint $F_{\tilde{y}} \leq F^* + \varepsilon$.
We suggest to face the plurality of perturbation relevant factors by the following approach consisting of four steps.

1. analysis of the whole shop and identification of spots and sections where perturbation occurs

2. analysis of the perturbation factors and classification (see Figure 3.5)

3. adjustment of planning steps and production stages in order to eliminate perturbation factors

4. design of a mini model for each section and connecting those models in series.

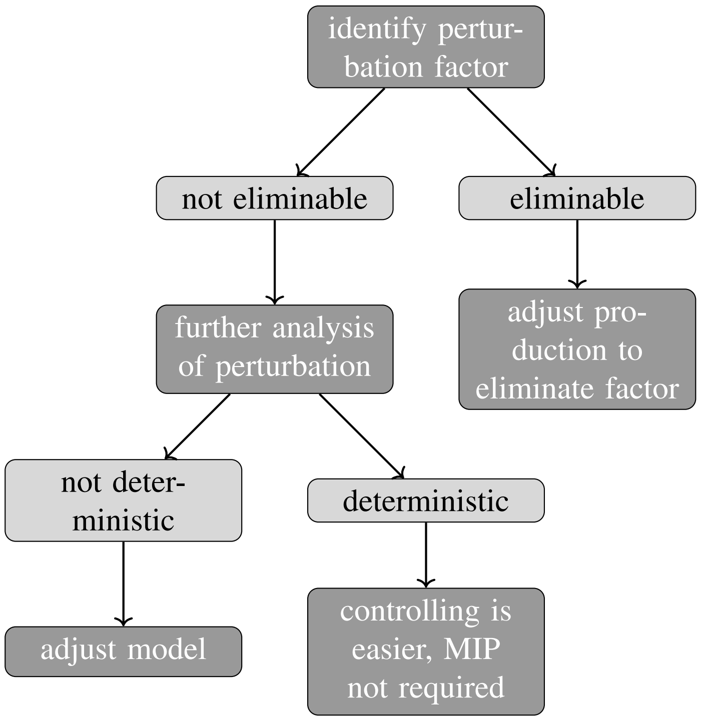

identify perturbation factor
not eliminable
eliminable
further analysis of perturbation
adjust production to eliminate factor
not deterministic
deterministic
adjust model
controlling is easier, MIP not required

Figure 3.5.: Classification of perturbation factors

Possible adjustments of the model

We decided to weight all the colors and Rohbaulack variants equally. Depending on the use case it might be necessary to assign different weights to the probability functions. In order to control both, the paint shop and the body shop, we recommend to design a MIP for each of them. The sequences are calculated against the production direction. As before the assembly line provides the optimal output sequence of the paint shop, the optimal input sequence of the paint shop then serves as optimal output sequence for the body shop. The more models combined, the more vulnerable the results for inaccuracies and variations.
Manufacturing or supplying reasons might require additional constraints on the input sequence. Examples are minimum distance for orders with a certain property or a lower/higher bound on the position of orders with a certain property.

Addressing the problem might give rise to the idea of splitting the given optimal output sequence into several subsequences optimizing them independently and then obtaining the optimal input sequence. However, this approach leads to greater distance between the shifts and the color code probability distributions. Furthermore, we strongly assume that the relations between the orders conflict with the splitting up of the sequences.

4. Heuristic

It is common knowledge that the selection of the branching variable and of the unsolved subproblem is of significant influence on the computational time in a branching algorithm. The impact depends on the structure of the problem and no universal method is known among the great number of selection strategies. State-of-the-art solvers such as Gurobi$^{\text{TM}}$Optimizer offer the choice between a handful of settings to adjust the solving strategy to a specific problem.
Customized algorithms can be a way to tackle a problem individually and thus can outperform commercial and noncommercial solvers for that specific kind of problem.

We find that the computational time of the mixed integer program presented in Section 3.5 grows exponentially and varies significantly between sequences of the same length. This is an indication that the Gurobi$^{\text{TM}}$algorithms do not handle the MIP perfectly, but there is potential for improvement on the solving strategy. Regarding the use case, the model is not yet proper for employment since in this particular case a computational time of 600s for sequences of 1400 to 1700 orders is not to be exceeded.
The long computational times and the variations can not be justified simply by the size of the program, in particular not the immense variations. We pinpoint one constraint as the main cause

$$F_{\tilde{y}}(x) \leq F^* + \varepsilon \quad . \tag{4.1}$$

The operating mode of the branch-and-cut algorithm can lead to a situation, where a solution branch is pursued far into the depth before it gets cut off. Constraint (4.1) can be valid for a partly relaxed solution but invalid as soon as we enforce integrality for all of the variables. This means adding some additional information to the solution algorithm offers the opportunity to cut off invalid solutions at an earlier stage in the solving process.

Assumption 4.1. *We assume that checking constraint (4.1) for integer satisfiability of subproblem solutions can cut off the size of the search space by a significant part and therefore reduce the computational time of some problems.*

In the following sections we give a survey on the main components of branch-and-bound algorithms in preparation for a customized branch-and-bound strategy for MIP (3.29a)-(3.29m).

4.1. A survey on branching methods, node selection strategies and pruning rules

4.1.1. Most common branching strategies

The branching strategy, the node selection strategy (also called search strategy) and the pruning rules are the three core components of every branch-and-bound algorithm and thus crucial for the performance of solvers.
We follow mainly the work of Achterberg [Ach07], Achterberg et al [AKM05] and Morrison et al. [Mor+16] giving a revealing insight of the strength of branching, variable selection and pruning rules. This section as well as Section 4.1.2 and Section 4.1.3 summarize the basics and the state of research.

The only way to split a problem in an LP based branch-and-bound method while keeping the property of having a LP relaxation is to branch on linear inequalities. The simplest and most common inequalities are trivial inequalities, which means inqualities that split the feasible interval of a single variable. If x_j is some integer variable with the fractional value $\tilde{x}_j$ at the current optimal relaxed solution, we obtain two subproblems: one by adding the inequality $x_j \leq \lfloor \tilde{x}_j \rfloor$, the other one by adding $x_j \geq \lceil \tilde{x}_j \rceil$. This procedure of branching on trivial inequalities is also called branching on variables. Branching on more complex inqualities or splitting into more than two subproblems is quite rarely implemented in common MIP solvers [Ach07].

In the following we focus on the most common branching rules that are all based on the same principle. For each branching variable candidate a score is determined. The variable with the highest score value is selected to be branched on. Since a global procedure is not known the variable selection depends on its benefit for the current branching. One possibility of measuring this benefit is by the change on the objective value of the two child nodes in comparison with the objective value of the relaxation of the parent node.

See the PhD thesis of Achterberg [Ach07] for a great survey [Ach07, Chapter 5] and computational analysis [Ach07, Section 5.11] of the most common branching rules.

Most fractional/infeasible branching is one of the most common rules. It chooses that variable, whose fractional value is the closest to 0.5. Its counterpart is the *least infeasible branching* that picks the variable closest to integrality. Both rules perform rather poorly.
The *pseudocost branching* rule estimates the objective change in the LP relaxation for branching upwards and downwards. It sums over all subproblems where it has been branched over the same variable x_l. The average objective change values are called the pseudocost. Variations of this rule are *uninitialized pseudocost branching* and *reliability pseudocost branching*, for example.
Originally developed for the TSP, *strong branching* was soon implemented in solvers as Gurobi$^{\text{TM}}$and CPLEX$^{®}$. Strong branching means to test which of the fractional candidates provides the best progress for the dual bound. This test is performed by introducing an upper bound $x_j \leq \lceil \check{x}_j \rceil$ and a lower bound $x_j \geq \lfloor \check{x}_j \rfloor$ on variable x_j with LP relaxation fractional value $\check{x}_j$ and solving the linear relaxations. If we choose the set of all variables with noninteger values for the candidate set, it is called *full strong branching*. It is apparent that this branching strategy has high computation times for each node. By restricting the candidate set, a speed up of the algorithm can be achieved.
In addition to the branching rules mentioned above, there are several variants and hybrids of these.

The choice of the branching strategy impacts both of the two phases of a branch-and-bound algorithm, the search phase and the verification phase [Mor+16].

Gurobi$^{\text{TM}}$offers five variable selection strategies. The default setting is the not further specified choice “depending on problem characteristics”. The available alternatives are pseudo reduced cost branching, pseudo shadow price branching, maximum infeasibility branching, and strong branching. The Gurobi$^{\text{TM}}$Manual points out that changing the selection strategy rarely provides significant benefit [Gura].

4.1.2. Most common search strategies

After one subproblem in the search tree has been processed it has to be chosen which node to continue with. This choice often impacts primarily the search phase of the branch-and-bound algorithm[Mor+16]. Search strategies are also referred to as node selection strategies. The available selection strategies can be divided in two categories: *depth first search* and *best first search*.

Both strategies have opposing goals. The depth first strategy is based on the observation that integral LP solutions are found deep in the search tree. This selection strategy always chooses a child of the current node to be processed next. If the current node is pruned the algorithm proceeds with a child of the most recent ancestor having an unprocessed child left. This strategy is a good choice for pursuing to find an improvement on the upper bound, which helps to prune the search tree. A second advantage is the similarity of the current subproblem and the next subproblem. Often, there are only a few changes required on the LP relaxation. In particular, when branching over variables by setting an upper and lower bound on an integer variable the update on the LP relaxation consists of changing the value of a single variable. The main disadvantage is the risk to explore nodes that could have been pruned if a good solution was available earlier [Ach07].
Best first search selects the node with the smallest dual bound of all remaining leaves in the search tree. When the branching rule is fixed, best first search leads to a minimal number of nodes that have to be processed [Ach07, Proposition 6.1]. However, since this strategy is not uniquely defined (several nodes can have the same value for the dual bound), not every best first search processes the minimal number of nodes. Tending to perform as a breadth first strategy, best first search leaves us with the disadvantage of higher computation time for each node since the next subproblem has little relation to the current one [Mor+16, Section 3].
Hybrids of both seach rules, such as the *cyclic best-first search*, seek to combine the advantages of both strategies. Changing the node selection strategy during the solving process is another possibility for improvement.

The Gurobi$^{\text{TM}}$Optimizer allows to specify which child node to process first via the BranchDir parameter. With the parameter MIPFocus one changes the focus between finding feasible solutions in good quality, proving optimality and a balance between both. However, the search strategy can not be specified explicitly.

4.1.3. Pruning rules

The worst-case-running time of a branch-and-bound-algorithm is given by

$$\mathbb{O}(Mb^d) \quad , \tag{4.2}$$

where b denotes the maximum number of children generated at any node in the tree, i.e the *branching factor*, and d denotes the *search depth*, that is the longest path from the root to a leaf. The factor M is a bound on the time required to explore a node. Pruning rules are the primary way to reduce this complexity in practice [Mor+16].

If it can be proven that no solution in the current branch has a better objective value than the incumbent solution, the current subproblem is pruned. One of the most common rules is *pruning by bound*. Pruning by bound means to introduce a lower bound on the objective function value and use this to prune nodes whose lower bounds are not better than the incumbent's solution value. The optimal objective value of the LP relaxation is a quite frequent choice [Mor+16, Section 5].
When applying *dominance relations*, a subproblem is pruned if it can be shown to be dominated by another subproblem. We call a subproblem P_1 dominating another subproblem P_2, if for every descendant of P_2 there exists a complete solution descending from P_1 that is at least as good. Thus, it suffices to explore S_1. We distinguish memory-based and non-memory-based dominance relations. The former one requires more memory since it stores the entire search tree during the solving process, the latter one just implies the existence of a dominating subproblem but tends to be less effective. This pruning rule is important for solving MIPs with a high degree of symmetry, in particular [Mor+16, Section 5.2]. Other pruning possibilities are *pruning by optimality* and *pruning by infeasibility*. In many cases pruning rules impact mainly the verification phase of the branch-and-bound algorithm [Mor+16].
Note, that the choices for the three components, branching strategy, node selection strategy and pruning rules affect each other. In particular, the choice of pruning rules impacts the choices that can be made for the other two components [Mor+16].

Node pruning in Gurobi$^{\text{TM}}$is managed by adding cutting planes via callbacks.

4.2. A customized branch-and-bound algorithm

4.2.1. Idea

We remind that the feasible region of MIP (3.29a)-(3.29m) ignoring constraint (3.29j) is the Birkhoff polytope. Recall from Section 2.2.1 that this polytope is huge which effects the performance of solution algorithms. Using knowledge about the problem structure and about constraint (3.29j) in particular, the search space of the MIP is expected to get cut enormously.

Idea 4.2. *For each subproblem determine the assignment of the variables x_j that have been branched on. These equations are added to the optimization problem* $\min F_{\check{y}}(x)$ *(3.30a) - (3.30d)as constraints. If the objective value of the optimal solution is greater than $F^* + \varepsilon$ the corresponding node is pruned.*

Since the Gurobi$^{\text{TM}}$Optimizer does not support retrieving branching variables and partly relaxed solutions at each node in the branching tree via callbacks[24] we implement a customized branch-and-bound algorithm.
Algorithm 2 combines a depth first search strategy with a problem specific pruning rule. The efficiency of this pruning rule is favored by the depth first search. For each node the upwards branch $x_j \geq \lceil \check{x}_j \rceil$ is chosen to be explored first. As branching rule we choose most fractional branching. Low computation times and simplicity of the implementation beat the usually rather poor performance of this branching rule.
Figure 4.1 visualizes the benefit of the heuristic compared to a standard solve call in the Gurobi$^{\text{TM}}$Optimizer regarding the size of the search tree. For the smaller nodes we have, the dark gray ones symbolize subproblems, where $F_{\check{y}}(x) < F^* + \varepsilon$ holds for an integer solution. Subproblems, where $F_{\check{y}}(x) < F^* + \varepsilon$ is satisfied by the corresponding partly relaxed solution but not for any corresponding integer solution are represented by the light gray nodes. The blue node represents the optimal solution. The bigger nodes correspond to those that are pruned with respect to the constraint represented by the respective color.

[24]Note, that the variables Gurobi$^{\text{TM}}$branches on may not coincide with the variables in the original formulation of the model due to Gurobi$^{\text{TM}}$presolve transformations.

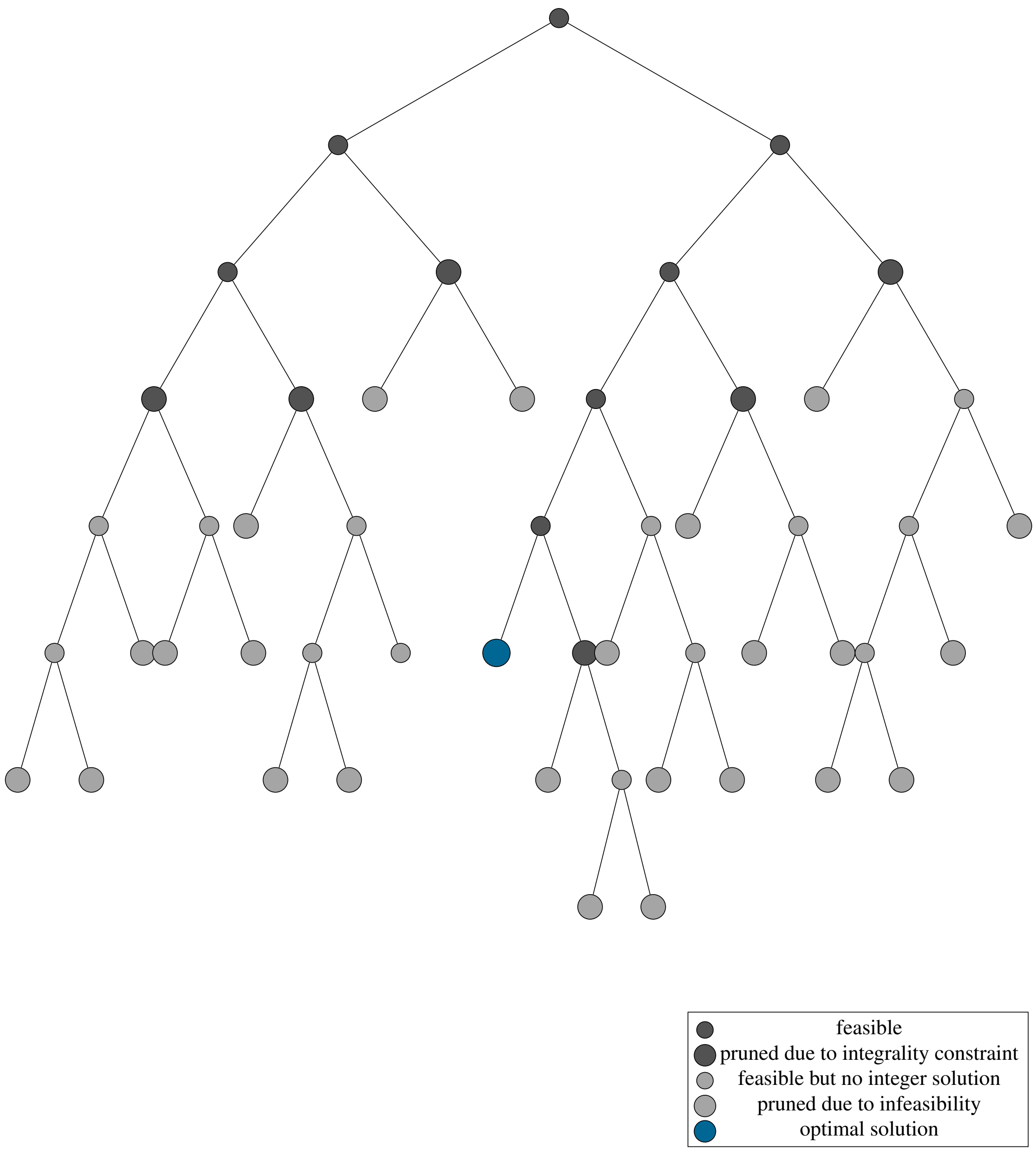

feasible
pruned due to integrality constraint
feasible but no integer solution
pruned due to infeasibility
optimal solution

Figure 4.1.: Exemplary search tree comparing heuristic and Gurobi Optimizer

4.2.2. A branch-and-bound algorithm with customized pruning rule

Let N denote the node pool. We denote the LP relaxation of the original problem by P_0. Let P_k denote the kth subproblem of P_0. The LP relaxation P_0 is given by

$$\min_{x} \omega_{\text{rel}}(x) = \min_{x_{ij}} \sum_{i=1}^{n} \left| z_i - \left(\sum_{l=1}^{|[x_i]|} {}^{x_i}b_l{}^{m} * t_l^{|[x_i]|} \right) \right| \tag{4.3a}$$

$$\text{s.t} \quad \sum_{l=1}^{|[x_i]|} {}^{x_i}b_l{}^{m} = 1 \qquad \text{for all } m = 1,\dots,|[x_i]| \tag{4.3b}$$

$$\text{for all equivalence classes in } \tilde{y}/\tau$$

$$\sum_{m=1}^{|[x_i]|} {}^{x_i}b_l{}^{m} = 1 \qquad \text{for all } l = 1,\dots,|[x_i]|, \tag{4.3c}$$

$$\text{for all equivalence classes in } \tilde{y}/\tau$$

$${}^{x_i}b_l{}^{m} \in \{0,1\} \tag{4.3d}$$

$$z_g = \sum_{i=1}^{n} r_{ig} - 1 \qquad \text{for all } g = 1,\dots,n \tag{4.3e}$$

$$p_g - p_i \le r_{ig} M \qquad \text{for all } i,g = 1,\dots,n \tag{4.3f}$$

$$-(p_g - p_i) \le (1 - r_{ig}) M \qquad \text{for all } i,g = 1,\dots,n \tag{4.3g}$$

$$r_{ig} \in \{0,1\} \qquad \text{for all } i,g = 1,\dots,n \tag{4.3h}$$

$$p_i = \sum_{j=1}^{n} x_{ij} * j + \mu_i \qquad \text{for all } i = 1,\dots,n \tag{4.3i}$$

$$F_{\tilde{y}}(x) \le F^* + \varepsilon \tag{4.3j}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for all } j = 1,\dots,n \tag{4.3k}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for all } i = 1,\dots,n \tag{4.3l}$$

$$x_{ij} \in [0,1] \tag{4.3m}$$

where F^* denotes the objective value of an optimal solution of the MIP

$$\min_x F_{\tilde{y}}(x) = \min_{x_{ij}} \sum_{h=1}^{C} \sum_{\substack{k=-n+1 \\ 1\leq i+k\leq n}}^{n-1} \left| \frac{s_{h,k}}{l_h} - f_h(k) \right| \tag{4.4a}$$

$$\text{s.t} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for all } j = 1,\ldots,n \tag{4.4b}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for all } i = 1,\ldots,n \tag{4.4c}$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } i,j = 1,\ldots,n \tag{4.4d}$$

Remark. In the context of Algorithm 2 we slightly abuse the term LP relaxation in the sense that not all of the integer variables of the original problem are relaxed but only the decision variables x_{ij}. The integrality constraints on the auxiliary variables ${}^{x_i}b_l{}^m$ and r_{ig} still hold.

We will use P_k to refer to both, the subset and its corresponding branch-and-bound node, from now on. Let x^k denote the partly relaxed solution of problem P_k and let $\tilde{x}^k$ be the setting of branching variables of Problem P_k.
We fix a number $\delta \in \mathbb{N}$. If $\delta = 0$, a second optimization problem is solved at each node

$$\min F_{\tilde{y}}(\tilde{x}^k) \quad . \tag{4.5}$$

The notation of x^k as argument means the following: the values of decision variables, that are already branched on, are added to the MIP $\min F_{\tilde{y}}$ as constraints. The MIP is solved over the decision variables that are not yet fixed to a value by branching. If $\delta > 0$, the MIP $\min F_{\tilde{y}}(\tilde{x}^k)$ is solved at each node P_k with $k = 0 \mod \delta$ (see Algorithm

2). Let *ub* denote the upper bound on the objective value.

Algorithm 2: Customized branch-and-bound algorithm

```
Input: $\delta$
initialize $N = \{P_0\}$, incumbent $\hat{x} \leftarrow \min F_{\bar{y}}(x)$, $ub \leftarrow \omega(\hat{x})$;
while N is not empty do
    take node $P_k$ from node pool;
    solve subproblem $P_k$;
    if $P_k$ is infeasible then
        node is pruned;
    end
    else
        $x^k \leftarrow \min P_k$;
        if $x^k > ub$ then
            node is pruned;
        end
        else
            if $x^k$ is integer then
                update incumbent $\hat{x} \leftarrow x^k$;
                unpdate upper bound $ub \leftarrow \omega(x^k)$;
            end
            else
                branch according to most fractional branching;
                get two nodes $P_{k_1}$ and $P_{k_2}$;
                foreach node do
                    if $\delta = 0$ then
                        solve MIP min $F_{\bar{y}}(\tilde{x}_k)$;
                        if $F_{\bar{y}}(\tilde{x}_{k_l}) \leq F^* + \varepsilon$ then
                            add node $P_{k_l}$ to $N$;
                        end
                        else
                            node is pruned;
                        end
                    end
                    else
                        if $k_l = 0 \mod \delta$ then
                            solve MIP min $F_{\bar{y}}(\tilde{x}_k)$;
                            if $F_{\bar{y}}(\tilde{x}_{k_l}) \leq F^* + \varepsilon$ then
                                add node $P_{k_l}$ to $N$;
                            end
                            else
                                node is pruned;
                            end
                        end
                        else
                            add node $P_{k_l}$ to $N$;
                        end
                    end
                end
            end
        end
    end
end
return $\hat{x}$;
```

There is no start solution handed over to the MIP solver at each node. Due to the relation between the current node and the successive node, depth first search suggests itself for so-called warm starts of the subproblems (see Section 4.1.2). However, in a common branch-and-bound algorithm the subproblems are relaxed and as such are LPs, whereas in Algorithm 2 a MIP is solved at each node. Generalizing the results in Section 3.6 to partly relaxed versions of the MIP we dispense potential improvements in computational time and thereby do not risk negative impact on the same parameter for $n > 30$. Recall that for sequence length of $n \leq 30$ we did not determine a significant benefit of passing over a start solution on the average computational time.

4.3. Computational results of the heuristic

Algorithm 2 is evaluated with mono-distributed sets 10_4-1 and 15_4-1. The profile data sets consist of 1000 pairs of input and output sequences. Table C.1 shows the results for ten sequences of each instance. Each problem is solved with two parameter settings for δ, i.e. $\delta = 0$ and $\delta = 4$.
As a reference, the values for number of explored nodes and computational time of the MIP are listed in the last two colums.
The results show a significant impact of the choice of δ over all problems. The number of explored nodes increases by a factor of 6.5 on average for 10_4-1 and by a factor of 9.7 for 15_4-1, respectively. And thus, the computational time shows a similar behavior. It increases by a factor of 5.1 and 14.3. Looking at the results individually it shows that the heuristic branch-and-bound produces solutions with great variations in number of explored nodes and computational time. We remind of the analysis of the Gurobi^{TM}Optimizer results in Section 3.6.
Comparing the results of Algorithm 2 and of Gurobi^{TM}, the computational times in particular, we recognize that the former one exceeds the latter one by the factor of 23.1 on average for 10_4-1 and by 279.3 on average for 15_4-1 (parameter setting $\delta = 0$).

The results have been generated with an Intel-i7 CPU, Gurobi^{TM}Optimizer version 8.0.1 on a Lenovo notebook, code in Visual-C# 2017.

4.4. Discussion of the heuristic results

The increase in both of the parameters, number of explored nodes and computational time, when changing $\delta = 0$ to $\delta = 4$, confirms Assumption 4.1. Checking min $F_{\tilde{y}}(\tilde{x}^k)$ for an integer solution with an objective value $< F^* + \varepsilon$ at each node P_k, reduces the size of the search tree by a significant amount.
The fact that at least one MIP is solved at every node encounters the positive effect of the search tree reduction to such an extent, that the heuristic is noncompetitive with the Gurobi™Optimizer for any sequence length $10 \leq n \leq 50$. We remark that the main part of the computational time is consumed by subproblems P_k.
We identify the following factors as main reasons:

(i) each subproblem is a MIP, since the auxiliary variables are not relaxed

(ii) no presolve before the branching process is started

(iii) Gurobi™Optimizer is implemented performantly

Further theoretical considerations on modifications of the customized branch-and-bound algorithm

We are confident that the basic idea 4.2 performed within a MIP solving process via callbacks is a possibility to significantly reduce the computational time for $n \geq 20$. In Algorithm 2 the benefit of the reduction of the search tree size competes with the extension of the processing time of each node.
A customized branching strategy could conceivably improve the performance of the Branch-and-bound Algorithm 2. The findings in connection with Figure 3.3 and Figure 3.4 in Section 3.7 motivate a hybrid of branching by color codes and most fractional branching. We suggest prioritization of the orders relative to the frequency of the respective color code, where orders of the rarest color code have the highest priority.
However, it can be assumed that the advantage of the search tree cuts do not compensate for the computational time of the subproblems.

5. Neural Network

5.1. Neural Networks in business and industries

In recent decades the interest in neural networks for solving business problems grew rapidly. The idea of solving a complex pattern recognition problem by using a data-driven approach is no longer just an interesting challenge for researchers[SG00]. Many of these problems have typically been in the scope of operations researchers, such as forecasting, clustering and classification. The use cases originate from diverse sectors. Citing some examples we mention medical image recognition such as cancer detection [She+19], demand forecasting in manufacturing companies [Cha+19] or credit risk assessment [YWL08]. More fields of application are management, security, engineering, trading commodity, education and art.
Many problems are difficult to access by conventional modeling approaches due to lack of knowledge about the involved system, a high degree of uncertainty, time-varying characteristics or strongly nonlinear behavior [JA05].
"The principal difference between neural networks and statistical approaches is that neural networks make no assumptions about the statistical distribution or properties of the data, and therefore tend to be more useful in practical situations" [SG00].

5.2. Feedforward Neural Network and Recurrent Neural Networks

5.2.1. Feedforward Neural Networks

We give an introduction of the quintessentials of deep learning following mainly the work of Goodfellow et al. [GBC16].
Deep feedforward networks are also called *feedforward neural networks* or *multilayer perceptrons*. They are the basis of most deep learning models. Their main goal is to approximate a non-linear function f.
A classic example is a classifier, where $y = f(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f(x,\theta)$ and learns the value of the parameters θ that results in the best function approximation. The most common structure is a chain of functions. For example, let f_1, f_2, f_3 be three functions forming

$$f(x) = f_3(f_2(f_1(x))) \quad . \tag{5.1}$$

The overall length of the chain determines the *depth* of the network. In this case, f_i is called the ith *layer* of the network. First and last layer are also called *input* and *output layer*, respectively. The layers between them are *hidden layers*, due to the fact that the training data do not show the desired output for these layers. The hidden layers are typically vector-valued. They can be seen as consisting of single units that act in parallel, each representing a vector-to-scalar function. From another point of view, neural networks can be interpreted as fitting a probability density function

$$p(y \mid x) \tag{5.2}$$

instead of fitting a non-linear function. This can be achieved by using the maximum likelihood to estimate the parameter vector θ for a parametric family of distributions

$$p(y \mid x; \theta) \quad . \tag{5.3}$$

Example. The linear regression corresponds to the parametric family

$$p(y \mid x; \theta) = \mathcal{N}(y; \theta^T x, I) \quad . \tag{5.4}$$

Working with a feedfoward network requires some design decisions: choosing the optimizer, the loss function (also called cost function), the activation function for the hidden units and the form of the output units.
The type of activation function is essential for the accuracy of the predictions of a network. Non-linear activation functions are the most-commonly used ones [SA20]. The default recommendation is to use the rectified linear unit (ReLU) defined by the activation function

$$g(z) = \max\{0,z\} \quad . \tag{5.5}$$

The nonlinearity of a neural network often causes the loss function to be nonconvex. For this reason they are usually trained by using iterative, gradient-based optimizers. Other than convex optimization, gradient descent applied to nonconvex functions has no guarantee to converge and is sensitive to the initial parameter values.

An example for a common loss function is the *cross-entropy loss function* between the training data and the model's predictions, this means the loss function L is the negative log-likelihood[25]

$$L(\theta) = -\mathbb{E}_{x,y\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\, y \mid x\,) \quad . \tag{5.6}$$

The specific form of L depends on the form of $\log p_{\text{model}}$. For example, if $p_{\text{model}}(\, y \mid x\,) = \mathcal{N}(y; f(x,\theta), I)$, then we have the *mean erros loss function*

$$L(\theta) = \frac{1}{2}\mathbb{E}_{x,y\sim\hat{p}_{\text{data}}} ||y - f(x,\theta)||^2 + const \quad , \tag{5.7}$$

up to a scaling factor of $1/2$ and a term that does not depend on θ. Unfortunately, the use of the mean squared error and the mean absolute error function lead to poor results when used with gradient-based optimization. This is one reason why the cross-entropy is so popular.

This approach has the advantage that the maximum likelihood provides a loss function, so that it has not to be designed for each model. The loss function having a large and predictable gradient is essential for the learning algorithm. In many cases, several output units involve an exponential function, that becomes very small for small arguments $\ll 0$. The log-likelihood cancels the exp and prevents the function from becoming flat and thus the gradient from becoming too small [GBC16, Section 6.2.1].

The choice of the loss function is connected with the design of the output units. The

[25]We maximize the log-likelihood by minimizing the negative log-likelihood using gradient descent.

most common output units are linear, sigmoid and softmax. See [GBC16, Sections 6.2.2.1 - 6.2.2.4] for a comprehensive description of the relation between the choice of output units and of the cost function.

We complete the task of basic design decisions for a feed forward neural network by choosing the type of hidden units. Rectified linear units

$$g(z) = \max\{0, z\} \tag{5.8}$$

are good default choice. Apart from those, there is a great number of other types of hidden units available. The perfect decision can be difficult and require some experience in network modeling. More popular hidden units are logistic sigmoid and hyperbolic tangent with activation functions

$$g(z) = \sigma(z) = \frac{1}{1+e^{-z}} \tag{5.9}$$

and

$$g(z) = \tanh(z) \quad , \tag{5.10}$$

respectively.

Remark. The activation functions of some types of hidden units are nondifferentiable at usually a small number of points. For example, the function (5.8) is nondifferentiable at $z = 0$. However, gradient descent algorithms perform quite well in practice. Since we do not expect to reach a point where the gradient is equal to 0, it is acceptable if the minimum of the loss function corresponds to points with undefined gradient [GBC16, Section 6.3].

In deciding about the ideal depth of the network and the width of each layer, we are left to try and error. The *universal approximation theorem* for neural networks states, that sufficiently wide and deep multilayer feedforward neural networks are able to approximate any continuous function to any desired degree of accuracy [HSW89; Hor91; Les+93]. Lack of success is mainly due to inadequate learning, insufficient size of the network, lack of a deterministic relation between input and target data [HSW89].

5.2.2. Recurrent neural networks

The term feedforward arises from the fact that the information flows through the network in one direction without any feedback connections. Feedforward networks having feedback memories included are called *recurrent neural networks* (RNN). Networks with this feature are able to learn stochastic mappings, functions with feedback, and probability distributions over a single vector.
RNNs are specialized on processing sequential data. A recurrent network works on a sequence of values $x_1, \ldots, x_n$. The index $t = 1, \ldots, n$ is often referred to as time step, but needs not literally represent a passage of time in real world. It can also refer to the position in the sequence. An RNN may even work on two-dimensional spatial data such as images.
The crucial difference to nonrecurrent networks is the parameter sharing across different parts of the network. Sharing parameters makes it possible to apply the model to different forms, such as sequences of different length. Each element of the output sequence is a function of the preceding elements and each element is produced using the same update rule applied to previous output sequences [GBC16, Chapter 10].

Remark. Sharing parameters in RNNs relies on the assumption that a parameter can be used for different time steps. Furthermore, it is assumed that the conditional probability distribution of time step $t+1$ given the data for the first t time steps is stationary, which means the relation between a time step and the next time step does not depend on t [GBC16, Section 10.2.3].

An RNN processes information by incorporating it in the hidden state h and passing that forward through time. Other parts of the network are able to read information from h and make predictions. In a network that is trained to predict the future from the past, h_t is typically used as a summary of task-relevant aspects of the past sequence up to element t. In general, this summary is necessarily lossy, since it maps a sequence of arbitrary length $x_1, \ldots, x_t$ to a fixed length vector h_t [GBC16, Chapter 10].

5.2.3. Long-short term memory

A great advantage of RNNs is the ability to take into account the context when mapping an input to an output sequence. The range of context than can be accessed by a standard RNN can be expanded by using a *Long-Short Term Memory* (LSTM) archi-

tecture [Gra12, Chapter 4].
An LSTM consists of a set of recurrently connected subsets, so-called memory blocks. Each block contains a self-connected memory cell and three mulitplicative units - input, output and forget gates.
For details on the architecture and functioning of LSTMs see [Gra12; GBC16; SVL14; GMH13]. LSTMs have proven to be successful in a wide spectrum of applications that require a long range of contextual information: image recognition, handwriting recognition and music generation, to name a few [Gra12].

5.2.4. Autoencoders

An *autoencoder* is a neural network that seeks to copy its input to its output. An autoencoder consists of two parts, an encoder function $h = f(x)$ and a decoder function $r = g(h)$ [GBC16, Chapter 14]. The idea of autoencoders is pursued and refined since the 1980s. Originally, they were used for dimensionality reduction or feature learning. Modern autoencoders are generalized to stochastic mappings $p_{\text{encoder}}(\, h \mid x \,)$ and $p_{\text{decoder}}(\, x \mid h \,)$.
The learning process can simply be described as minimizing the function

$$L(x, g(f(x))) \quad , \tag{5.11}$$

where L denotes a loss function that penalizes $g(f(x))$ for being dissimilar to x, such as mean squared error.
When being trained to perform the copying task, the autoencoder extracts information about the training data, that are often of greater interest than the output of the network.

Example. Consider an autoencoder with a linear decoder and the mean squared error as loss function L. Let h be of smaller dimension than the input x. Then, the autoencoder learns to span the same subspace as Principal Component Analysis (PCA)[26].

An encoder maps a sequence of variable length to a fixed-dimensional vector. The symbol positions in the input are aligned to steps in computation time and thus generate a sequence of hidden states h_t as a function of the previous hidden state h_{t-1}

[26]PCA is a variance maximizing technique to reduce dimensionality. It is a linear transformation that projects the data into a lower dimension such that the variance is maximized [Pla18].

and the input for position t. As a sequential procedure this is not suitable for parallelization and becomes computationally inefficient for long sequences [Vas+17]. In order to overcome these difficulties, an *attention mechanism* is included in the network connecting encoder and decoder [FCB16]. The core of the attention mechanism is the attention function. The output of this function is a weighted sum of a query and a set of key-value pairs. The weights are called attention weights [FCB16]. They are learnable parameters.

If the hidden code has a dimension equal or greater than the input, the network may lern to copy the input without learning any useful features about the data distribution. There are several variants of autoencoders differing in the form of the loss function L or having an additional penalty term $\Omega(h,x)$, for example.
Autoencoders are often, but not necessarily, designed with a single layer encoder and a single layer decoder. However, the advantages of depth in a feedforward network also apply to autoencoders.

5.2.5. Hopfield neural networks

In the context of optimization problems and neural networks one almost necessarily encounters *Hopfield networks*. For the sake of completeness, this section is dedicated to a brief overview of the potential and limitations. In their pioneering work of 1985, Hopfield and Tank [HT85] applied a neural network to the TSP of the size of 10 and 30 cities. They presented a way of mapping a linear program into a closed-loop network, known as Hopfield network.
Solving the TSP is $\mathcal{NP}$-hard. In order to find optimal tours efficiently in practice, heuristics are designed. Although these heuristics perform well on the TSP, they have to be revised, if the problem statement changes slightly. Machine learning methods have the potential of developing their own heuristics from the data, thus they may require less handmade adjustments and be all-purpose.
An extension of the results of Hopfield and Tank to nonlinear optimization problems can be found in [XFW08] and to Bilevel programs in [He+14]. Neural networks can be powerful in finding good approximate solutions to difficult combinatorial optimization problems. Much of the literature about neural networks in optimization theory focus on the TSP, due to its benchmark status and the amount of research that has been done on this problem [Pot93].

A Hopfield network is a single layer, recurrent neural network [LMS99]. It consists of n interconnected neurons that are both, input and output neurons. Any two neurons are connected in both directions with weights. Let w_{ij} denote the weight of the connection from the jth to the ith neuron. We have $w_{ij} = w_{ji}$. The dynamics of a Hopfield network are defined by

$$\begin{aligned} \frac{du_i}{dt} &= \sum_j w_{ij} v_j + h_i \\ v_i &= \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{T}\right)\right) \quad , \end{aligned} \tag{5.12}$$

where $v_i \in [0,1]$, u_i and h_i are an output value, internal value and bias of neuron i.
The energy function of the network is given by

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij} v_i v_j - \sum_i h_i v_i \quad . \tag{5.13}$$

It holds that

$$T\frac{dv_i}{dt} = -2v_i(1-v_i)\frac{\partial E}{\partial v_i} \quad . \tag{5.14}$$

Equation (5.14) shows that the value of a neuron changes in order to decrease the energy E and converges to an equilibrium state, where ${dv_i}/{dt} = 0$, that is to zero or to a state where ${\partial E}/{\partial v_i}$.
Representing a combinatorial optimization problem in the form of an energy function and designing a Hopfield network with this energy function provides the opportunity to get an optimal or near optimal solution of the optimization problem by minimizing the energy function [Mat98].
Consider the optimization problem

$$\min \ c \ x \tag{5.15a}$$
$$\text{s.t.} \quad A_1 x \leq b_1 \tag{5.15b}$$
$$\vdots \tag{5.15c}$$
$$A_m x \leq b_m \quad . \tag{5.15d}$$

The Hopfield energy function is

$$E(x) = \alpha c x + \beta_1 (A_1 x - b_1)^2 + \cdots + \beta_m (A_m x - b_m)^2 \quad , \tag{5.16}$$

where $\alpha, \beta_1, \ldots, \beta_m$ are penalty parameters to reflect the relative importance of each term in the function. The network parameters weights and inputs are determined by comparing (5.16) and (5.13) [SM99]. See [Hop84; Mat98; SM99; JAS02; KPKP90] for a more detailed survey on this topic.

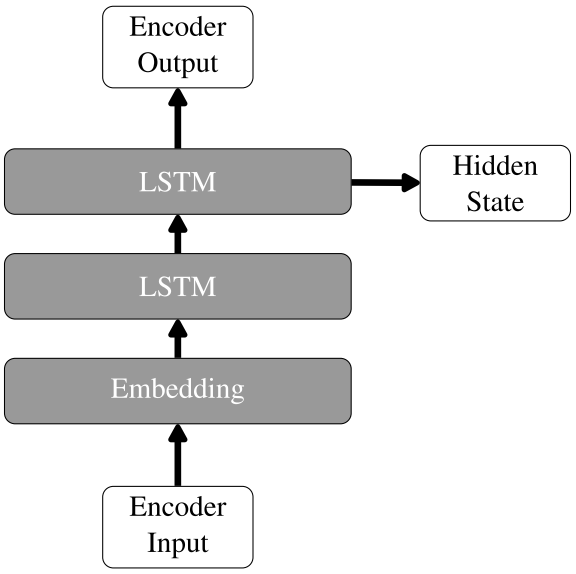
One of the major pitfalls of solving optimization problems with Hopfield networks is the fact that stable states of the network do not necessarily correspond to feasible solutions of the optimization problem. Several terms in the energy function compete to be minimized, a trade-off occurs. As soon as one term is not equal to zero, the corresponding solution is infeasible for the optimization problem.

MIP (3.29a)-(3.29m) requires an enormous energy function due to the number of constraints. A great number of penalty parameters increases the probability to get infeasible solutions.
We decided not to apply a Hopfield network but to feed the data to a neural network detached from the interpretation of the formulated optimization problem. We assume that the most valuable information comes from the input/output data and that no physical insight is available. This approach is known under the term *black box modeling* [JA05].

5.3. A sequence-to-sequence neural network

5.3.1. Design of the sequence-to-sequence network

In this chapter we do not distinguish individual orders but project every order on its color code and consider sequences of color codes. In the application of the network presented in the following we understand the mapping of output to input sequences as translation of a sentence consisting of color codes as words into a sentence of the exact same words.
In designing a neural network, we followed a sequence-to-sequence translation network provided by MathWorks® [Seq]. It is a recurrent encoder-decoder model.
Encoder and decoder of an RNN process the input and output data, respectively. An encoder typically includes a recurrent layer, such as an LSTM and extracts a fixed-length representation of a variable-length input sequence. The encoder of [Seq] is a stack of three layers, an embedding layer and two LSTM layers. Figure 5.1 provides an illustration.



Quelle: [Seq]

Figure 5.1.: Structure of the encoder

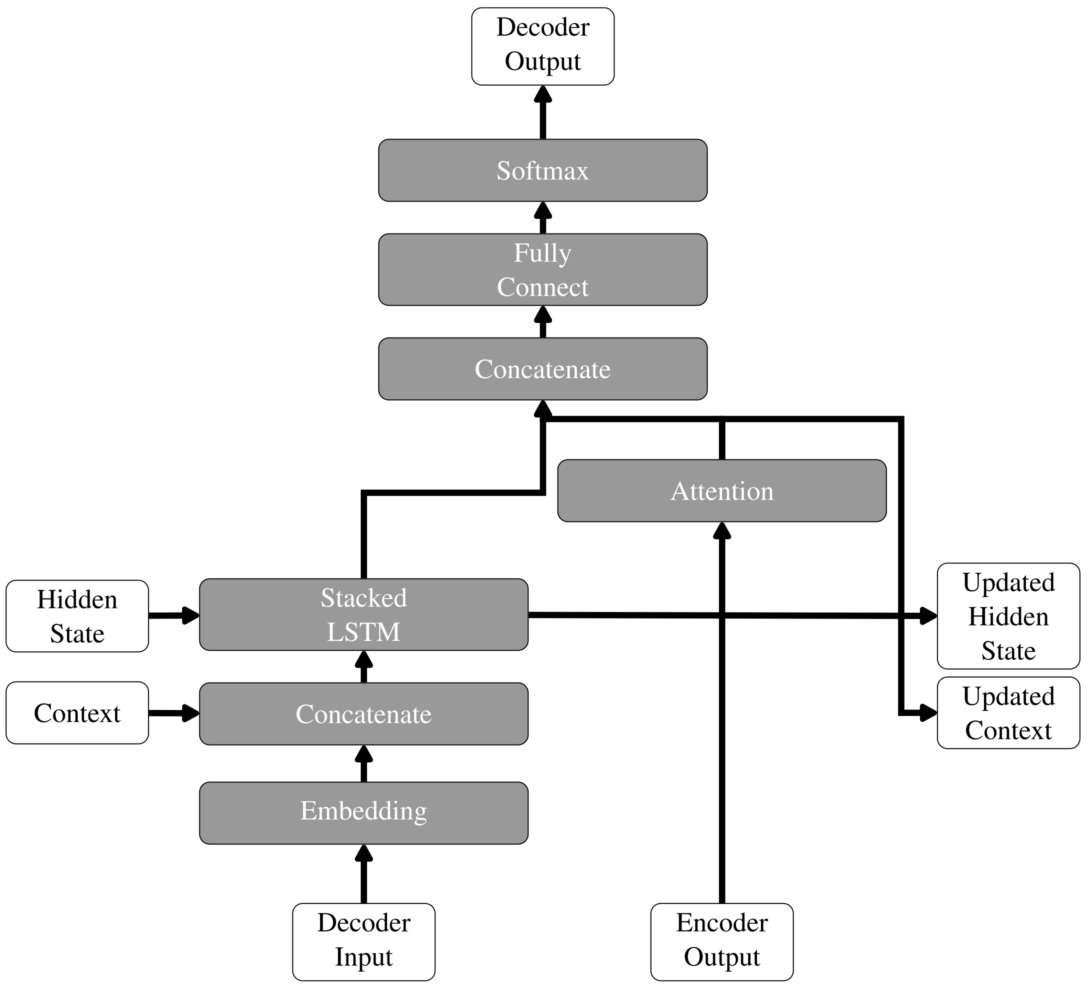
By the Matlab® Documentation [Wor] it is required to use a word embedding layer with an LSTM network. A word embedding layer maps word indizes to vectors and is mainly implemented for computational efficiency. Its dimension is set to 128. The number of hidden units in the two LSTM layers is set to 100. The probability of dropout layers with random dropout is set to 0.05. We summarize the parameter settings:

embedding dimension = 128

number of hidden units = 100

dropout = 0.05

In order to initalize weights and bias of layers MathWorks® Deeplearning Toolbox™ provides several functions. The weights of the word embedding layer are sampled from a normal distribution, for instance. Besides the attention mechanism, the decoder consists of five more layers, an embedding layer, two LSTM layers, a fully connected layer and a softmax layer (see Figure 5.2). It remains the task to define the encoder and decoder function, as well as the attention function and the model gradient function. The encoder function gets the input data, the encoder model parameters (input



Quelle: [Seq]

Figure 5.2.: Structure of the decoder

weights, recurrent weights and bias) and the mask, that determines the correct outputs for training, and returns the model output and the LSTM hidden state. In this case, the mask is the length of the input sequences. This is important when the data is varying in length which holds for most use cases.
The decoder function gets the input data, decoder model parameters, the context vector, the LSTM initial hidden state, the outputs of the encoder and the dropout probability. It returns the decoder output, the updated context vector, the updated LSTM state and the attention scores [Seq].
The attention function returns the attention scores, according to Loung scoring[27], and the updated context vector. The model gradients function takes the encoder and decoder model parameters, a mini-batch of the input data, the padding mask of the input data and the dropout probability. It returns the gradient of the loss with respect to the learnable parameters and the corresponding loss. The loss is computed by cross entropy.

The training options are specified

mini-batch size = 32

number of epochs = 100

learn rate = 0.002

We adopt the options of [Seq].

5.3.2. Preprocessing and formatting of the training and test data

We run the neural network on 27 data sets varying in overall size and sequence length. The data is generated by algorithm (3.3.2) with the parameters according to (3.3). We maintain the previous notation and write NN-n_C-1 for a data set with sequences of length n with C different mono-distributed color codes and NN-n_C-2 for a poly-distributed set, respectively. Compared to the MIP data, there is an additional mono-distributed set for each instance. Both of the mono-distributed sets differ by size, they

[27]There two different attentions used in an Encoder-Decoder network, Luong attention [LPM15] and Bahdanau attention [BCB15].

are denoted by NN-n_C-1-min and NN-n_C-1-max. Table 5.1 gives an overview of the data sets and their sizes.

data set	**size**	**data set**	**size**
NN-10_4-1-min	967	NN-n_4-1-min[28]	1000
NN-10_4-1-max	4135	NN-n_4-1-max	5000
NN-10_4-2	11200	NN-n_4-2	11200
NN-15_4-1-min	1000	NN-m_8-1-min[29]	1000
NN-15_4-1-max	4999	NN-m_8-1-max	5000
NN-15_4-2	11200	NN-m_8-2	11200

Table 5.1.: Neural network data set overview

In distinction to the data for the MIP we project each order on its color code before feeding it to the neural network. Thus, the network operates on sequences of ${}^{S}/_{\tau_3}$, where S denotes an order set.

The training set constists of 70% of the data set. The evaluated test set contains tuples of given output and input sequences and the calculated input sequence.
In order to analyze the results we defined some key figures. As a first step of the result analysis we remove tuples of sequences[30] from the test set, where the length of the sequence calculated by the network is not equal n. We refer to the adjusted test set as test set from now on. For an input sequence x and the corresponding calculated input sequence x_{calc} in a test set let the number of incorrect color codes ICC be

$$\text{ICC} = \sum_{h=1}^{C} |\nu_h(x) - \nu_h(x_{\text{calc}})| \quad , \tag{5.17}$$

where $\nu_h(x)$ denotes the number of color h in sequence x. We determine the mean value of incorrect color codes $\mu(\text{ICC})$ over all sequences in a test set.
For x and x_{calc} we determine the number of correctly placed color codes CPCC. Let

[28]for $n = 20, 30, 40, 50$
[29]for $m = 30, 40, 50$
[30]We slightly abuse the term sequence by calling each result of the neural network a sequence, even though it is not a permutation of the given output sequence.

$\mu(\text{CPCC})$ denote the mean value over all tuples of sequences in a test set, in particular also for sequences with ICC > 0.
For x, x_{calc} with x_{calc} being a permutation of x we define a measure in analogy to 3.22

$$d_{NN} = \sum_{i=1}^{n} |i - j| \quad , \tag{5.18}$$

where

$$j = \min\{k \,|\, x_i = x_k \text{ and } x_k \text{ is not assigned to another color code yet}\} \quad . \tag{5.19}$$

5.4. Description of the results for the sequence-to-sequence neural network

Table D.1 lists the computational results of the network for the parameter setting and training options defined in Section 5.3.1. In total, there are five data sets where the number of removed sequences equals zero, NN-10_4-1-min, NN-10_4-2 and NN-15_4-1-min, NN-50_4-2 and NN-50_8-2. In three data sets every pair of sequences is removed from the test set, which means none of the sequences determined by the network is of length n. Across all data sets, the majority of the calculated input sequences are not permutations of the given output sequences. The best results regarding the $\mu(\text{ICC})$ are achieved for NN-10_4-2. This value increases with decreasing size of the data set and with growing sequence length. We recognize the same relations for the values of P.
The average distance $\mu(d_{NN})$ (5.18) we have the lower bound of approximately $\frac{n}{2}$ and $\frac{10}{3}n$.
The mean value $\mu(\text{CPCC})$ is - quite roughly spoken - about $\frac{n}{2}$, where the best results are determined for the biggest data sets for each n.
The last column of Table D.1 lists the computational time. By increasing with growing sequence length and increasing size of data set, the computation time behaves as expected. Since a net is trained once and can then be applied as long as the problem parameters and the data do not change the computational time of a neural network is not as notable as that of a MIP.
For $n > 20$ the data sets NN-n_C-1-min lead to considerably worse results across all parameters than the greater sets. For $n > 30$ we rate the sets NN-n_C-1-max too small as well.

Test calculations with number of hidden units 20 and 200 as well as an word embedding layer of dimension 2 are summarized in Table D.2. For NN-20_4-2 we achieve improvements for *RS* and deterioration for all of the other parameters, except for the computational time[31].

The results have been generated with an Intel-i5 CPU, Matlab® version R2020b on a Lenovo notebook.

5.5. Limitations and applicability of the network

From the results evaluated in Section 5.4 we conclude that the considered neural network is not applicable for any instance of the presented problem. As we observed in Chapter 3 the problem is of very high complexity but at the same time it is not in the typical application area of neural networks (see Section 5.2). We identify the non-deterministic relation of the input and target data as a main factor for the discrepancy between problem and network in this case.
The quality of the results indicates that the problem requires at least a customized loss function. However, we assume that far more constraints have to be implemented in the network, analogically to how they are made designing the MIP.
We assume that the projection on color codes and, associated therewith, the non-distinctness of individual orders makes it more diffcult to recognize the structure in the data. Using order sequences instead of color sequences would blow up the size of the network by a multiple.
We end this section with considerations on inter- and intra-sequence relations. The kind of color codes and the number of each kind in an input sequence to be determined depends on the given output sequence. The order of color codes depends on the output sequence and the behavior of each color code. There is no intra-sequence relation. The $(k+1)$th color code is not based on the first k color codes in the input sequence but on the positions in the output sequence and the shifts. We assume that the network is not able to detect the problem structure based on non-deterministic shifts.

[31]The computational time increases with increasing NHU.

Further theoretical considerations on applicability and modifications of the neural network

We assume designing a loss function can possibly yield results making the neural network competitive with the MIP regarding the quality of results. As analyzed in Section 5.4, a very small part of the calculated sequences in the test sets are permutations and those have a significant distance to given input sequences. This leads to the conclusion that it is necessary to implement a problem-specific measure. However, it is not possible to derive the network loss function from the MIP objective function without adopting the simulation as well.

Neural networks show great potential for a range of application areas, which justifies the boost they are having in recent years. As for any method, it is crucial that the problem matches the method. We claim that the described problem does not fulfill this condition for a neural network. Even if a good quality of the results can be achieved by customizing the loss function, we conjecture a level of modeling and interpretation being similarly high as for the MIP. On the one hand, this affects the size of the net and thus the computational time, on the other hand, the aspect of using the neural network to reveal structure in the data fails.

6. Conclusion

All models are wrong, but
some models are useful.

(George Edward Pelham Box)

This work is dedicated to a non-deterministic MIP and solution approaches thereof. The structure and the complexity of the problem is tackled by a customized branch-and-bound algorithm and a sequence-to-sequence neural network. We mention just in sequence production in the automotive sector as a typical application of the presented problem.
The analysis of the real data provided by a specific use case results in the development and implementation of an algorithm for the generation of artificial test data. These form a basis for extensive computations.

The results provided by the branch-and-cut algorithm in the Gurobi^{TM}Optimizer show that the presented MIP is stable. The impact of the number of colors and the color code-specific distributions are pointed out and discussed in detail. The analysis of the results leads to the identification of the main factor for the long and strongly varying computational times. This knowledge about the problem structure goes beyond the solver's and is processed in a solution algorithm.

We developed a branch-and-bound algorithm with a customized pruning rule. Comparing the results for different parameter settings shows the benefit of the heuristic. The search space of the MIP is successfully reduced by a significant amount. We are confident that a hybrid of the customized pruning rule and a solution algorithm in a state-of-the-art solver such as Gurobi^{TM}can achieve the desired reduction of computational time.

Several possibilities of adjustment for potential use cases offered by the MIP are presented. Due to its non-deterministic character, the model can only be tested and tuned for a use case during operation. Mathematical intuition supported by the results allow considerations and recommendations, such as parameter tuning and weighting.

The processing of the raw data via a sequence-to-sequence neural network and the results thereof reflect the complex and non-deterministic structure of the problem. We are confident that the result quality can be improved by modifications of the network and the loss function, in particular. It is pointed out that this problem requires to balance the benefit of learning from raw data and problem interpretation processed in network modifications. Despite the great interest neural networks experience over the last decades it is important to bear the suitability of the problem in mind. From our research we conclude that the optimization approach can be considered superior to the sequence-to-sequence neural network for this specific problem.

We finish this work by returning to Henry Ford. A black-cars-only policy would simplify controlling the paint shop exceedingly. However, lively discussions, profound thought experiments, several misperceptions and numerous moments of enthusiasm would have been denied to us. This research highly encouraged our understanding and delight for optimization theory and deep learning.

7. Bibliography

[Ach07] Tobias Achterberg. “Constraint Integer Programming”. PhD thesis. Technische Universität Berlin, 2007.

[AKM05] Tobias Achterberg, Thorsten Koch, and Alexander Martin. “Branching rules revisited”. In: *Operations Research Letters* 33 (2005), pp. 42–54.

[APR14] Ilan Adler, Christos Papadimitriou, and Aviad Rubinstein. “On Simplex Pivoting Rules and Complexity Theory”. In: *Integer Programming and Combinatorial Optimization*. Ed. by Jon Lee and Jens Vygen. Cham: Springer International Publishing, 2014, pp. 13–24. ISBN: 978-3-319-07557-0.

[Art99] Benno Artmann. *Euclid - The Creation of Mathematics*. Springer New York, 1999.

[Ath05] Christos Athanasiadis. “Ehrhart polynomials, simplicial polytopes, magic squares and a conjecture of Stanley”. In: *Journal für reine und angewandte Mathematik* 583 (2005), pp. 163–174.

[Bad+18] Jörg Bader et al. “Mixed integer reformulations of integer programs and the affine TU-dimension of a matrix”. In: *Mathematical Programming* 169 (2018), pp. 565–584.

[Bau+09] Barbara Baumeister et al. “On permutation polytopes”. In: *Advances in Mathematics* (2009).

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473 (2015).

[BGT81] Robert Bland, Donald Goldfarb, and Michael Todd. “The Ellipsoid Method: A Survey”. In: *Operations Research* 29.6 (1981).

[Bir46] George Birkhoff. “Tres observaciones sobre el algebra lineal”. In: *Universidad Nacional de Tucuman Revista, Serie A* 5 (1946), pp. 147–151.

[BM98] Brian Borchers and John Mitchell. *Using an Interior point Method in a Branch and Bound Algorithm for Integer Programming*. 1998.

[BP03] Matthias Beck and Dennis Pixton. *The volume of the* 10*th Birkhoff polytope*. 2003.

[Bra+91] Jeremy Brandman et al. “Convex hulls of Coxeter groups”. In: *Mathematics Subject Classification* (1991).

[Bro83] Arne Brondsted. *An Introduction to Convex Polytopes*. Springer-Verlag New York, 1983.

[BS96] Louis Billera and Aravamuthan Sarangarajan. “All 0/1-polytopes are traveling salesman polytopes”. In: *Combinatorica* 16.2 (1996), pp. 175–188.

[Bys+20] Sara Bysko et al. “Automotive Paint Shop 4.0”. In: *Computers & Industrial Engineering* 139 (2020).

[CCZ10] Michele Conforti, Gérard Cornuójols, and Giacomo Zambelli. “Extended formulations in combinatorial optimization”. In: *4OR* 8.1 (2010), pp. 1–48.

[CCZ14] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer International Publishing, 2014, pp. 85–128.

[Cha+19] Aditya Chawla et al. “Demand Forecasting Using Artificial Neural Networks - A Case Study of American Retail Corporation”. In: *Advances in Intelligent Systems and Computing*. Ed. by Natalya Shakhoyska and Mykola Medykovskyy. Springer, 2019, pp. 79–89.

[CJ05] Hadrien Cambazard and Narendra Jussien. “Integrating Benders Decomposition Within Constraint Programming”. In: *Principles and Practice of Constraint Programming - CP 2005*. 2005, pp. 752–756.

[CM07] E. Rodney Canfield and Brendan D. McKay. *The asymptotic volume of the Birkhoff polytope*. 2007.

[Dan63] George Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[Dax97] Achiya Dax. “An Elementary Proof of Farkas’ Lemma”. In: *SIAM Review* 39.3 (1997), pp. 503–507.

[DKM06] Andreas Drexl, Alf Kimms, and Lars Matthießen. “Algorithms for the car sequencing and the level scheduling problem”. In: *Journal of Scheduling* 9 (2006), pp. 153–176.

[DSA98] Amal De Silva and David Abramson. “A parallel Interior Point Method and its application to Facility Location Problems”. In: *Computational Optimization and Applications* 9 (1998), pp. 249–273.

[Fad15] Salman Fadaei. “Mechanism Design via Dantzig-Wolfe Decomposition”. In: *CoRR* (2015).

[FB08] Malte Fliedner and Nils Boysen. “Solving the car sequencing problem via Branch & Bound”. In: *European Journal of Operational Research* 191.3 (2008), pp. 1023–1042.

[FCB16] Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. “Multi-Way, Multilingual Neural Machine Tranlation with a Shared Attention Mechanism”. In: 2016, pp. 866–875.

[FHZ11] Oliver Friedmann, Thomas Hansen, and Uri Zwick. “Subexponential lower bounds for randomized pivoting rules for solving linear programs”. In: *Proceedings of the forty-third annual ACM symposium on Theory of computing* (2011), pp. 283–292.

[Fis04] Marshall Fisher. “The Lagrangian Relaxation Method for solving Integer Programming problems”. In: *Management Science* 50.12 (2004), pp. 1861–1871.

[Fog+13] Fajwel Fogel et al. “Convex Relaxations for Permutation Problems”. In: *Advances in Neural Information Processing Systems* (2013).

[Fri11] Oliver Friedmann. “A Subexponential Lower Bound for Zadeh’s Pivoting Rule for Solving Linear Programs and Games”. In: *Integer Programming and Combinatoral Optimization*. Ed. by Oktay Günlük and Gerhard J. Woeginger. Springer Berlin Heidelberg, 2011, pp. 192–206.

[FT94] Alan Frieze and Shang-Hua Teng. “On the complexity of computing the diameter of a polytope”. In: *Comput Complexity* 4 (1994), pp. 207–219.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GL81] Peter Gács and Laszlo Lovász. “Khachiyan’s algorithm for linear programming”. In: *Mathematical Programming at Oberwolfach*. Ed. by H. König, B. Korte, and K. Ritter. Springer, Berlin, 1981, pp. 61–68.

[GLS88] Michel Groetschel, Laszlo Lovász, and Alexander Schrijver. “The Ellipsoid Method”. In: *Geometric Algorithms and Combinatorial Optimization*. Vol. 2. Springer, Berlin, 1988, pp. 64–101.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6645–6649.

[Goe15] Michel Goemans. “Smallest compact formulation for the permutahedron”. In: *Mathematical Programming* 153 (2015), pp. 5–11.

[GP06] Robert Guralnick and David Perkinsons. “Permutation polytopes and indecomposable elements in permutation groups”. In: *Journal of Combinatorial Theory, Series A* (2006).

[Gra12] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Vol. 385. Studies in Computational Intelligence. Springer, Berlin, Heidelberg, 2012.

[Gru13] Branko Gruenbaum. *Convex Polytopes*. Ed. by Günther M. Ziegler. Springer-Verlag New York, 2013.

[Gura] *Gurobi^{TM}Optimizer Reference Manual*. Version 9.0. Accessed: 2021-02-13. Gurobi$^{\mathrm{TM}}$Optimization. 2020.

[Gurb] *MIP Starts*. `https://www.gurobi.com/documentation/9.1/examples/mip_starts.html`. Accessed: 2021-02-04.

[He+14] Xing He et al. “A Recurrent Neural Network for Solving Bilevel Linear Programming Problem”. In: *IEEE Transactions on Neural Networks and Learning Systems* 25.4 (2014), pp. 824–830.

[HK70] Michael Held and Richard Karp. “The Traveling-salesman Problem and minimum spanning trees”. In: *Operations Research* 18 (1970), pp. 1138–1162.

[HKL04] Marcel Herzog, Gil Kaplan, and Arieh Lev. “Representation of permutations as products of two cycles”. In: *Discrete Mathematics* 285 (2004), pp. 323–327.

[Hop84] John Hopfield. “Neurons with graded response have collective computational properties like those of two-state neurons”. In: *Proceedings of the National Academy of Science* (1984).

[Hor91] Kurt Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Networks* 4 (1991), pp. 251–257.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.

[HT85] John Hopfield and David Tank. “Neural Computation of Decisions in Optimization Problems”. In: *Biological Cybernetics* 52 (1985), pp. 141–152.

[Hur] Glenn Hurlbert. “A short Proof of the Birkhoff-Von Neumann Theorem”.

[JA05] B.A. Jensen and János Abonyi. “Neural networks for process modeling”. In: *Instrument Engineers Handbook, Fourth Edition: Process Control and Optimization*. Ed. by Béla Lipták. CRC Press, 2005, pp. 253–264.

[JAS02] Gonzola Joya, Miguel Atencia, and Francisco Sandoval. “Hopfield neural networks for optimization: study of different dynamics”. In: *Neurocomputing* 43 (2002), pp. 219–237.

[JKM04] Colin Jones, Eric Kerrigan, and Jan Maciejowski. *Equality Set Projection: A new algorithm for the projection of polytopes in halfspace representation*. Tech. rep. CUED/F-INFENG/TR.463, 2004.

[JNS00] Ellis Johnson, George Nemhauser, and Martin Savelsbergh. “Progress in Linear Programming-Based Algorithms for Integer Programming: An Exposition”. In: *Informs Journal on Computing* 12.1 (2000).

[Kai11a] Volker Kaibel. “Basic Polyhedral Theory”. In: *Wiley Encyclopedia of Operations Research and Management Science* (2011).

[Kai11b] Volker Kaibel. “Extended Formulations in Combinatorial Optimization”. In: *Optima 85* (2011).

[Kar84] Narendra Karmarkar. “A New Polynomial-Time Algorithm for Linear Programming-II”. In: *Combinatorica* 4.4 (Dec. 1984), pp. 373–395. DOI: 10.1007/BF02579150.

[Kha79] Leonid Khachiyan. “A polynomial algorithm in linear programming”. In: *Dokl. Akad. Nauk SSSR* 244.5 (1979), pp. 1093–1096.

[Kis04] Tamás Kis. “On the complextity of the car sequencing problem”. In: *Operations Research Letters* 32.4 (2004), pp. 331–335.

[KK92] Gil Kalai and Daniel Kleitman. “A quasi-polynomial bound for the diameter of graphs of polyhedra”. In: *Bulletin of the American Mathematical Society* 26 (1992).

[KPKP90] Behzad Kamgar-Parsi and Behrooz Kamgar-Parsi. “On Problem Solving with Hopfield Neural Networks”. In: *Biological Cybernetics* 62 (1990), pp. 415–423.

[KS09] Edward Kim and Francisco Santos. “An Update on the Hirsch Conjecture”. In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* (2009).

[KW67] Victor Klee and David Walkup. “The d-step conjecture for polyhedra of dimension d”. In: *Acta Mathematica* 117 (1967), pp. 53–78.

[LAR12] Thaynara Arielly de Lima and Mauricio Ayala-Rincón. “Complexity of Cayley Distance and other General Metrics on Permutation Groups”. In: *7th Colombian Computing Congress (CCC)*. 2012.

[Law76] Eugene Lawler. *Combinatorial Optimization. Networks and Matroids*. Holt, Rinehart and Winston, 1976.

[Les+93] Moshe Leshno et al. “Multilayer Feedforward Networks with a Nonpolynomial Activation Function can approximate any Function”. In: *Neural Networks* 6 (1993), pp. 861–867.

[LLY09] Jesús Loera, Fu Liu, and Ruriko Yoshida. “A generating function for all semi-magic squares and the volume of the Birkhoff Polytope”. In: *Journal of Algebraic Combinatorics* 30 (2009), pp. 113–139.

[LMS99] George Lendaris, Karl Mathia, and Richard Saeks. “Linear Hopfield networks and constrained optimization”. In: *IEEE Transactions on Cybernetics* 29 (1999), pp. 114–118.

[Loe13] Edward Loera Jesús Kim. “Combinatorics and Geometry of Transportation Polytopes: an Update”. In: *Contemporary Mathematics* 625 (2013).

[LPM15] Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2015, pp. 1412–1421. URL: https://www.aclweb.org/anthology/D15-1166.

[LRZ06] Udo Lindemann, Ralf Reichwald, and Michael Zäh. *Individualisierte Produkte - Komplexität beherrschen in Entwicklung und Produktion*. Springer, 2006.

[LW14] Cong Lim and Stephen Wright. “Beyond the Birkhoff Polytope: Convex Relaxations for Vector Permutation Problems”. In: *Advances in Neural Information Processing Systems* 3 (2014).

[Mac03] David MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[Mat98] Satoshi Matsuda. “Optimal Hopfield Network for Combinatorial Optimization with Linear Cost Function”. In: *IEEE Transactions on Neural Networks* 9.6 (1998), pp. 1319–1330.

[MG07] Jirí Matousek and Bernd Gärtner. *Understanding and Using Linear Programming*. Springer-Verlag Berlin Heidelberg, 2007.

[Mit96] John Mitchell. “Interior Point Algorithms for Integer Programming”. In: *Advances in Linear and Integer Programming*. Ed. by John Beasley. Clarendon Press, 1996, pp. 223–248.

[Mor+16] David Morrison et al. “Branch-and-bound algorithms:A survey of recent advances in searching, branching and pruning”. In: *Discrete Optimization* 19 (2016), pp. 79–102.

[MPR98] John Mitchell, Panos Pardalos, and Mauricio Resende. “Interior Point Methods for Combinatorial Optimization”. In: *Handbook of Combinatorial Optimization*. Ed. by Ding-Zhu Du and Panos Pardalos. Vol. 1. Kluwer Academic Publishers, 1998, pp. 189–297.

[NT09] Arkadi Nemirovski and Michael Todd. “Interior-point methods for optimization”. In: *Acta Numerica* (2009).

[Onn93] Shmuel Onn. “Geometry, Complexity and Combinatorics of Permutation Polytopes”. In: *Journal of Combinatorial Theory* (1993), pp. 31–49.

[Pak00] Igor Pak. “Four Questions on Birkhoff Polytope”. In: *Annals of Combinatorics 4* (2000).

[PKW86] Bruce Parello, Waldo Kabat, and Larry Wos. “Job-scheduling using automated reasoning: A case study of the car-sequencing problem”. In: *Journal of Automated Reasoning* 2 (1986), pp. 1–42.

[Pla18] Elad Plaut. *From Principal Subspaces to Principal Components with Linear Autoencoders*. 2018.

[Pot93] Jean.-Yves. Potvin. “State-of-the-Art Survey - The Traveling Salesman Problem: A Neural Network Perspective”. In: *Journal on Computing* 5.4 (1993), pp. 328–348.

[PR08] Matthias Prandtstetter and Günther Raidl. “An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem”. In: *European Journal of Operational Research* 191.3 (2008), pp. 1004–1022.

[Reb09] Steffen Rebennack. “Ellipsoid Method”. In: *Encyclopedia of Optimization*. Ed. by C Floudas and P. Pardalos. Springer US, 2009, pp. 890–899.

[SA20] Simone Sharma and Anidhya Athaiya. “Activation functions in neural networks”. In: *International Journal of Engineering, Applied Sciences and Technology* 4.12 (2020), pp. 310–316.

[Seq] *SeqToSeq*. `https://de.mathworks.com/help/deeplearning/ug/sequence-to-sequence-translation-using-attention.html`. Accessed: 2021-02-11.

[SG00] Kate Smith and Jatinder Gupta. “Neural networks in business: techniques and applications for the operations researcher”. In: *Computers & Operations Research* 27.11-12 (2000), pp. 1023–1044.

[She+19] Li Shen et al. “Deep Learning to Improve Breast Cancer Detection on Screening Mammography”. In: *Scientific Reports* 9 (2019), pp. 1–12.

[SM99] Kate Smith-Miles. “Neural Networks for Combinatorial Optimization: A Review of more than a Decade of Research”. In: *Informs Journal on Computing* 11.1 (1999), pp. 15–34.

[Sol+08] Christine Solnon et al. “The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF’2005 challenge problem”. In: *European Journal of Operational Research* 191.3 (2008), pp. 912–927.

[ST04] Daniel Spielman and Shang-Hua Teng. “Smoothed Analysis of Algorithms: Why the Simplex Algorithm usually takes polynomial time”. In: *Journal of the ACM* 51.3 (2004), pp. 385–463.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc Le. “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* (2014).

[SZ71] Naum Shor and N Zhurbenko. “The minimization method using space dilatationin direction of difference of two sequential gradients”. In: *Kibernetika* 3 (1971), pp. 51–59.

[Tod02] Michael Todd. “The many facets of linear programming”. In: *Mathematical Programming* (2002), pp. 417–436.

[Van14] Robert Vanderbei. *Linear Programming. Foundations and Extensions*. International Series in Operations Research & Management Science. Springer US, 2014.

[Vas+17] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[VS06] François Vanderbeck and Martin Savelsbergh. “A generic view of Dantzig–Wolfe decomposition in mixed integer programming”. In: *Operations Research Letters* 34 (2006), pp. 296–306.

[VW10] François Vanderbeck and Laurence Wolsey. “Reformulation and Decomposition of Integer Programs”. In: *50 Years of Integer Programming 1958-2008*. Jünger, M. et al., 2010, pp. 431–502.

[Wor] *WordEmbedding*. https://www.mathworks.com/help/textanalytics/ref/nnet.cnn.layer.wordembeddinglayer.html. Accessed: 2021-03-26.

[Wri97] Stephen Wright. *Primal-dual Interior-Point Methods*. SIAM, 1997.

[XFW08] Youshen Xia, Gang Feng, and Jun Wang. “A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints”. In: *IEEETransactions on Neural Networks* 19.8 (2008), pp. 1340–1353.

[YN76a] David Yudin and Arkadi Nemirovski. “Evaluation of the Informational Complexity of Mathematical Programming Problems”. In: *Èkon Math Metod* 12 (1976), pp. 128–142.

[YN76b] David Yudin and Arkadi Nemirovski. “Informational Complexity and Efficient Methods for the Solution of Convex Extremal Problems”. In: *È Math Metod* 12 (1976), pp. 357–369.

[YN77] David Yudin and Arkadi Nemirovski. “Optimization Methods Adapting to the “Significant” Dimension of the Problem”. In: *Autom Telemekhanika* 38 (1977), pp. 75–87.

[YWL08] Lean Yu, Shouyang Wang, and Kin Keung Lai. “Credit risk assessment with a multistage neural network ensemble learning approach”. In: *Expert Systems with Applications* 34 (2008), pp. 1434–1444.

[Zie01] Günther Ziegler. “Questions about Polytopes”. In: *Mathematics Unlimited - 2001 and Beyond*. Ed. by Björn Engquist and Wilfried Schmid. Springer Berlin Heidelberg, 2001, pp. 1195–1211.

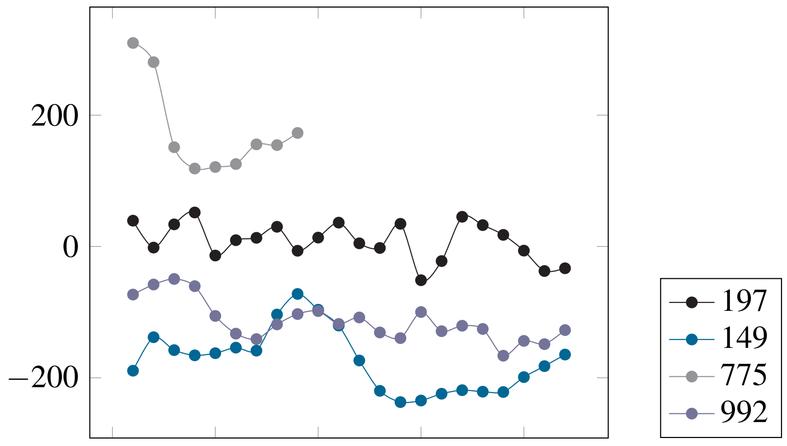
[Zie12a] Günther Ziegler. *Lectures on Polytopes*. Springer-Verlag New York, 2012.

[Zie12b] Günther. Ziegler. “Who solved the Hirsch conjecture?” In: *Documenta Mathematica* (2012), pp. 75–85.

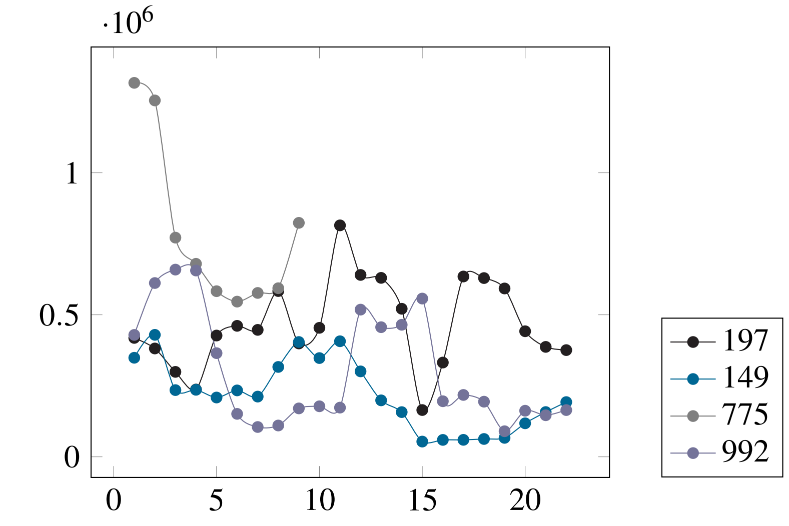
A. Analysis of real and artificial data

All figures in Section A.1 as well as Sections A.2.1 to A.2.2 are supposed to carry the label “subset” on the x-axis and “mean(variance/skewness) value” on the y-axis.
The x-axes of the figures in Section A.2.3 are supposed to be labeled “shift” and the y-axes “probability”.
Due to reason of clearity we omit the axis labels, they are clear from the figure captions.

A.1. Statistical analysis of real data

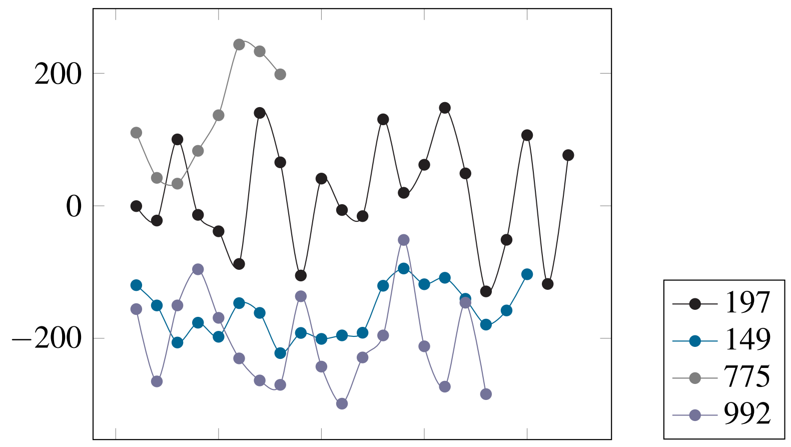


(a) mean values

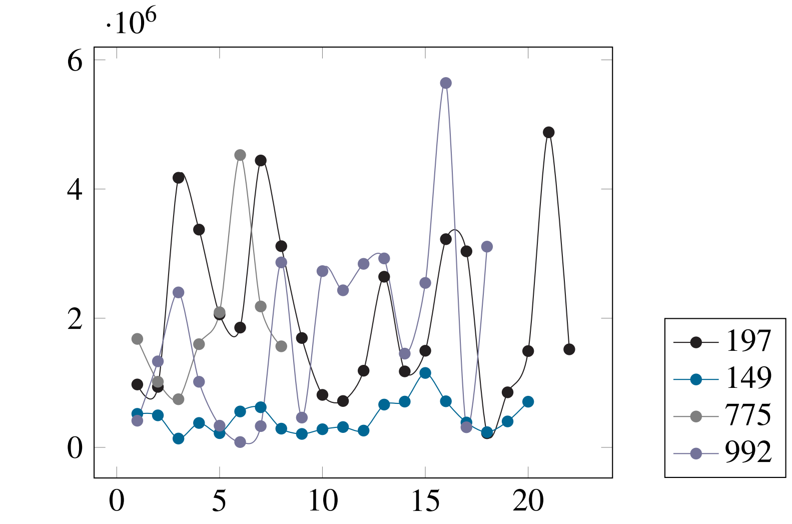


(b) variance values

Figure A.1.: Statistic values for HD-1

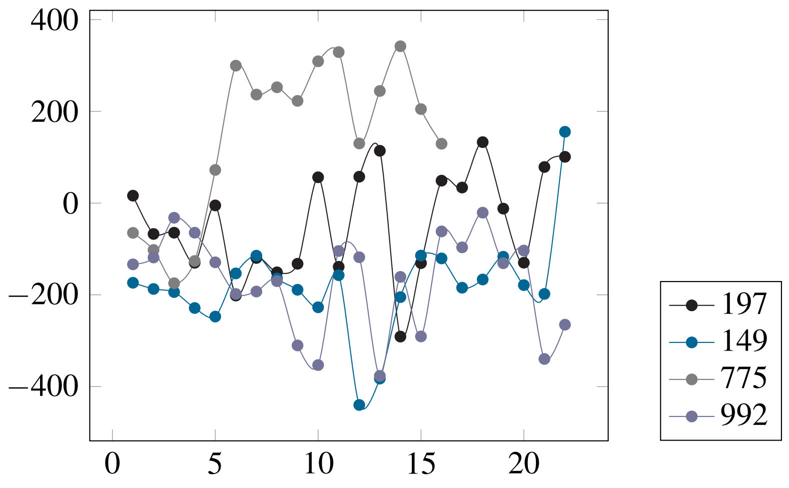


(a) mean values

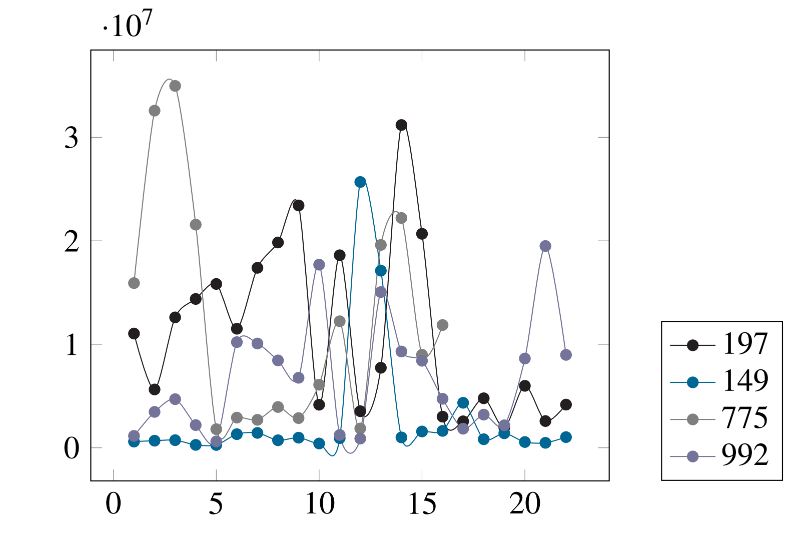


(b) variance values

Figure A.2.: Statistic values for HD-2



(a) mean values

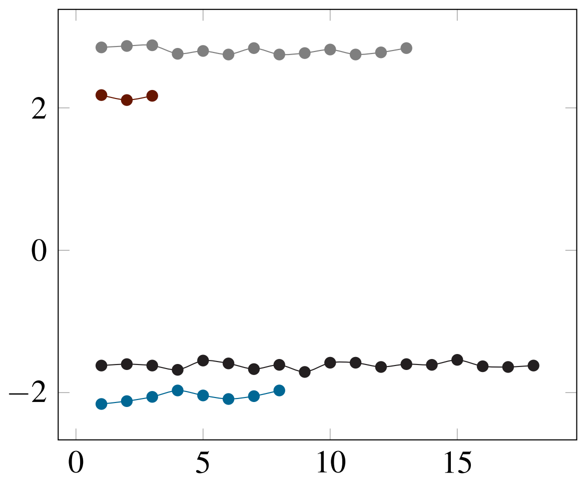


(b) variance values

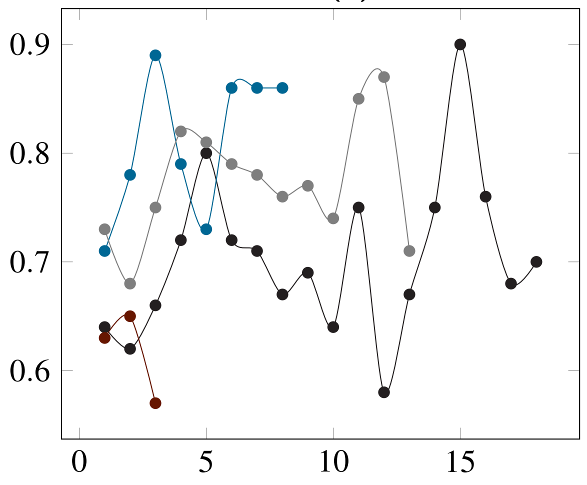
Figure A.3.: Statistic values for HD-3

A.2. Statistical analysis of artificial data

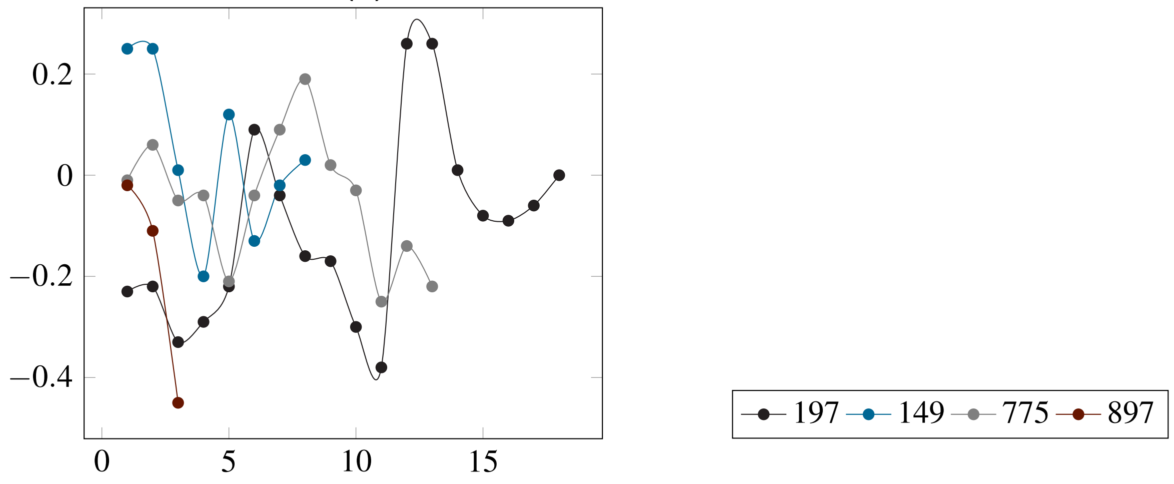
A.2.1. Moving statistic values for mono-distributed sets with n = 10, 50, 100, 200



(a) mean values

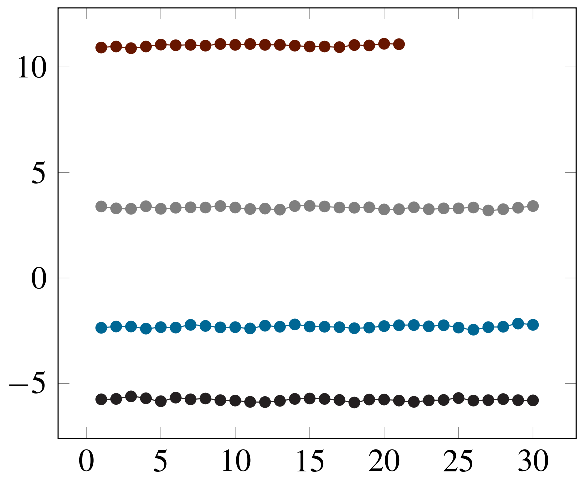


(b) variance values

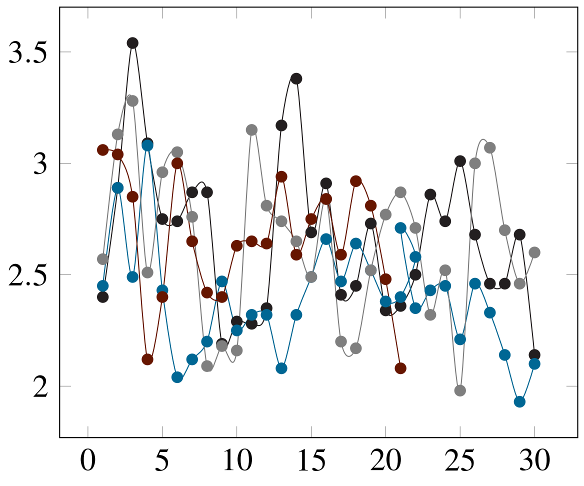


(c) skewness values

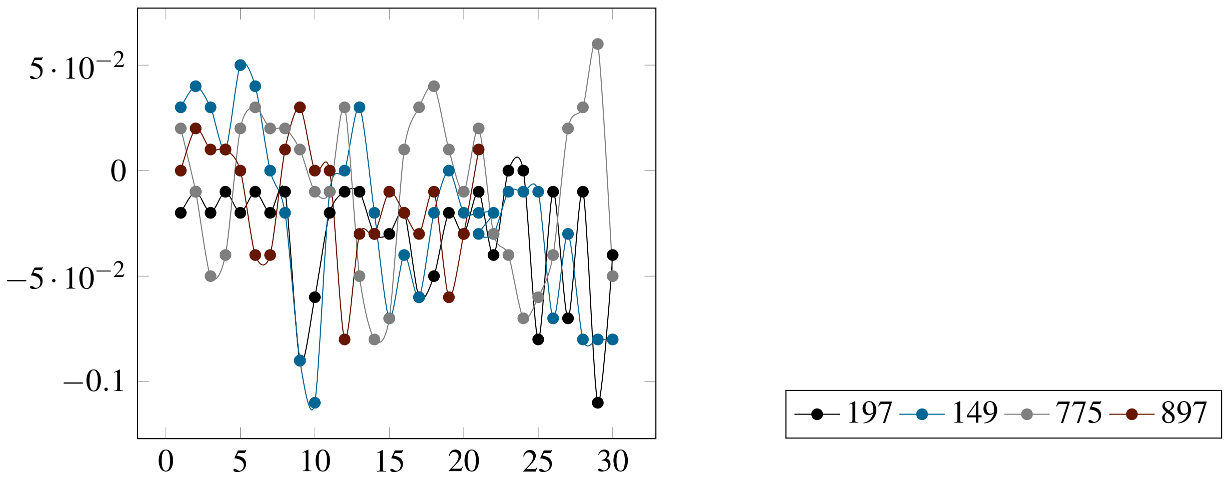
Figure A.4.: Statistic values for 10_4-1



(a) mean values

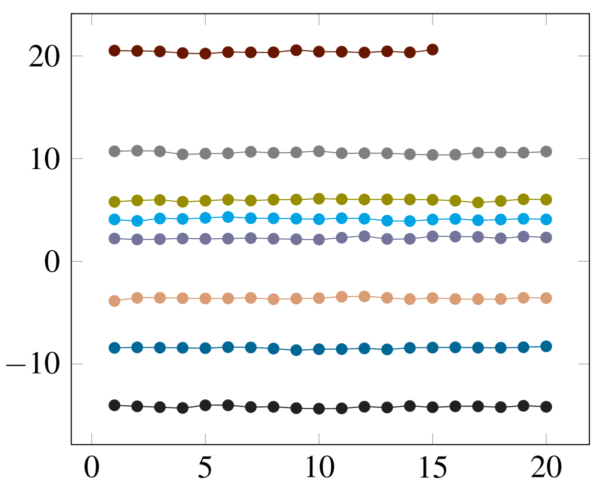


(b) variance values

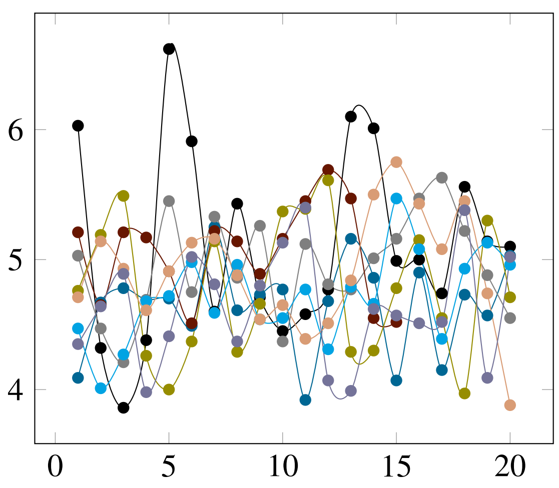


(c) skewness values

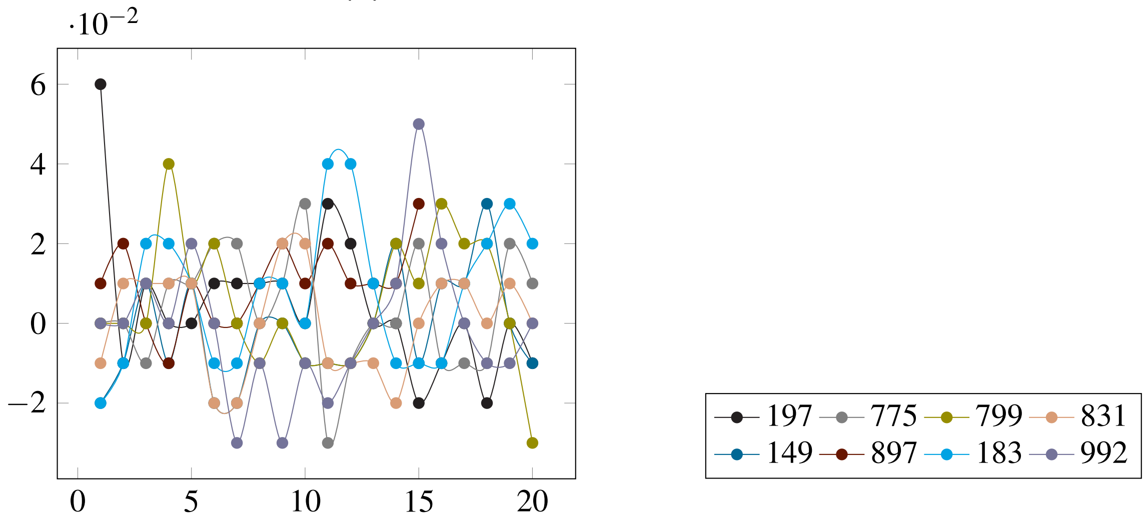
Figure A.5.: Statistic values for 50_4-1



(a) mean values

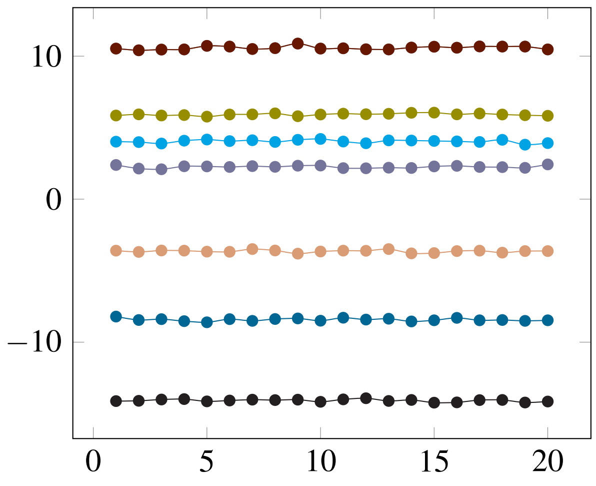


(b) variance values

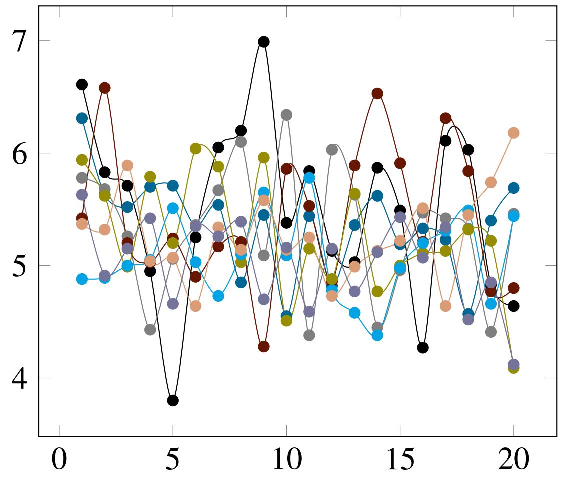


(c) skewness values

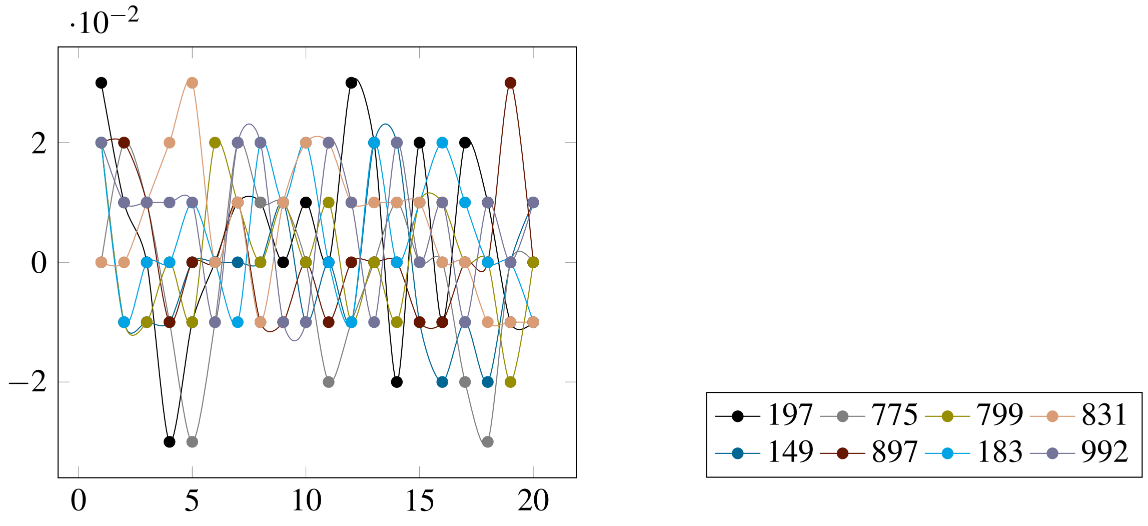
Figure A.6.: Statistic values for 100_8-1



(a) mean values



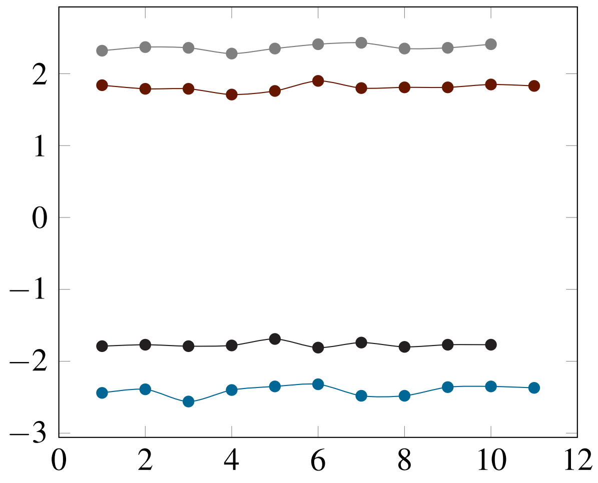
(b) variance values



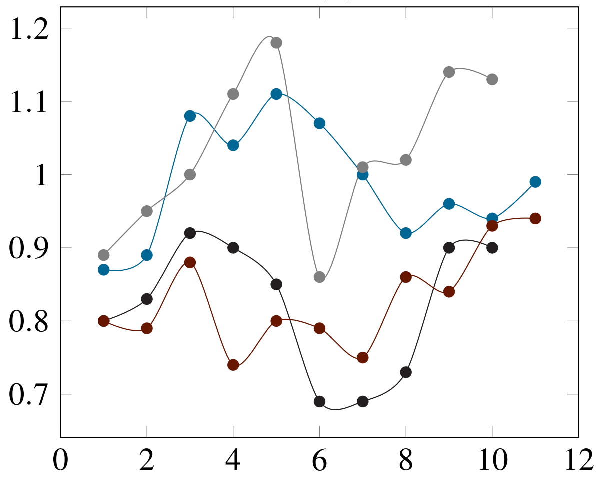
(c) skewness values

Figure A.7.: Statistic values for 200_8-1

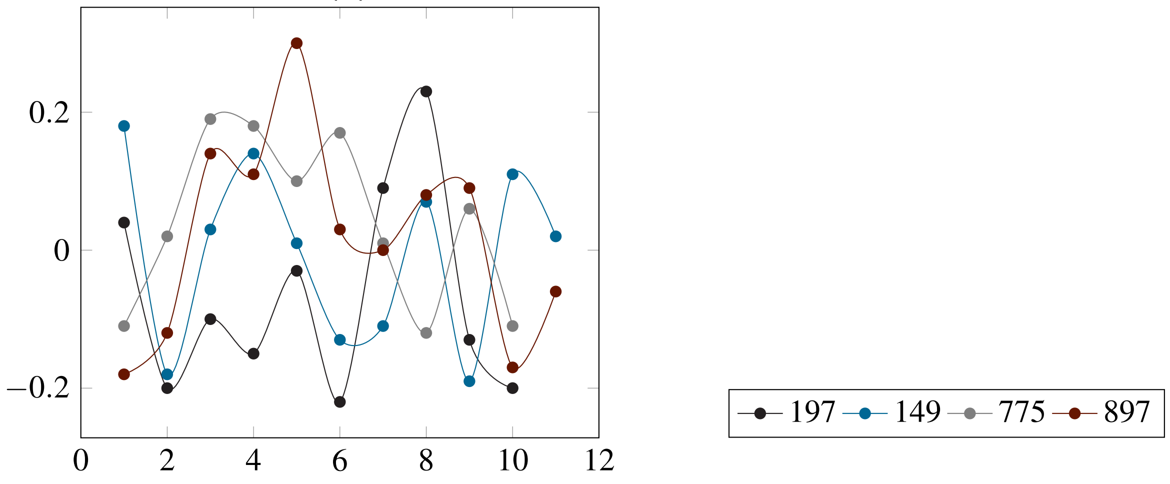
A.2.2. Moving statistic values for poly distributed sets with n = 10, 50, 100, 200



(a) mean values

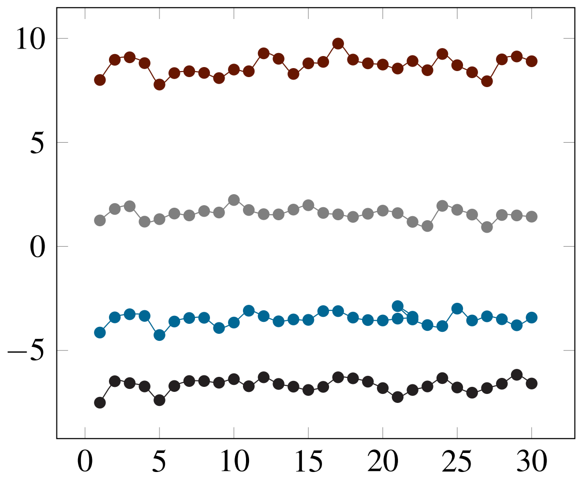


(b) variance values

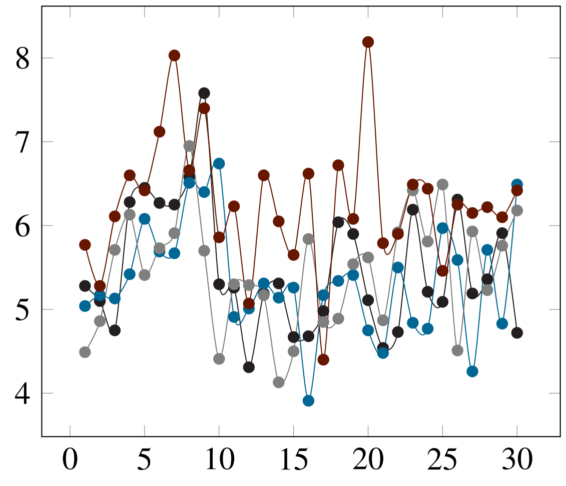


(c) skewness values

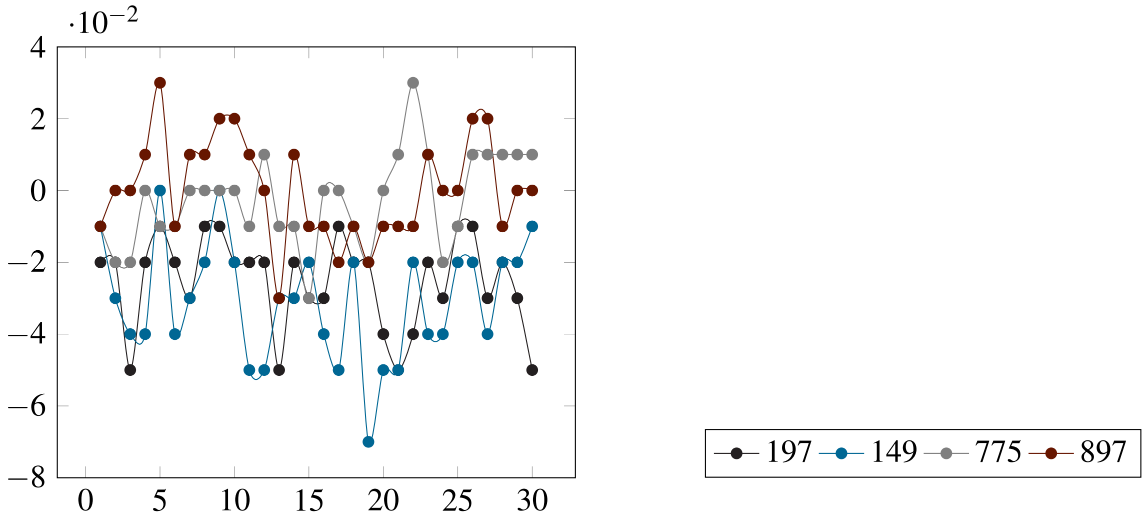
Figure A.8.: Statistic values for 10_4-2



(a) mean values

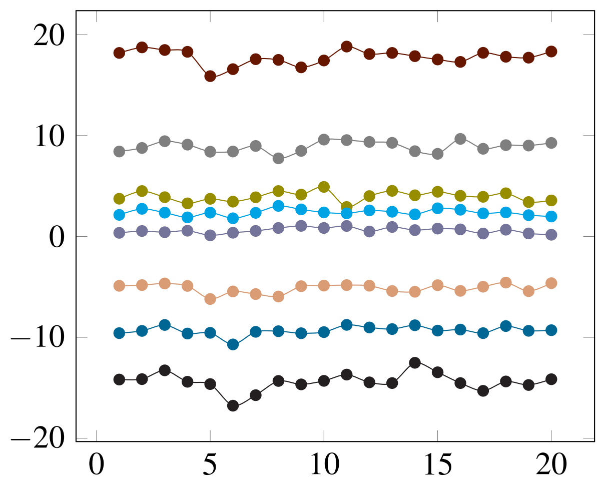


(b) variance values

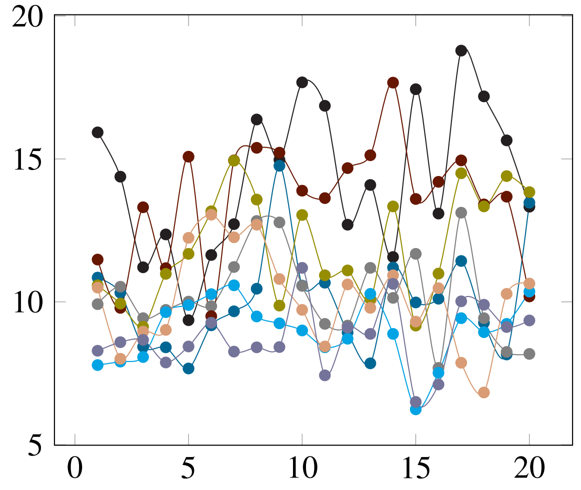


(c) skewness values

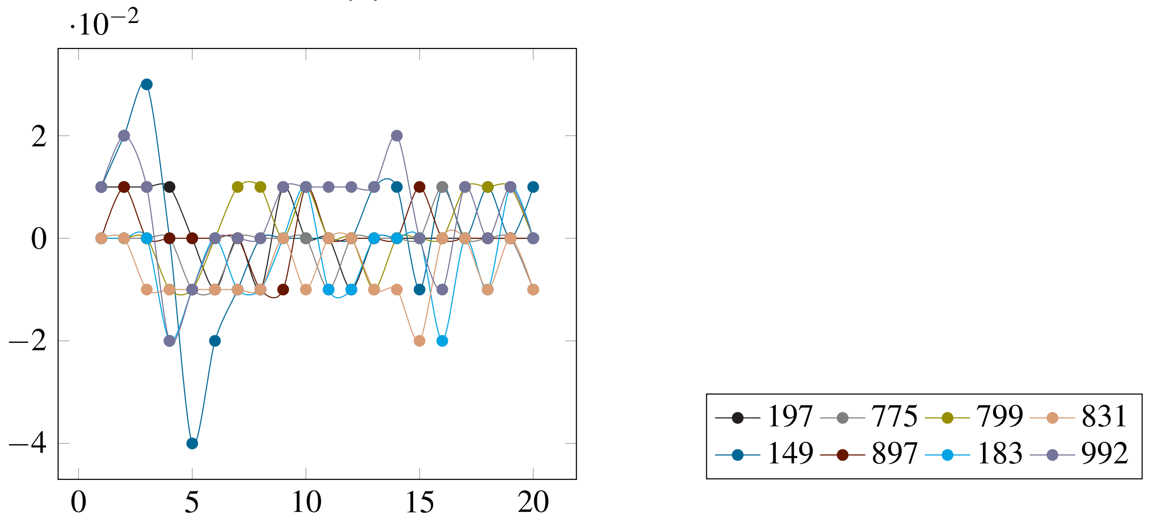
Figure A.9.: Statistic values for 50_4-2



(a) mean values

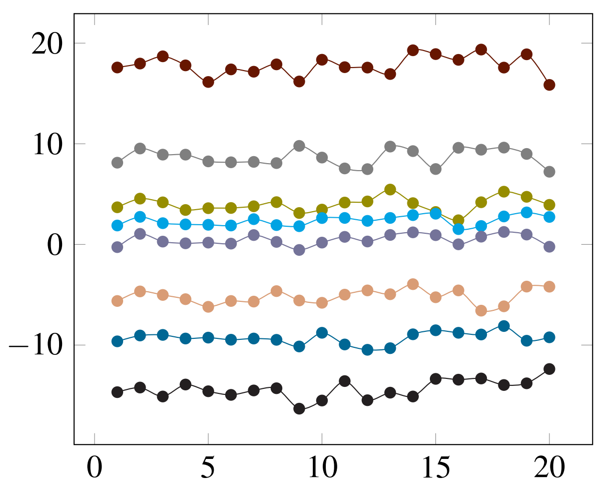


(b) variance values

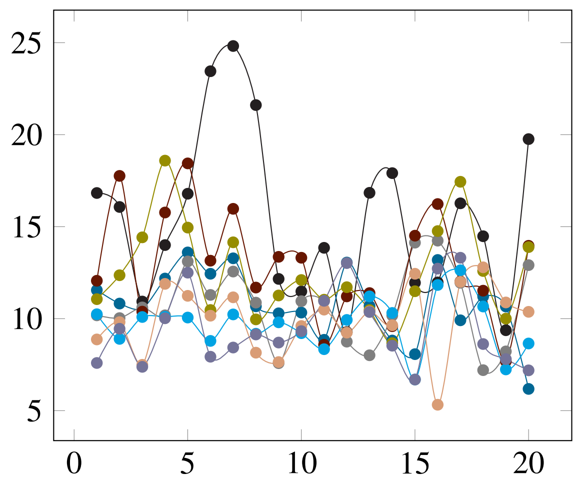


(c) skewness values

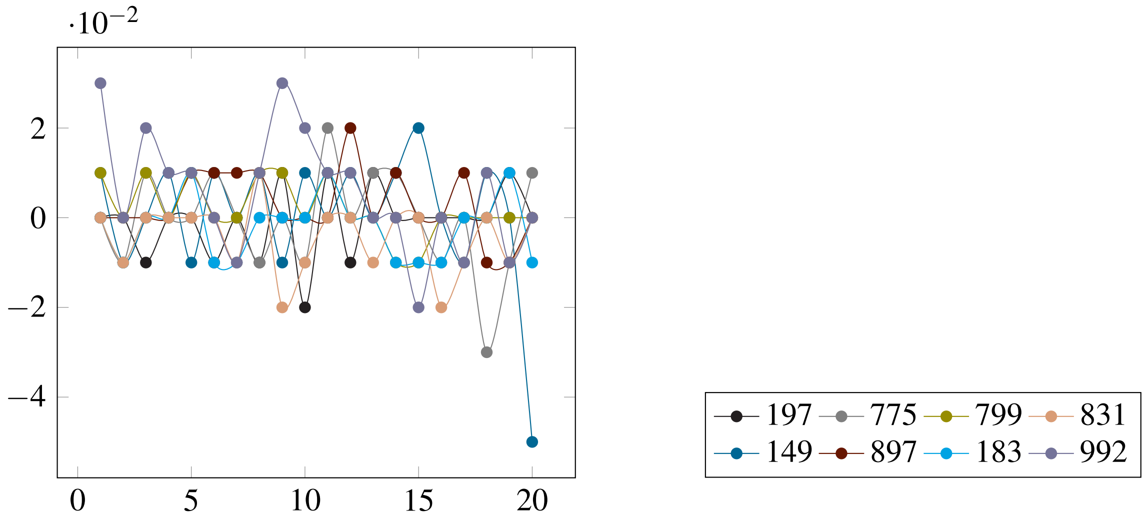
Figure A.10.: Statistic values for 100_8-2



(a) mean values



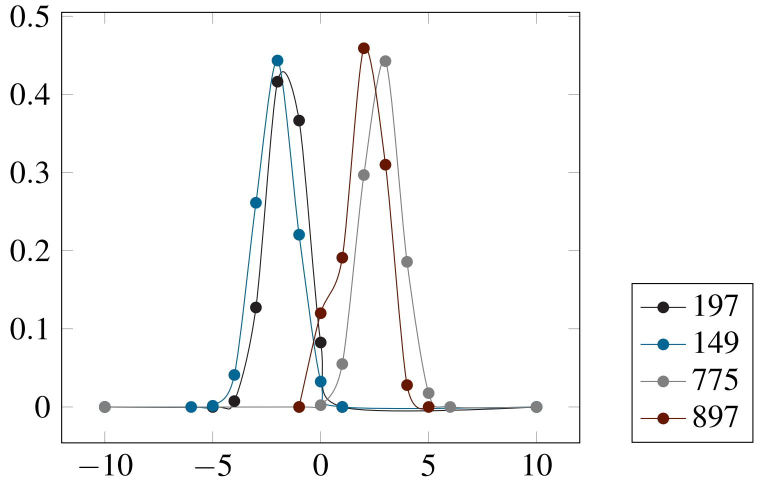
(b) variance values



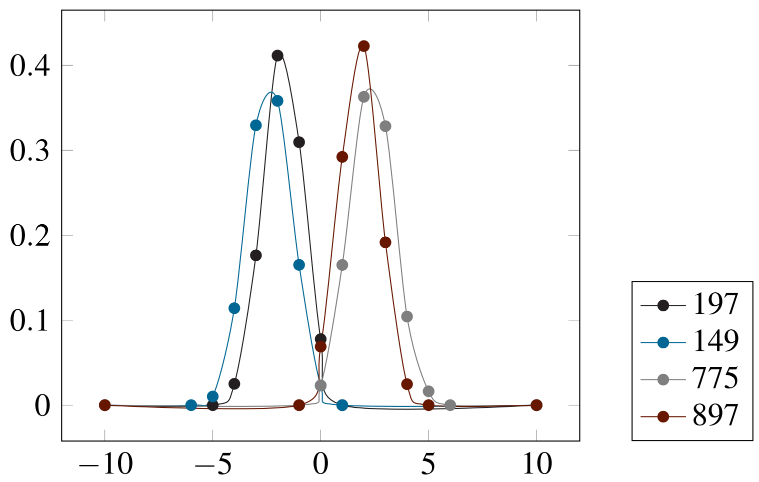
(c) skewness values

Figure A.11.: Statistic values for 200_8-2

A.2.3. Probability distributions for n = 10, 50, 100, 200

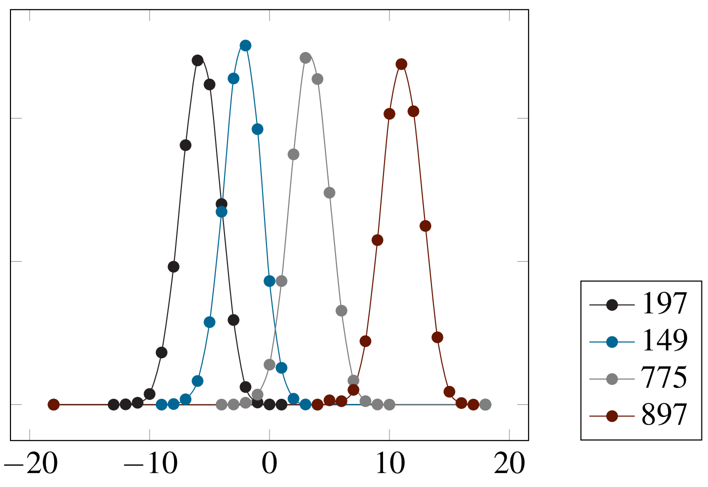


(a) 10_4-1

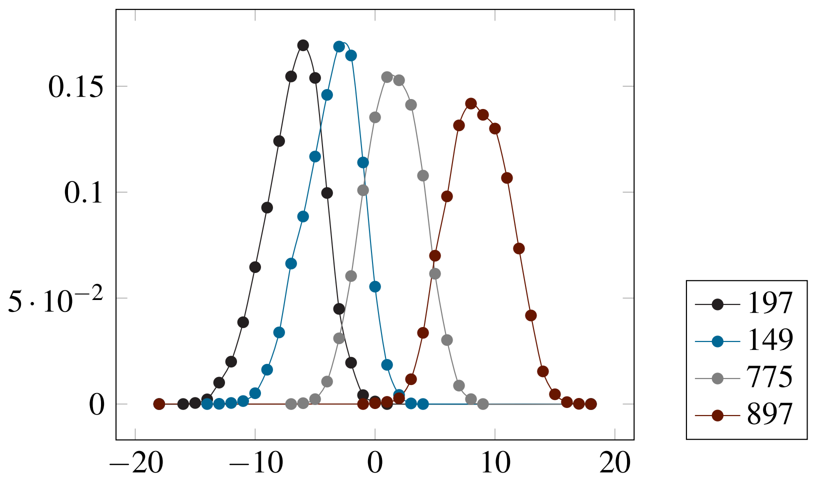


(b) 10_4-2

Figure A.12.: Probability distributions for 10_4-1 and 10_4-2

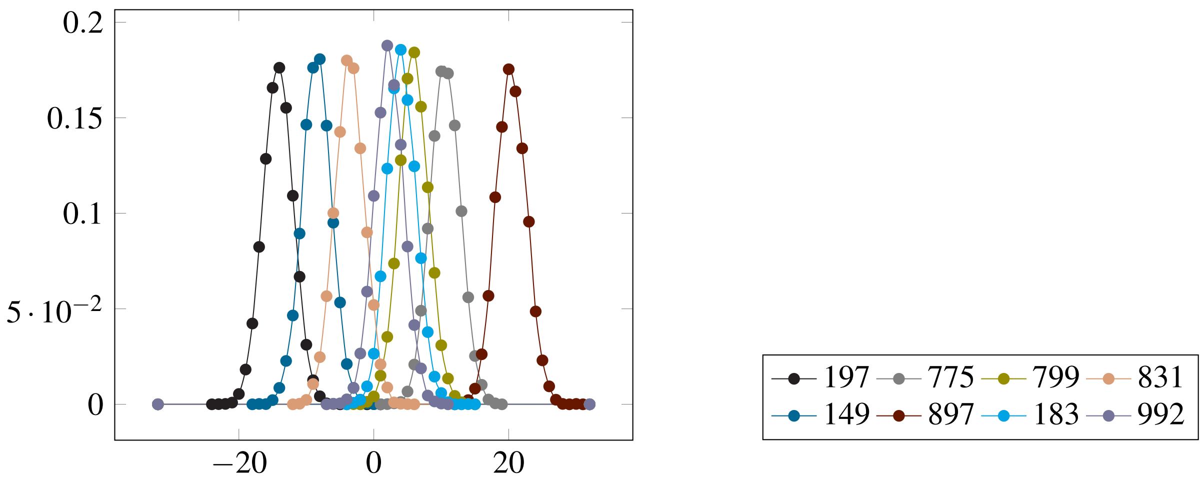


(a) 50_4-1

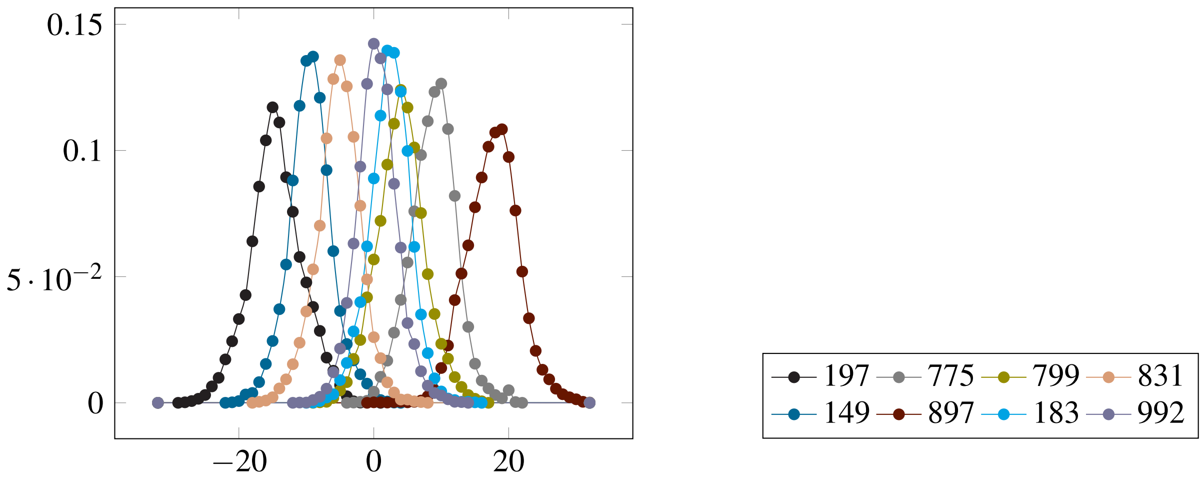


(b) 50_4-2

Figure A.13.: Probability distributions for 50_4-1 and 50_4-2

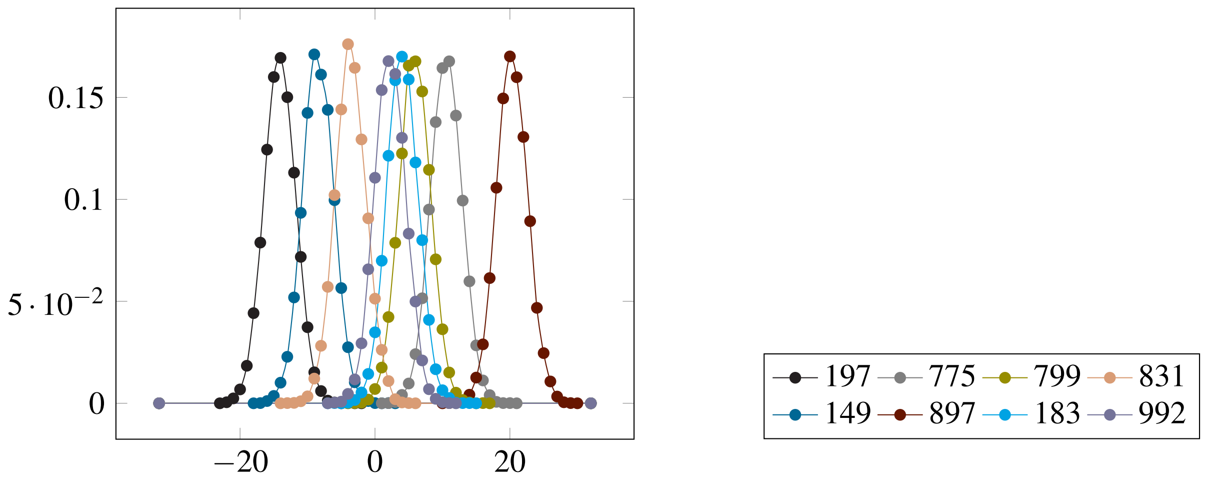


(a) 100_8-1

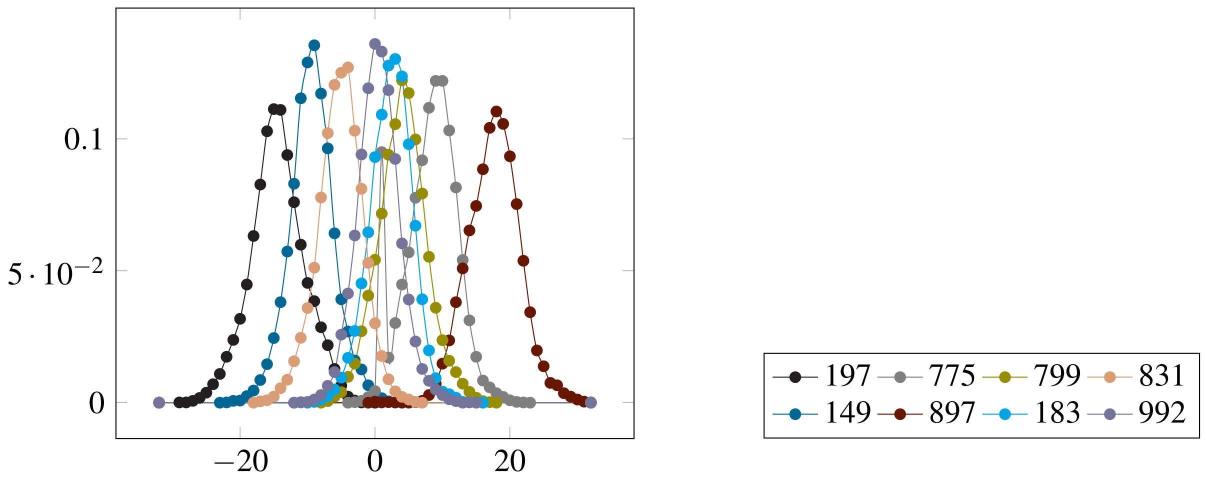


(b) 100_8-2

Figure A.14.: Probability distributions for 100_8-1 and 100_8-2



(a) 200_8-1



(b) 200_8-2

Figure A.15.: Probability distributions for 200_8-1 and 200_8-2

B. Computational results of the MIP

B.1. Gurobi Optimizer without start solution

The columns of Table B.1 and B.3 are defined as follows

Problem - problem in the usual notation

MIPStart - objective value of the solution provided by MIP (3.30a)-(3.30d) handed over to Gurobi™via the Start attribute, only affects Table B.3

Nodes - number of nodes explored during solving process

simplex interations - number of simplex iterations performed until termination, only affects Table B.1

Status - model status, either optimal, when optimal solution is found, or interrupted, when time limit is reached

Obj Val - objective value of the optimal or incumbent solution, respectively

Gap - gap between upper bound ub and lower bound lb, $\frac{|ub-lb|}{|ub|}$ in %

Time - time elapsed when solver terminates in s, rounded up to two decimal points

Problem	**Nodes**	**simplex iterations**	**Status**	**Obj Val**	**Gap**	**Time**
10_2-1-1	1	1122	optimal	2	0	0.23
10_2-1-2	256	6949	optimal	0	0	0.33
10_2-1-3	214	3825	optimal	3	0	0.25

Problem	Nodes	simplex iterations	Status	Obj Val	Gap	Time
10_2-1-4	1	855	optimal	3	0	0.19
10_2-1-5	1	856	optimal	3	0	0.14
10_2-1-6	1	1531	optimal	6	0	0.33
10_2-1-7	1	782	optimal	3	0	0.13
10_2-1-8	1	1573	optimal	2	0	0.20
10_2-1-9	1	444	optimal	0	0	0.09
10_2-1-10	1	727	optimal	0	0	0.11
10_2-2-1	2560	108611	optimal	8	0	3.64
10_2-2-2	854	27295	optimal	3	0	0.77
10_2-2-3	278	4044	optimal	6	0	0.25
10_2-2-4	1	1799	optimal	5	0	0.67
10_2-2-5	4282	208512	optimal	2	0	4.84
10_2-2-6	1	1534	optimal	0	0	0.51
10_2-2-7	7705	490088	optimal	2	0	12.34
10_2-2-8	403	22147	optimal	6	0	0.85
10_2-2-9	1	1742	optimal	0	0	0.43
10_2-2-10	2416	121164	optimal	5	0	2.34
10_4-1-1	1	694	optimal	1	0	0.15
10_4-1-2	65	2095	optimal	6	0	0.22
10_4-1-3	1	1321	optimal	1	0	0.31
10_4-1-4	21	1491	optimal	8	0	0.37
10_4-1-5	1	590	optimal	2	0	0.12
10_4-1-6	648	21358	optimal	2	0	0.51
10_4-1-7	1	920	optimal	14	0	0.19
10_4-1-8	1	653	optimal	9	0	0.11
10_4-1-9	122	4545	optimal	2	0	0.29
10_4-1-10	1	906	optimal	6	0	0.17
10_4-2-1	118	3723	optimal	2	0	0.61
10_4-2-2	1	1066	optimal	7	0	0.32
10_4-2-3	64	2172	optimal	11	0	0.47

Problem	**Nodes**	**simplex iterations**	**Status**	**Obj Val**	**Gap**	**Time**
10_4-2-4	0	19	optimal	5	0	0.20
10_4-2-5	624	22771	optimal	4	0	1.02
10_4-2-6	0	7	optimal	2	0	0.15
10_4-2-7	146	4656	optimal	3	0	0.80
10_4-2-8	46	1663	optimal	2	0	0.55
10_4-2-9	1	684	optimal	5	0	0.19
10_4-2-10	1	685	optimal	0	0	0.10
15_2-1-1	76	4487	optimal	0	0	0.65
15_2-1-2	23	5691	optimal	3	0	0.60
15_2-1-3	360	17598	optimal	3	0	0.81
15_2-1-4	1	3000	optimal	3	0	0.44
15_2-1-5	4194	356422	optimal	4	0	1.30
15_2-1-6	2549	125810	optimal	2	0	2.51
15_2-1-7	1248	54176	optimal	4	0	2.14
15_2-1-8	3891	254441	optimal	3	0	4.59
15_2-1-9	1	2526	optimal	0	0	0.41
15_2-1-10	1	2729	optimal	6	0	0.38
15_2-2-1	2123	227601	optimal	3	0	6.09
15_2-2-2	497	12981	optimal	5	0	0.77
15_2-2-3	2810	71232	optimal	12	0	1.63
15_2-2-4	9034	973575	optimal	2	0	39.90
15_2-2-5	5126	547405	optimal	2	0	19.24
15_2-2-6	2124	76064	optimal	7	0	1.81
15_2-2-7	2272	326713	optimal	2	0	8.35
15_2-2-8	1216	93394	optimal	12	0	3.18
15_2-2-9	5759	597830	optimal	2	0	23.07
15_2-2-10	248	11513	optimal	3	0	0.65
15_4-1-1	2098	157327	optimal	11	0	3.34
15_4-1-2	251	8789	optimal	11	0	0.66
15_4-1-3	539	45501	optimal	3	0	1.34

Problem	Nodes	simplex iterations	Status	Obj Val	Gap	Time
15_4-1-4	4861	432003	optimal	8	0	14.73
15_4-1-5	1	3442	optimal	1	0	0.56
15_4-1-6	2727	149263	optimal	3	0	3.57
15_4-1-7	97	6184	optimal	20	0	0.79
15_4-1-8	87	6543	optimal	8	0	0.76
15_4-1-9	1	1516	optimal	2	0	0.38
15_4-1-10	744	58243	optimal	5	0	1.45
15_4-2-1	1211	97004	optimal	5	0	2.39
15_4-2-2	93	7410	optimal	8	0	0.63
15_4-2-3	191	7736	optimal	9	0	0.55
15_4-2-4	564	29220	optimal	7	0	1.13
15_4-2-5	1	3109	optimal	4	0	0.50
15_4-2-6	1	2526	optimal	6	0	0.33
15_4-2-7	3411	361435	optimal	4	0	7.98
15_4-2-8	1	2197	optimal	10	0	0.39
15_4-2-9	6358	662229	optimal	4	0	20.28
15_4-2-10	1303	72862	optimal	15	0	1.46
20_2-1-1	1	3514	optimal	0	0	0.81
20_2-1-2	3166	314830	optimal	6	0	37.30
20_2-1-3	1988	145974	optimal	2	0	15.88
20_2-1-4	41815	5410465	optimal	6	0	443.30
20_2-1-5	1	4582	optimal	3	0	0.75
20_2-1-6	5595	331610	optimal	5	0	10.53
20_2-1-7	7850	5609977	optimal	7	0	21.01
20_2-1-8	1	3740	optimal	3	0	0.58
20_2-1-9	12527	1462252	optimal	4	0	58.81
20_2-1-10	201	17553	optimal	0	0	1.58
20_2-2-1	113367	8799089	optimal	5	0	407.34
20_2-2-2	7860	489972	optimal	5	0	28.77
20_2-2-3	2093	221926	optimal	3	0	10.48

Problem	**Nodes**	**simplex iterations**	**Status**	**Obj Val**	**Gap**	**Time**
20_2-2-4	2241	166658	optimal	2	0	7.36
20_2-2-5	1730	122591	optimal	2	0	6.85
20_2-2-6	15402	1152807	optimal	7	0	36.97
20_2-2-7	6228	283944	optimal	3	0	14.52
20_2-2-8	1	6088	optimal	3	0	1.33
20_2-2-9	3670	215469	optimal	6	0	10.76
20_2-2-10	13411	692132	optimal	4	0	68.89
20_4-1-1	1731	95345	optimal	5	0	4.04
20_4-1-2	1	3584	optimal	5	0	0.78
20_4-1-3	3972	564466	optimal	8	0	29.85
20_4-1-4	5283	281327	optimal	9	0	11.72
20_4-1-5	1	4116	optimal	5	0	0.88
20_4-1-6	778	48707	optimal	0	0	2.52
20_4-1-7	1	3799	optimal	1	0	0.80
20_4-1-8	5888	529123	optimal	5	0	18.08
20_4-1-9	4818	579244	optimal	9	0	35.51
20_4-1-10	246	13381	optimal	7	0	1.42
20_4-2-1	4357	607944	optimal	5	0	51.5
20_4-2-2	2170	182871	optimal	4	0	24.85
20_4-2-3	185	13560	optimal	7	0	3.66
20_4-2-4	1	3403	optimal	3	0	1.50
20_4-2-5	1646	167067	optimal	18	0	12.52
20_4-2-6	3179	266848	optimal	8	0	21.07
20_4-2-7	1	12481	optimal	14	0	4.69
20_4-2-8	6360	1317354	optimal	4	0	114.63
20_4-2-9	24610	2008326	optimal	13	0	168.72
20_4-2-10	1	4311	optimal	11	0	0.90
30_2-1-1	1	12267	optimal	0	0	5.49
30_2-1-2	8	18254	optimal	0	0	17.06
30_2-1-3	22080	1158025	optimal	4	0	1001.43

Problem	Nodes	simplex iterations	Status	Obj Val	Gap	Time
30_2-1-4	4173	557693	optimal	0	0	89.25
30_2-1-5	37599	187835	optimal	1	0	362.65
30_2-1-6	706	61561	optimal	4	0	15.56
30_2-1-7	284391	29782599	optimal	4	0	2371.17
30_2-1-8	1	12977	optimal	0	0	2.86
30_2-1-9	1200	138260	optimal	2	0	9.10
30_2-1-10	1	17939	optimal	1	0	3.54
30_2-2-1	1089	101590	optimal	5	0	13.42
30_2-2-2	18568	1631150	optimal	3	0	46.96
30_2-2-3	1202	122916	optimal	11	0	21.44
30_2-2-4	337391	53585981	optimal	10	0	1223.37
30_2-2-5	93306	5937693	optimal	39	2.5641	365.44
30_2-2-6	8303737	649975006	interrupted	6	66.667	50000.05
30_2-2-7	52512	3579851	optimal	38	2.6316	25010.43
30_2-2-8	12376	1292665	optimal	3	0	48.33
30_2-2-9	2978	229267	optimal	3	0	6.89
30_2-2-10	3559	343770	optimal	16	0	29.47
30_4-1-1	382	46024	optimal	2	0	7.56
30_4-1-2	11291	1381795	optimal	5	0	85.57
30_4-1-3	4800	501522	optimal	4	0	51.78
30_4-1-4	130825	18698574	optimal	14	0	1869.82
30_4-1-5	21515	3101402	optimal	7	0	286.79
30_4-1-6	31533	2768581	optimal	2	0	5269.63
30_4-1-7	2167	425595	optimal	7	0	42.65
30_4-1-8	1	9042	optimal	6	0	2.37
30_4-1-9	1	11828	optimal	5	0	2.68
30_4-1-10	53971	6438066	optimal	9	0	49789.46
30_4-2-1	17579	3962755	optimal	14	0	278.89
30_4-2-2	71	15916	optimal	5	0	2.89
30_4-2-3	1763	228662	optimal	9	0	25.49

Problem	**Nodes**	**simplex iterations**	**Status**	**Obj Val**	**Gap**	**Time**
30_4-2-4	261322	35011933	optimal	14	0	2474.55
30_4-2-5	1485859	105316434	optimal	25	0	5776.07
30_4-2-6	1067579	123574543	optimal	13	0	33 825.67
30_4-2-7	1	8521	optimal	7	0	2.29
30_4-2-8	228	24827	optimal	24	0	5.94
30_4-2-9	2465	288071	optimal	17	0	37.09
30_4-2-10	1	12799	optimal	12	0	3.38
30_8-1-1	97750	6664211	optimal	30	0	676.78
30_8-1-2	371	45584	optimal	12	0	6.05
30_8-1-3	6705	510808	optimal	13	0	17.81
30_8-1-4	149	36022	optimal	18	0	8.42
30_8-1-5	4721	775084	optimal	27	0	29.06
30_8-1-6	11866	2625520	optimal	16	0	302.92
30_8-1-7	81273	9832639	optimal	5	0	524.91
30_8-1-8	127833	18965452	optimal	13	0	7584.43
30_8-1-9	1696	49379	optimal	25	0	10.95
30_8-1-10	241	29708	optimal	11	0	5.46
30_8-2-1	5381	762988	optimal	32	0	29.27
30_8-2-2	398	39434	optimal	12	0	5.26
30_8-2-3	3895	286450	optimal	12	0	19.71
30_8-2-4	23765	2544924	optimal	37	0	283.43
30_8-2-5	89171	11668494	optimal	16	0	861.52
30_8-2-6	80679	12184261	optimal	24	0	1008.93
30_8-2-7	34053	3775060	optimal	8	0	375.57
30_8-2-8	1	21475	optimal	29	0	5.91
30_8-2-9	3808	296503	optimal	5	0	28.72
30_8-2-10	238	39481	optimal	24	0	5.27
40_4-1-1	44244	8572568	optimal	50	0	2352.00
40_4-1-2	2179	146663	optimal	37	0	26.27
40_4-1-3	22058	3123955	optimal	27	0	1108.90

Problem	**Nodes**	**simplex iterations**	**Status**	**Obj Val**	**Gap**	**Time**
40_4-1-4	3492	440488	optimal	82	1.2195	227.83
40_4-1-5	159292	24246249	optimal	41	0	28 832.90
40_4-1-6	201943	21441442	optimal	40	0	3071.36
40_4-1-7	1089022	123947018	optimal	8	0	2026.98
40_4-1-8	731429	160012696	interrupted	6	100	50 000.05
40_4-1-9	3166	189589	optimal	19	0	21.03
40_4-1-10	15944	1189780	optimal	41	2.4390	173.13
40_4-2-1	732958	177796577	interrupted	28	100	50 000.10
40_4-2-2	621677	234448174	interrupted	60	50	50 000.03
40_4-2-3	2379734	648559892	interrupted	29	13.7931	50 000.09
40_4-2-4	893850	201285975	interrupted	37	100	50 000.15
40_4-2-5	322481	53445900	optimal	44	0	23 901.34
40_4-2-6	53801	4601221	optimal	52	0	2830.51
40_4-2-7	485664	1849555	optimal	48	2.1344	17 269.41
40_4-2-8	422	26443	optimal	54	0	829.41
40_4-2-9	6554	60056	optimal	46	2.0054	4412.67
40_4-2-10	1612	2557	optimal	37	0	2854.80
40_8-1-1	1362832	112635591	optimal	68	2.9412	35 566.02
40_8-1-2	85608	6991181	optimal	26	0	1572.34
40_8-1-3	51659	7274944	optimal	40	0	5024.46
40_8-1-4	4277	230775	optimal	44	0	27.78
40_8-1-5	6804	886934	optimal	50	0	349.14
40_8-1-6	58936	11323469	optimal	16	0	40 651.50
40_8-1-7	3952	358631	optimal	24	0	41.15
40_8-1-8	2257675	268968432	interrupted	38	100	50 000.15
40_8-1-9	1	44234	optimal	57	1.7544	50.47
40_8-1-10	12276	1192188	optimal	17	0	124.91
40_8-2-1	26001	8399463	optimal	78	0	328.51
40_8-2-2	42764	3270094	optimal	91	0	2095.63
40_8-2-3	10385	2774622	optimal	51	2.0863	1003.52

Problem	**Nodes**	**simplex iterations**	**Status**	**Obj Val**	**Gap**	**Time**
40_8-2-4	4952	388465	optimal	75	2.1984	177.05
40_8-2-5	15309	10294344	optimal	63	0	842.77
40_8-2-6	735	24409	optimal	82	0	51.09
40_8-2-7	3849	488374	optimal	109	2.2666	541.38
40_8-2-8	638422	2346664	interrupted	60	100	50 000.10
40_8-2-9	1173405	103744432	interrupted	50	100	50 000.09
40_8-2-10	9388	200934	optimal	76	0	930.41
50_4-1-1	2711	438223	optimal	27	0	280.11
50_4-1-2	791365	268400868	interrupted	30	23.33	50 000.10
50_4-1-3	24652	3679633	optimal	38	2.6316	1150.73
50_4-1-4	2657	356704	optimal	38	2.6316	102.49
50_4-1-5	240710	27567454	interrupted	42	100	50 000.13
50_4-1-6	632452	174433628	optimal	23	0	8117.42
50_4-1-7	471915	57721795	optimal	62	1.6129	4675.64
50_4-1-8	1124	112908	optimal	38	2.6316	56.21
50_4-1-9	2723	1071715	interrupted	85	85.8824	50 000.04
50_4-1-10	114062	9360474	optimal	27	0	475.97
50_4-2-1	456369	178885571	interrupted	48	100	50 000.15
50_4-2-2	1119001	227933588	interrupted	77	24.6753	50 000.20
50_4-2-3	264708	92080217	interrupted	98	100	50 000.13
50_4-2-4	962758	445765326	interrupted	44	38.6364	50 000.13
50_4-2-5	32845	7942764	optimal	64	0	921.38
50_4-2-6	890734	74364780	interrupted	80	100	50 000.11
50_4-2-7	138424	11543975	optimal	63	1.989	26 590.61
50_4-2-8	52905	1052222	optimal	62	0	1420.91
50_4-2-9	553756	982176554	optimal	78	2.3211	28 004.68
50_4-2-10	9052113	9834254211	optimal	50	0	48 012.25
50_8-1-1	108977	6132090	interrupted	54	100	50 000.15
50_8-1-2	456656	80143298	optimal	24	0	35 998.22
50_8-1-3	5744	21409	optimal	34	1.9867	1420.98

Problem	Nodes	simplex iterations	Status	Obj Val	Gap	Time
50_8-1-4	534432	53286874	optimal	44	0	10972.41
50_8-1-5	894770	7400989	optimal	19	0	16309.37
50_8-1-6	143768	5446571	optimal	15	0	5260.76
50_8-1-7	8975	879090	optimal	28	2.1332	834.05
50_8-1-8	6372	1450857	optimal	70	2.8571	356.06
50_8-1-9	565680	99794627	interrupted	45	100	50000.20
50_8-1-10	2823	1624796	optimal	32	0	781.90
50_8-2-1	87433	9371774	optimal	103	2.9126	8926.87
50_8-2-2	257364	34432876	interrupted	88	100	50000.05
50_8-2-3	64473	7854309	optimal	93	0	1461.61
50_8-2-4	831146	68401126	optimal	99	2.0202	4322.59
50_8-2-5	532713	608335179	interrupted	66	100	50000.25
50_8-2-6	99836	90647145	optimal	92	0	5198.32
50_8-2-7	192324	64887143	optimal	62	0	24954.23
50_8-2-8	654656	730909843	interrupted	84	100	50000.15
50_8-2-9	99761	655639	interrupted	72	46.9897	50000.13
50_8-2-10	376822	4281143	interrupted	88	100	50000.05

Table B.1.: MIP results

B.1.1. Statistic values for Gurobi Optimizer without start solution

The columns of Table B.2 and B.4 are defined as follows

Instance - problem instances in the usual notation

μ(Nodes) - mean value of number of explored nodes over all sequences

σ(Nodes) - standard deviation of number of explored nodes over all 100 problems of a instance

μ(Obj Val) - mean of objective values, only affects Table B.2

μ(Time) - mean value of computational time over all sequences

σ(Nodes) - standard deviation of computational time over all sequences

Δ - ratio of problems that are solved until optimality

The values μ(Nodes) and σ(Nodes) are rounded up to integers, μ(Obj Val) is rounded up to one decimal point, μ(Time) and σ(Time) are rounded up to four decimal points.

Instance	**μ(Nodes)**	**σ(Nodes)**	**μ(Obj Val)**	**μ(Time)**	**σ(Time)**	**Δ**
10_2-1	178	384	2.4	0.2882	0.2736	
10_2-2	1139	1873	2.8	1.6326	2.5258	
10_4-1	33	81	4.7	0.2164	0.0642	
10_4-2	50	104	6.2	0.3690	0.1467	
15_2-1	986	2027	2.6	2.0670	4.6445	
15_2-2	4474	4202	4.7	8.5786	8.2114	
15_4-1	469	865	6.4	1.0756	1.5786	
15_4-2	684	1228	8.1	1.5507	2.8000	
20_2-1	1996	4537	2.9	10.4456	44.3706	
20_2-2	11936	19159	4.2	41.9028	73.0439	

Instance	μ(Nodes)	σ(Nodes)	μ(Obj Val)	μ(Time)	σ(Time)	Δ
20_4-1	1602	1842	7.4	7.4917	11.0009	
20_4-2	3240	6467	9.9	16.6039	26.7905	
30_2-1	70823	605066	2.4	969.3047	8221.0383	
30_2-2	272192	1296445	6	1859.05	8599.9129	0.98
30_4-1	15884	52755	7.6	679.9996	4979.1035	
30_4-2	50767	185711	13.8	833.9253	4165.6043	
30_8-1	11577	31687	17.2	260.5132	1029.9150	
30_8-2	11784	24811	22.7	469.8185	3417.5650	
40_4-1	189789	373012	34.7	6927.1679	14698.3453	0.92
40_4-2	327485	579326	45.2	9636.2526	15734.4021	0.88
40_8-1	125661	317561	39.8	5180.71	9531.1251	0.93
40_8-2	223094	528831	48.2	7052.8564	10435.5419	0.9
50_4-1	475933	1539808	41	24385.2291	48688.4240	0.75
50_4-2	558974	22436875	52.7	37125.1911	49273.9833	0.71
50_8-1	354527	963444	42.5	22764.0935	45928.6232	0.77
50_8-2	468853	1365390	78.7	29803.746	48873.2565	0.72

Table B.2.: Statistic values for MIP results

B.2. Gurobi Optimizer with MIPstart

Problem	MIPstart	Nodes	Status	Obj Val	Gap	Time
10_2-1-1	2	1	optimal	2	0	0.40
10_2-1-2	4	610	optimal	0	0	1.06
10_2-1-3	5	1	optimal	3	0	1.17
10_2-1-4	3	1	optimal	3	0	0.27
10_2-1-5	4	1	optimal	3	0	0.22
10_2-1-6	14	1	optimal	6	0	0.70
10_2-1-7	3	15	optimal	3	0	0.31
10_2-1-8	5	556	optimal	2	0	0.68
10_2-1-9	0	0	optimal	0	0	0.09
10_2-1-10	0	0	optimal	0	0	0.07
10_2-2-1	10	2334	optimal	8	0	3.09
10_2-2-2	6	865	optimal	3	0	1.04
10_2-2-3	6	171	optimal	6	0	0.32
10_2-2-4	12	1	optimal	5	0	0.71
10_2-2-5	2	3894	optimal	2	0	7.19
10_2-2-6	4	1	optimal	0	0	0.34
10_2-2-7	2	4413	optimal	2	0	8.07
10_2-2-8	8	914	optimal	6	0	1.05
10_2-2-9	2	1	optimal	0	0	0.47
10_2-2-10	5	2369	optimal	5	0	2.84
10_4-1-1	1	1	optimal	1	0	0.30
10_4-1-2	6	1	optimal	6	0	0.53
10_4-1-3	1	1	optimal	1	0	0.37
10_4-1-4	8	5	optimal	8	0	0.55
10_4-1-5	2	1	optimal	2	0	0.21
10_4-1-6	2	48	optimal	2	0	0.49
10_4-1-7	14	93	optimal	14	0	0.41
10_4-1-8	13	1	optimal	9	0	0.18
10_4-1-9	2	193	optimal	2	0	0.63

Problem	**MIPstart**	**Nodes**	**Status**	**Obj Val**	**Gap**	**Time**
10_4-1-10	6	1	optimal	6	0	0.18
10_4-2-1	2	160	optimal	2	0	0.68
10_4-2-2	7	1	optimal	7	0	0.52
10_4-2-3	11	1	optimal	11	0	1.21
10_4-2-4	14	0	optimal	5	0	0.19
10_4-2-5	5	965	optimal	4	0	1.02
10_4-2-6	4	0	optimal	2	0	0.22
10_4-2-7	3	166	optimal	3	0	0.46
10_4-2-8	2	15	optimal	2	0	0.27
10_4-2-9	5	1	optimal	5	0	0.34
10_4-2-10	0	0	optimal	0	0	0.18
15_2-1-1	1	1	optimal	0	0	0.64
15_2-1-2	6	3069	optimal	3	0	5.67
15_2-1-3	4	1	optimal	3	0	1.55
15_2-1-4	4	1	optimal	3	0	0.61
15_2-1-5	6	6704	optimal	4	0	21.27
15_2-1-6	4	1565	optimal	2	0	2.78
15_2-1-7	14	2173	optimal	4	0	4.14
15_2-1-8	3	2763	optimal	3	0	4.07
15_2-1-9	3	1	optimal	0	0	0.66
15_2-1-10	6	1	optimal	6	0	0.69
15_2-2-1	7	4517	optimal	3	0	23.30
15_2-2-2	8	1	optimal	5	0	1.34
15_2-2-3	19	3094	optimal	12	0	2.38
15_2-2-4	2	18154	optimal	2	0	31.00
15_2-2-5	3	5003	optimal	2	0	22.44
15_2-2-6	8	2412	optimal	7	0	4.96
15_2-2-7	11	2593	optimal	2	0	7.32
15_2-2-8	21	2145	optimal	12	0	13.37
15_2-2-9	8	4576	optimal	2	0	46.84

Problem	**MIPstart**	**Nodes**	**Status**	**Obj Val**	**Gap**	**Time**
15_2-2-10	4	674	optimal	3	0	1.73
15_4-1-1	11	1580	optimal	11	0	2.99
15_4-1-2	11	1	optimal	11	0	0.86
15_4-1-3	3	398	optimal	3	0	1.29
15_4-1-4	9	2436	optimal	8	0	10.63
15_4-1-5	1	1	optimal	1	0	0.50
15_4-1-6	3	912	optimal	3	0	1.93
15_4-1-7	20	705	optimal	20	0	1.68
15_4-1-8	8	898	optimal	8	0	1.62
15_4-1-9	2	1	optimal	2	0	0.41
15_4-1-10	5	541	optimal	5	0	1.74
15_4-2-1	6	1412	optimal	5	0	3.27
15_4-2-2	8	62	optimal	8	0	1.19
15_4-2-3	9	1	optimal	9	0	2.27
15_4-2-4	8	729	optimal	7	0	1.7
15_4-2-5	6	1	optimal	4	0	0.93
15_4-2-6	6	1	optimal	6	0	0.56
15_4-2-7	8	3618	optimal	4	0	10.45
15_4-2-8	10	1	optimal	10	0	0.66
15_4-2-9	4	5217	optimal	4	0	20.71
15_4-2-10	16	1131	optimal	15	0	1.83
20_2-1-1	2	1	optimal	0	0	0.96
20_2-1-2	9	4070	optimal	6	0	15.12
20_2-1-3	4	46718	optimal	2	0	595.56
20_2-1-4	15	389572	optimal	6	0	2499.01
20_2-1-5	5	1	optimal	3	0	0.69
20_2-1-6	5	5253	optimal	5	0	7.53
20_2-1-7	9	1515	optimal	7	0	26.74
20_2-1-8	3	1	optimal	3	0	0.49
20_2-1-9	4	21649	optimal	4	0	131.28

Problem	**MIPstart**	**Nodes**	**Status**	**Obj Val**	**Gap**	**Time**
20_2-1-10	1	1	optimal	0	0	0.90
20_2-2-1	19	238659	optimal	5	0	898.73
20_2-2-2	12	23866	optimal	5	0	53.48
20_2-2-3	6	2128	optimal	3	0	8.90
20_2-2-4	6	4516	optimal	2	0	13.21
20_2-2-5	6	1226	optimal	2	0	5.08
20_2-2-6	21	32600	optimal	7	0	68.75
20_2-2-7	6	5161	optimal	3	0	8.39
20_2-2-8	4	1	optimal	3	0	1.92
20_2-2-9	11	5428	optimal	6	0	16.08
20_2-2-10	13	6194	optimal	4	0	21.80
20_4-1-1	11	1850	optimal	5	0	3.26
20_4-1-2	6	1	optimal	5	0	0.98
20_4-1-3	8	1478	optimal	8	0	7.79
20_4-1-4	15	6857	optimal	9	0	12.19
20_4-1-5	6	1	optimal	5	0	0.77
20_4-1-6	2	803	optimal	0	0	3.23
20_4-1-7	4	1	optimal	1	0	0.67
20_4-1-8	7	3718	optimal	5	0	8.82
20_4-1-9	13	13003	optimal	9	0	54.69
20_4-1-10	7	150	optimal	7	0	1.42
20_4-2-1	8	4272	optimal	5	0	43.58
20_4-2-2	7	3009	optimal	4	0	10.43
20_4-2-3	7	1569	optimal	7	0	13.12
20_4-2-4	3	1	optimal	3	0	1.63
20_4-2-5	18	2253	optimal	18	0	13.91
20_4-2-6	9	2159	optimal	8	0	23.94
20_4-2-7	14	60	optimal	14	0	1.79
20_4-2-8	10	2078	optimal	4	0	38.37
20_4-2-9	18	24015	optimal	13	0	136.95

Problem	**MIPstart**	**Nodes**	**Status**	**Obj Val**	**Gap**	**Time**
20_4-2-10	12	1	optimal	11	0	3.17
30_2-1-1	4	1	optimal	0	0	2.54
30_2-1-2	2	209	optimal	0	0	6.99
30_2-1-3	4	21127	optimal	4	0	334.70
30_2-1-4	5	10234	optimal	0	0	35.82
30_2-1-5	9	9532	optimal	1	0	105.93
30_2-1-6	6	1126	optimal	4	0	6.81
30_2-1-7	7	60578	optimal	4	0	630.86
30_2-1-8	2	296	optimal	0	0	13.48
30_2-1-9	2	2126	optimal	2	0	28.42
30_2-1-10	3	1	optimal	1	0	4.33
30_2-2-1	17	1416	optimal	5	0	16.63
30_2-2-2	12	7993	optimal	3	0	45.22
30_2-2-3	23	10596	optimal	11	0	92.15
30_2-2-4	16	48283	optimal	10	0	179.67
30_2-2-5	43	47710	optimal	39	0	19117.78
30_2-2-6	47	1479638	optimal	6	0	19642.34
30_2-2-7	42	60287	optimal	38	2.6316	384.52
30_2-2-8	12	6781	optimal	3	0	85.36
30_2-2-9	9	3642	optimal	3	0	7.63
30_2-2-10	28	4297	optimal	16	0	25.18
30_4-1-1	9	2032	optimal	2	0	24.78
30_4-1-2	8	3374	optimal	5	0	22.47
30_4-1-3	9	6456	optimal	4	0	122.70
30_4-1-4	17	66082	optimal	14	0	712.38
30_4-1-5	13	14058	optimal	7	0	50.61
30_4-1-6	5	34046	optimal	2	0	757.34
30_4-1-7	10	8729	optimal	7	0	28.74
30_4-1-8	10	1	optimal	6	0	2.19
30_4-1-9	5	1	optimal	5	0	3.42

Problem	**MIPstart**	**Nodes**	**Status**	**Obj Val**	**Gap**	**Time**
30_4-1-10	13	22129	optimal	9	0	347.88
30_4-2-1	19	75819	optimal	14	0	245.95
30_4-2-2	10	48	optimal	5	0	2.91
30_4-2-3	9	6296	optimal	9	0	60.73
30_4-2-4	18	27729	optimal	14	0	288.20
30_4-2-5	44	822675	optimal	25	0	2033.25
30_4-2-6	22	601215	optimal	13	0	4638.76
30_4-2-7	7	1	optimal	7	0	2.13
30_4-2-8	24	3152	optimal	24	0	9.36
30_4-2-9	23	7649	optimal	17	0	54.78
30_4-2-10	12	1	optimal	12	0	2.28
30_8-1-1	34	6317	optimal	30	0	192.32
30_8-1-2	13	760	optimal	12	0	5.30
30_8-1-3	15	6196	optimal	13	0	20.51
30_8-1-4	19	1838	optimal	18	0	33.28
30_8-1-5	39	2233	optimal	27	0	49.10
30_8-1-6	18	129505	optimal	16	0	651.56
30_8-1-7	9	18504	optimal	5	0	88.98
30_8-1-8	20	934552	optimal	13	0	8279.58
30_8-1-9	55	313	optimal	25	0	4.45
30_8-1-10	11	1	optimal	11	0	5.92
30_8-2-1	42	3167	optimal	32	0	66.38
30_8-2-2	12	474	optimal	12	0	4.68
30_8-2-3	12	21492	optimal	12	0	179.61
30_8-2-4	46	59069	optimal	37	0	456.62
30_8-2-5	25	170815	optimal	16	0	1526.36
30_8-2-6	37	308862	optimal	24	0	2606.09
30_8-2-7	18	26854	optimal	8	0	311.45
30_8-2-8	33	1	optimal	29	0	5.19
30_8-2-9	9	2213	optimal	5	0	41.07

Problem	**MIPstart**	**Nodes**	**Status**	**Obj Val**	**Gap**	**Time**
30_8-2-10	24	274	optimal	24	0	8.06
40_4-1-1	59	27724	optimal	50	0	1024.85
40_4-1-2	43	357	optimal	37	0	32.24
40_4-1-3	43	4035	optimal	27	0	217.74
40_4-1-4	93	5545	optimal	82	2.439	60.22
40_4-1-5	94	15592	optimal	41	2.439	436.79
40_4-1-6	53	85334	optimal	40	2.5	1655.34
40_4-1-7	20	49904	optimal	8	0	355.38
40_4-1-8	34	798552	interrupted	6	100	50000.15
40_4-1-9	27	4332	optimal	19	0	25.63
40_4-1-10	58	18851	optimal	41	2.439	372.12
40_4-2-1	64	887232	interrupted	30	100	50000.10
40_4-2-2	58	998126	interrupted	52	27.9817	50000.17
40_4-2-3	72	655428	optimal	25	0	41621.27
40_4-2-4	66	665872	interrupted	37	100	50000.12
40_4-2-5	78	99872	optimal	44	0	17253.67
40_4-2-6	94	76823	optimal	52	2.1213	5411.98
40_4-2-7	88	78655	optimal	48	0	11219.45
40_4-2-8	82	6528	optimal	54	2.4981	622.52
40_4-2-9	57	7878	optimal	46	2.0018	987.63
40_4-2-10	62	12326	optimal	37	0	1128.55
40_8-1-1	88	658129	optimal	68	2.1342	24562.87
40_8-1-2	28	2994	optimal	26	0	370.07
40_8-1-3	50	18692	optimal	40	2.5	793.58
40_8-1-4	81	2200	optimal	44	0	31.81
40_8-1-5	53	2442	optimal	50	0	33.77
40_8-1-6	48	187763	optimal	16	2.9816	22871.97
40_8-1-7	30	2626	optimal	24	0	52.44
40_8-1-8	102	635810	interrupted	35	99.3	50000.12
40_8-1-9	64	1	optimal	57	0	19.94

Problem	MIPstart	Nodes	Status	Obj Val	Gap	Time
40_8-1-10	53	654896	optimal	17	0	10484.13
40_8-2-1	86	21554	optimal	78	0	316.47
40_8-2-2	110	29793	optimal	91	2.1978	1167.23
40_8-2-3	62	5818	optimal	51	0	258.95
40_8-2-4	101	57219	optimal	75	2.6667	226.54
40_8-2-5	65	129	optimal	63	0	14.88
40_8-2-6	91	48724	optimal	82	2.439	351.12
40_8-2-7	115	2121	optimal	109	0	32.98
40_8-2-8	78	523311	interrupted	63	100	50000.14
40_8-2-9	56	89624	interrupted	49	100	50000.11
40_8-2-10	83	6741	optimal	76	2.2651	587.17
50_4-1-1	38	19342	optimal	27	0	791.94
50_4-1-2	53	2034311	interrupted	37	94.5946	50000.25
50_4-1-3	71	13862	optimal	38	2.6316	236.24
50_4-1-4	96	14289	optimal	38	2.6316	694.74
50_4-1-5	53	31661	interrupted	44	100	50000.15
50_4-1-6	48	21749	optimal	23	0	12850.84
50_4-1-7	82	13225	optimal	62	2.0132	6419.05
50_4-1-8	45	1195	optimal	38	2.7027	70.61
50_4-1-9	88	264322	optimal	68	0	36740.73
50_4-1-10	42	57831	optimal	27	0	262.25
50_4-2-1	60	324551	interrupted	46	100	50000.05
50_4-2-2	117	1222985	interrupted	68	6.7663	50000.13
50_4-2-3	114	589012	interrupted	78	27.9437	50000.10
50_4-2-4	83	829061	interrupted	56	64.5521	50000.10
50_4-2-5	118	98001	optimal	64	0	1287.54
50_4-2-6	92	916227	interrupted	76	100	50000.25
50_4-2-7	101	100823	optimal	63	2.0421	18432.65
50_4-2-8	77	62341	optimal	62	0	982.67
50_4-2-9	108	476219	optimal	78	0	26443.81

Problem	MIPstart	Nodes	Status	Obj Val	Gap	Time
50_4-2-10	84	1098232	optimal	50	2.3227	35720.16
50_8-1-1	56	56743	interrupted	52	100	50000.20
50_8-1-2	32	481008	optimal	24	0	30524.59
50_8-1-3	48	41085	optimal	34	2.9412	769.30
50_8-1-4	133	91455	optimal	44	2.2727	8192.32
50_8-1-5	43	50365	optimal	19	0	19953.34
50_8-1-6	41	192693	optimal	15	0	1148.41
50_8-1-7	29	1769	optimal	28	0	111.19
50_8-1-8	148	94542	optimal	70	2.8571	2293.06
50_8-1-9	66	698873	interrupted	36	14.9723	50000.05
50_8-1-10	32	1724	optimal	32	0	196.08
50_8-2-1	111	43344	optimal	103	2.9126	1453.72
50_8-2-2	106	210096	interrupted	84	100	50000.30
50_8-2-3	115	19212	optimal	93	2.0753	467.00
50_8-2-4	167	429069	optimal	99	2.0202	4572.29
50_8-2-5	61	596122	interrupted	54	100	50000.10
50_8-2-6	98	92761	optimal	92	0	4299.76
50_8-2-7	72	223891	optimal	62	0	20981.45
50_8-2-8	102	369982	interrupted	98	100	50000.20
50_8-2-9	66	326155	interrupted	58	38.8671	50000.10
50_8-2-10	82	482130	interrupted	76	78.1134	50000.10

Table B.3.: MIP results with MIPStart

B.2.1. Statistic values for Gurobi Optimizer with start solution

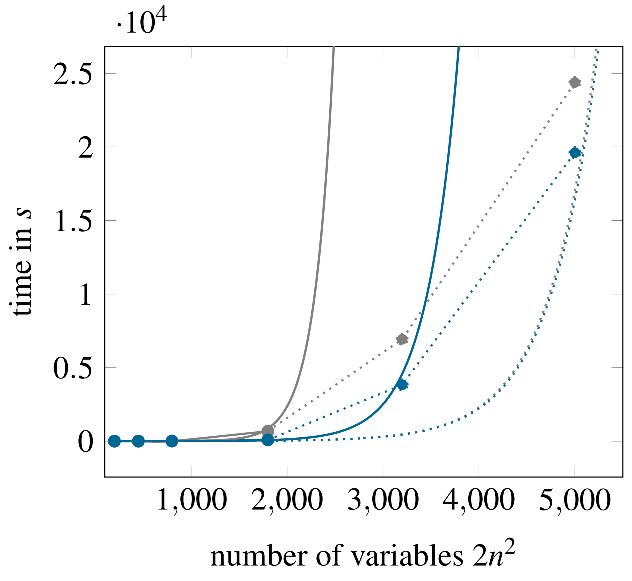
Instance	μ(Nodes)	σ(Nodes)	μ(Time)	σ(Time)	Δ
10_2-1	163	347	0.5218	0.4996	
10_2-2	1755	7099	1.6258	2.4818	
10_4-1	49	258	0.3464	0.3567	
10_4-2	55	136	0.4092	0.1584	
15_2-1	845	1588	2.1947	3.3072	
15_2-2	3507	3920	15.5188	17.9783	
15_4-1	506	894	1.5322	2.1026	
15_4-2	874	1212	2.5746	3.5803	
20_2-1	6415	38954	39.2584	254.5784	
20_2-2	15052	34960	48.1776	118.3943	
20_4-1	2018	2314	8.4606	10.7791	
20_4-2	3171	4230	22.4838	25.9126	
30_2-1	57259	292002	822.1245	4703.9788	
30_2-2	189370	820964	1504.9326	5756.8158	0.98
30_4-1	9387	23984	85.0376	183.5775	
30_4-2	29709	103787	518.9885	3520.0314	
30_8-1	17229	93644	178.6525	835.1429	
30_8-2	13654	38478	225.2411	688.0066	
40_4-1	146465	407446	3830.0334	7544.9800	0.93
40_4-2	287654	809513	7971.2516	14644.9071	0.9
40_8-1	95546	297644	3577.8907	7501.5445	0.93
40_8-2	175390	438997	5981.1437	10436.6087	0.89
50_4-1	437066	1365996	19611.2629	38745.9603	0.76
50_4-2	516990	1745988	32241.9744	48995.5307	0.71
50_8-1	275953	805543	18755.8004	37644.9610	0.75

Instance	μ**(Nodes)**	σ**(Nodes)**	μ**(Time)**	σ**(Time)**	Δ
50_8-2	397501	1153543	25769.2651	42751.1754	0.72

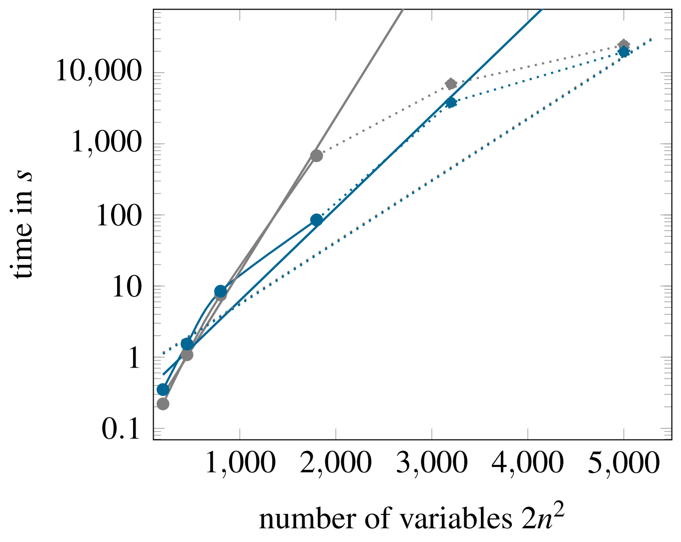
Table B.4.: Statistic values for MIP results with MIPStart

B.2.2. Overview computational time for Gurobi Optimizer

(a) linear axis



(b) logarithmic axis



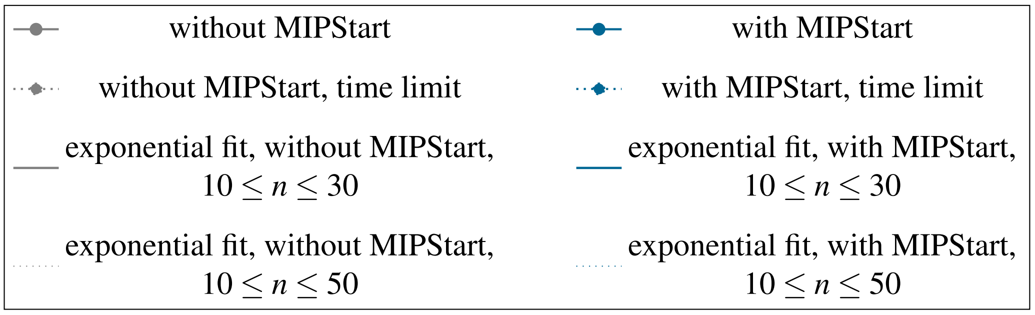

without MIPStart
with MIPStart
without MIPStart, time limit
with MIPStart, time limit
exponential fit, without MIPStart, $10 \leq n \leq 30$
exponential fit, with MIPStart, $10 \leq n \leq 30$
exponential fit, without MIPStart, $10 \leq n \leq 50$
exponential fit, with MIPStart, $10 \leq n \leq 50$

Figure B.1.: Average computational time for mono-distributed sets n_4-1

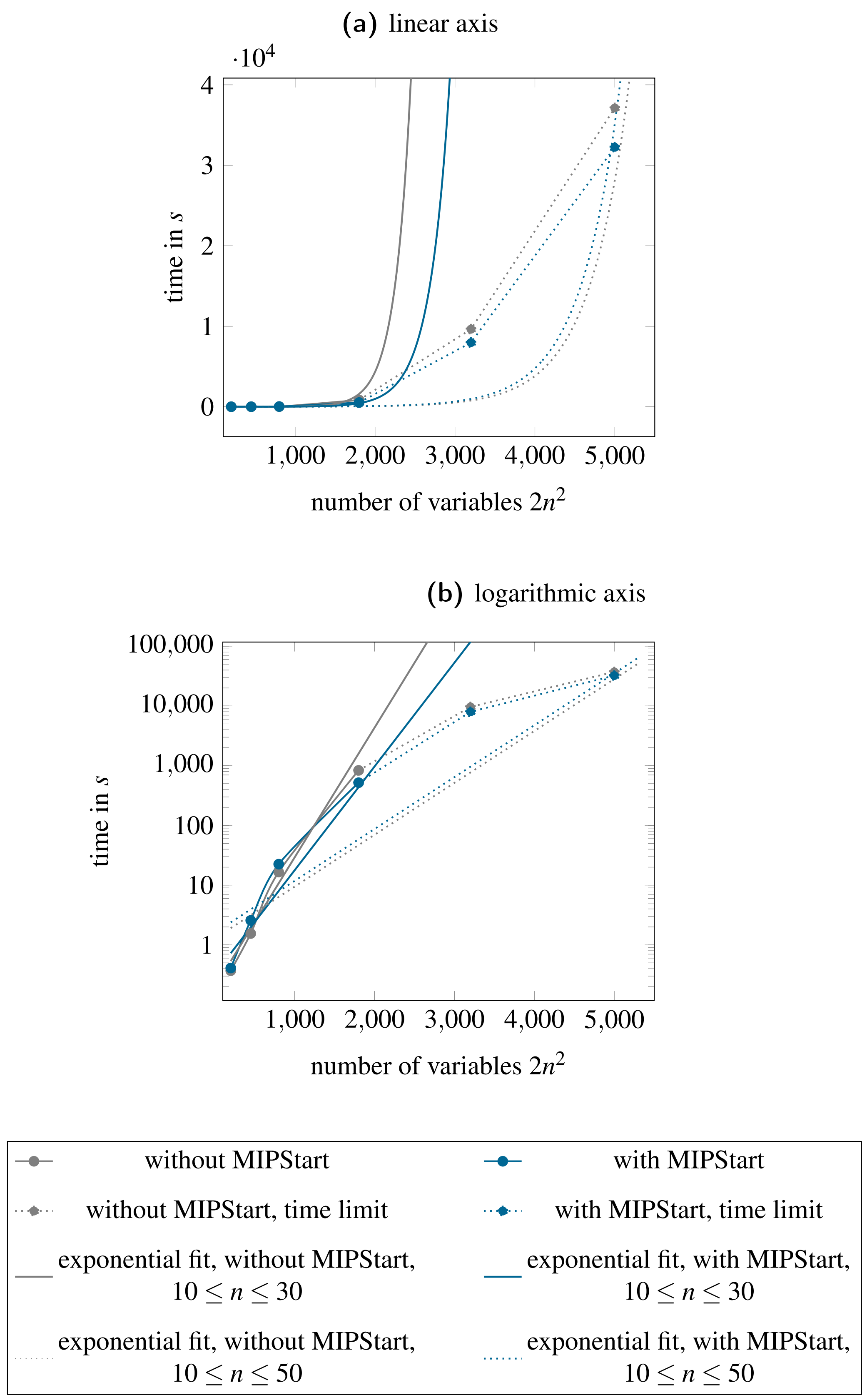


Figure B.2.: Average computational time for poly-distributed sets n_4-2

B.3. Gurobi logs of problem 30_2-2-96 with and without MIPStart

```
Gurobi 8.0.1 (win64, .NET) logging started 01/20/21 16:19:53

Optimize a model with 2217 rows, 2376 columns and 110210 nonzeros
Variable types: 148 continuous, 2228 integer (2228 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 9e+02]
  RHS range        [8e-01, 2e+02]
Presolve removed 175 rows and 6 columns
Presolve time: 0.21s
Presolved: 2042 rows, 2370 columns, 107945 nonzeros
Variable types: 115 continuous, 2255 integer (2225 binary)

Root relaxation: objective 0.000000e+00, 4067 iterations, 0.66 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0    0.00000    0  272          -    0.00000      -     -
1s
     0     0    0.00000    0  351          -    0.00000      -     -
2s
     0     0    0.00000    0  371          -    0.00000      -     -
2s
     0     0    0.00000    0  195          -    0.00000      -     -
3s
     0     0    0.00000    0  203          -    0.00000      -     -
3s
     0     0    0.00000    0  144          -    0.00000      -     -
4s
     0     0    0.00000    0  193          -    0.00000      -     -
4s
     0     0    0.00000    0  160          -    0.00000      -     -
5s
     0     0    0.00000    0  131          -    0.00000      -     -
5s
     0     2    0.00000    0  131          -    0.00000      -     -
7s
*  635   431             162      92.0000000    0.00000   100%  59.5
9s
H  676   461                      23.0000000    0.00000   100%  59.4
9s
H  913   557                      21.0000000    0.00000   100%  49.8
9s
  1173   720    0.00000   78  131   21.00000    0.00000   100%  45.2
13s
  1175   721    0.00000   42 1001   21.00000    0.00000   100%  45.1
15s
  1182   727    7.26629   86  849   21.00000    0.00000   100%  58.7
21s
  1185   729    0.48726   58 1013   21.00000    0.00000   100%  58.6
25s
H 1186   693                       7.0000000    0.00000   100%  58.5
26s
H 1187   659                       6.0000000    0.00000   100%  58.5
26s
```

```
H 1188   626                          5.0000000    0.00000   100%  58.4
28s
  1207   632    0.00000   27  883    5.00000    0.00000   100%  77.9
30s
  3388   527    0.00000   57  788    5.00000    0.00000   100%  43.6
36s
H 3397   501                          4.0000000    0.00000   100%  43.7
36s
  3525   501    0.00000   65  104    4.00000    0.00000   100%  44.3
42s
  4642   687 infeasible   57           4.00000    0.00000   100%  38.3
45s
  8784  1444 infeasible   98           4.00000    0.00000   100%  31.7
52s
  9596  1546    0.00000   73   73    4.00000    0.00000   100%  31.3
55s
 10541  1722    1.00000   83  131    4.00000    0.00000   100%  30.6
63s
 10543  1723    0.00000   62  986    4.00000    0.00000   100%  30.6
65s
 10549  1727    3.00000   99  979    4.00000    0.00000   100%  30.5
70s
 10555  1731    0.00000   64  915    4.00000    0.00000   100%  30.5
75s
 10566  1740    1.00000   70  814    4.00000    0.00000   100%  33.1
80s
 10578  1749    1.00000   82  789    4.00000    0.00000   100%  35.2
85s
 10594  1758    0.00000   57  626    4.00000    0.00000   100%  36.6
90s
H11337  1609                          3.0000000    0.00000   100%  39.0
94s
 11580  1567 infeasible   99           3.00000    0.00000   100%  39.1
95s
 11960  1500    1.00000   76   58    3.00000    0.00000   100%  40.6
100s
 13394  1423 infeasible   78           3.00000    0.00000   100%  44.8
105s
 13521  1401    1.00000   76  661    3.00000    0.00000   100%  45.8
114s
 13731  1340     cutoff   80           3.00000    0.00000   100%  47.0
115s
 14646  1205    1.00000   72  661    3.00000    0.00000   100%  51.5
120s
 17449  1602    0.06667   75  532    3.00000    0.00000   100%  55.3
125s
 19765  1794    1.00000   73  690    3.00000    0.00000   100%  59.1
130s
 24723  2343    1.00000   86  228    3.00000    1.00000  66.7%  58.1
138s
 25315  2282    1.15208   91  333    3.00000    1.00000  66.7%  58.3
181s
 28036  2453    1.00000   78   38    3.00000    1.00000  66.7%  60.5
185s
 29901  2196    1.00000   81   53    3.00000    1.00000  66.7%  63.2
190s
 32356  2406    1.00000   75  482    3.00000    1.00000  66.7%  64.6
195s
 36282  3036 infeasible   89           3.00000    1.00000  66.7%  65.0
201s
```

```
 37854  2855    1.00000   77  496    3.00000    1.00000  66.7%  66.8
206s
 40358  3293    1.17201   84  131    3.00000    1.00000  66.7%  67.9
211s
 40367  3299    2.00000   89  761    3.00000    1.00000  66.7%  67.9
215s
 40389  3304    1.00000   68  373    3.00000    1.00000  66.7%  68.3
220s
 40573  3325    1.00000   77  685    3.00000    1.00000  66.7%  68.4
226s
 41166  3274    1.00000   86  555    3.00000    1.00000  66.7%  68.8
230s
 43889  3074    1.00000   90  357    3.00000    1.00000  66.7%  70.4
235s
 46890  2606    1.13810   84  561    3.00000    1.00000  66.7%  71.8
243s
 47142  2535 infeasible   82         3.00000    1.00000  66.7%  72.1
245s
 50171  2222 infeasible   90         3.00000    1.00000  66.7%  72.4
250s
 52489  2061    1.01235   74  789    3.00000    1.00000  66.7%  73.7
255s
 55766  2268    1.00000   98   47    3.00000    1.00000  66.7%  73.1
260s
 59717  2429 infeasible   97         3.00000    1.00000  66.7%  72.2
266s
 62179  2422 infeasible  115         3.00000    1.00000  66.7%  71.9
270s
 66330  2679 infeasible  101         3.00000    1.00000  66.7%  71.2
275s
 70542  3093 infeasible   89         3.00000    1.00000  66.7%  70.5
281s
 73490  3291 infeasible  122         3.00000    1.00000  66.7%  70.5
285s
 76121  3180    1.00000  102  613    3.00000    1.00000  66.7%  71.5
290s
 79113  3281 infeasible  102         3.00000    1.00000  66.7%  71.9
295s
 82773  3623    1.00000  100  215    3.00000    1.00000  66.7%  72.1
301s
 85948  3811 infeasible  108         3.00000    1.00000  66.7%  72.1
309s
 86113  3683 infeasible  101         3.00000    1.00000  66.7%  72.2
310s
 87171  3596    1.00000  104  441    3.00000    1.00000  66.7%  73.0
315s
 90249  3914 infeasible  109         3.00000    1.00000  66.7%  73.7
321s
 92127  3972    1.00000   98  453    3.00000    1.00000  66.7%  74.4
325s
 95106  4050 infeasible   85         3.00000    1.00000  66.7%  75.4
330s
 96795  4052 infeasible   91         3.00000    1.00000  66.7%  76.2
335s
 99105  4186 infeasible  118         3.00000    1.00000  66.7%  76.6
340s
 101306  4221 infeasible  112         3.00000    1.00000  66.7%  77.0
345s
 103318  4155 infeasible   98         3.00000    1.00000  66.7%  77.9
357s
```

```
 103635  3943    2.00000   87  533    3.00000    1.00000  66.7%  78.2
361s
 104262  3836 infeasible   95         3.00000    1.00000  66.7%  78.8
365s
 106160  3816    1.03846   95   91    3.00000    1.00000  66.7%  79.5
371s
 108297  3833 infeasible  106         3.00000    1.00000  66.7%  80.0
376s
 110244  3777 infeasible   98         3.00000    1.00000  66.7%  80.3
380s
 112186  3746    1.00000   89  403    3.00000    1.00000  66.7%  81.1
386s
 114152  3633 infeasible  100         3.00000    1.00000  66.7%  81.7
390s
 116318  2345 infeasible   90         3.00000    2.00000  33.3%  81.3
395s
 118349  1506 infeasible  112         3.00000    2.00000  33.3%  80.8
400s
 121150   958 infeasible   92         3.00000    2.00000  33.3%  80.1
405s
 122882   586    2.00000   89  474    3.00000    2.00000  33.3%  80.1
410s
 124799   254    2.00000   97  656    3.00000    2.00000  33.3%  80.0
415s
 127097    26 infeasible   92         3.00000    2.00000  33.3%  80.2
420s

Cutting planes:
  Learned: 1
  Gomory: 1
  Cover: 19
  Implied bound: 18
  Clique: 33
  MIR: 13
  StrongCG: 3
  Flow cover: 55
  GUB cover: 31
  Inf proof: 82
  Zero half: 3

Explored 127395 nodes (10252692 simplex iterations) in 420.91 seconds
Thread count was 8 (of 8 available processors)

Solution count 8: 3 4 5 ... 92

Optimal solution found (tolerance 3.00e-02)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap
0.0000%
```

Figure B.3.: Gurobi log of problem 30_2-2-96 without MIPStart

```
Gurobi 8.0.1 (win64, .NET) logging started 01/25/21 21:29:51

Optimize a model with 2217 rows, 2376 columns and 110210 nonzeros
Variable types: 148 continuous, 2228 integer (2228 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 9e+02]
  RHS range        [8e-01, 2e+02]

MIP start produced solution with objective 87 (0.02s)
MIP start produced solution with objective 40 (0.02s)
MIP start produced solution with objective 14 (0.02s)
MIP start produced solution with objective 10 (0.02s)
MIP start produced solution with objective 9 (0.02s)
Loaded MIP start with objective 9

Presolve removed 175 rows and 6 columns
Presolve time: 0.20s
Presolved: 2042 rows, 2370 columns, 107945 nonzeros
Variable types: 115 continuous, 2255 integer (2225 binary)

Root relaxation: objective 0.000000e+00, 4067 iterations, 0.60 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0    0.00000    0  272    9.00000    0.00000   100%     -
1s
     0     0    0.00000    0  357    9.00000    0.00000   100%     -
2s
     0     0    0.00000    0  347    9.00000    0.00000   100%     -
2s
     0     0    0.00000    0  176    9.00000    0.00000   100%     -
3s
     0     0    0.00000    0  211    9.00000    0.00000   100%     -
3s
H    0     0                    8.0000000    0.00000   100%     -
4s
     0     0    0.00000    0  180    8.00000    0.00000   100%     -
4s
     0     0    0.00000    0  183    8.00000    0.00000   100%     -
4s
     0     0    0.00000    0  169    8.00000    0.00000   100%     -
4s
     0     0    0.00000    0  167    8.00000    0.00000   100%     -
5s
     0     2    0.00000    0  158    8.00000    0.00000   100%     -
5s
   873   532    0.00000   36  202    8.00000    0.00000   100%  96.6
10s
  1019   646    0.00000   42  188    8.00000    0.00000   100%   102
17s
  1084   698    0.00000   23  167    8.00000    0.00000   100%   106
21s
  1088   701    3.13438   25 1095    8.00000    0.00000   100%   106
25s
```

```
  1095   706    1.87983   27  937    8.00000    0.00000   100%   122
31s
  1100   711    0.75645   35  167    8.00000    0.00000   100%   133
35s
  1105   714    1.34284   69  847    8.00000    0.00000   100%   133
40s
  1113   721    0.03030   81  965    8.00000    0.00000   100%   153
46s
  1118   726    4.50280   37  167    8.00000    0.00000   100%   164
50s
  1123   729    0.65029   50  996    8.00000    0.00000   100%   163
55s
  1129   735    3.31067   85 1018    8.00000    0.00000   100%   174
60s
  1135   739    0.03030   43  953    8.00000    0.00000   100%   174
65s
  1621   764 infeasible  132         8.00000    0.00000   100%  79.5
70s
  4918  1047    6.00000  138  670    8.00000    0.00000   100%  43.9
76s
H 5003   866                       4.0000000    0.00000   100%  43.5
76s
  6469  1124    0.07143  149  639    4.00000    0.00000   100%  42.9
80s
  9262  1638     cutoff  151         4.00000    0.00000   100%  40.1
86s
  9524  1659    1.12026  117  655    4.00000    0.00000   100%  40.2
90s
  9716  1680    1.37119  142  634    4.00000    0.00000   100%  40.6
95s
 11149  1988    1.00000  111  167    4.00000    0.00000   100%  39.4
104s
 11151  1989    2.00000  126  799    4.00000    0.00000   100%  39.4
106s
 11156  1993    0.00000   95  899    4.00000    0.00000   100%  39.4
110s
 11161  1997    0.00000   80  745    4.00000    0.00000   100%  41.4
115s
 11172  1995    0.00000   86  810    4.00000    0.00000   100%  41.5
120s
 11327  2011    1.50000  103  793    4.00000    0.00000   100%  41.9
125s
 13947  2281    0.05000  135  125    4.00000    0.00000   100%  38.8
130s
 14694  2209    0.00000  120  134    4.00000    0.00000   100%  38.1
141s
 15245  2192    1.00000  133  481    4.00000    0.00000   100%  38.6
145s
 15440  2157 infeasible  124         4.00000    0.00000   100%  38.9
151s
 15859  2036    0.00000  117  413    4.00000    0.00000   100%  39.3
155s
H16720  1586                       3.0000000    0.00000   100%  38.5
158s
 17547  1857     cutoff  148         3.00000    0.00000   100%  38.8
160s
 18748  2244 infeasible  135         3.00000    0.00000   100%  40.0
169s
 18765  2226    2.00000  127  147    3.00000    0.00000   100%  40.2
170s
```

```
 19483  2378    1.00000  138  126    3.00000    0.00000   100%  42.0
175s
 21631  3164    0.00000  146   78    3.00000    0.00000   100%  40.1
262s
 23099  3301    0.18610  126  854    3.00000    0.00000   100%  39.6
265s
 23979  3471    0.38281  145  832    3.00000    0.00000   100%  41.9
270s
 24591  3587    0.40793  142  741    3.00000    0.00000   100%  42.8
288s
 24940  3675    0.14537  122  761    3.00000    0.00000   100%  42.9
290s
 25844  3855 infeasible  111         3.00000    0.00000   100%  44.1
295s
 26479  3963 infeasible  121         3.00000    0.00000   100%  45.7
300s
 27055  4121 infeasible  103         3.00000    0.00000   100%  47.1
305s
 28145  4342 infeasible  134         3.00000    0.00000   100%  48.8
310s
 29611  4850 infeasible  127         3.00000    0.00000   100%  49.6
315s
 29852  4987 infeasible  131         3.00000    0.00000   100%  49.8
324s
 29859  4940 infeasible  132         3.00000    0.00000   100%  49.8
325s
 31338  5458 infeasible  148         3.00000    0.00000   100%  50.1
331s
 32595  5900    0.00000  121   87    3.00000    0.00000   100%  49.9
335s
 34154  6361    0.00000  123   78    3.00000    0.00000   100%  49.6
353s
 34983  6538    1.00000  167  185    3.00000    0.00000   100%  49.6
356s
 36418  7065 infeasible  175         3.00000    0.00000   100%  49.3
360s
 37912  7464    0.60000  115  851    3.00000    0.00000   100%  49.5
365s
 40059  8087 infeasible  119         3.00000    0.00000   100%  49.4
370s
 42578  8909 infeasible  121         3.00000    0.00000   100%  48.6
375s
 44363  9529    0.37752  129  626    3.00000    0.00000   100%  48.2
380s
 46672 10363 infeasible  110         3.00000    0.00000   100%  48.0
385s
 48423 10773 infeasible  129         3.00000    0.00000   100%  48.2
390s
 50023 10993     cutoff  148         3.00000    0.00000   100%  48.2
395s
 50268 11074    1.02461  132  924    3.00000    0.00000   100%  48.3
403s
 50539 11084    0.00000  105  862    3.00000    0.00000   100%  48.5
405s
 51805 11359    0.00000  113  700    3.00000    0.00000   100%  48.9
410s
 53119 11898 infeasible  169         3.00000    0.00000   100%  48.6
415s
 54260 12206     cutoff  130         3.00000    0.00000   100%  48.9
420s
```

```
 57132 13177    0.25139  144  376    3.00000    0.00000   100%  48.6
425s
 57626 13269 infeasible  119         3.00000    0.00000   100%  48.9
430s
 58891 13576    1.00000  129  411    3.00000    0.00000   100%  49.3
447s
 59295 13644    0.00000  103  978    3.00000    0.00000   100%  49.5
451s
 59705 13699    0.00000  122  790    3.00000    0.00000   100%  50.0
455s
 61154 14021    2.00000  126  457    3.00000    0.00000   100%  50.5
460s
 62038 14222 infeasible  122         3.00000    0.00000   100%  50.9
465s
 64118 14796    0.00000  143   82    3.00000    0.00000   100%  50.6
470s
 65974 15454 infeasible  114         3.00000    0.00000   100%  50.5
475s
 67551 15865    0.00000  124  132    3.00000    0.00000   100%  50.7
498s
 67938 15916 infeasible  122         3.00000    0.00000   100%  50.8
501s
 68773 16050 infeasible  153         3.00000    0.00000   100%  51.2
505s
 69510 16202    0.00000  124  777    3.00000    0.00000   100%  51.5
510s
 71617 16729 infeasible  155         3.00000    0.00000   100%  51.6
515s
 72537 16813 infeasible  138         3.00000    0.00000   100%  52.0
520s
 73649 17064 infeasible  140         3.00000    0.00000   100%  52.6
525s
 74400 17236    0.00000  120  862    3.00000    0.00000   100%  52.9
530s
 75085 17539 infeasible  132         3.00000    0.00000   100%  52.9
547s
 75410 17637    1.00000  137  568    3.00000    0.00000   100%  53.0
550s
 76148 17768    2.00000  125  748    3.00000    0.00000   100%  53.3
555s
 77387 18078 infeasible  146         3.00000    0.00000   100%  53.8
560s
 78622 18402 infeasible  114         3.00000    0.00000   100%  54.0
565s
 80979 19103 infeasible  133         3.00000    0.00000   100%  53.7
570s
 82679 19360 infeasible  151         3.00000    0.00000   100%  53.9
576s
 83252 19488    1.00000  119  873    3.00000    0.00000   100%  54.1
593s
 83257 19451    1.00000  119  756    3.00000    0.00000   100%  54.1
595s
 84174 19661 infeasible  130         3.00000    0.00000   100%  54.4
600s
 84650 19732 infeasible  104         3.00000    0.00000   100%  54.8
607s
 85031 19727 infeasible  110         3.00000    0.00000   100%  55.1
611s
 86036 19839    0.19240  108  942    3.00000    0.00000   100%  55.4
616s
```

```
 86604 19923    1.00000  124  603    3.00000    0.00000   100%  55.8
620s
 87711 20142    0.08522  134  765    3.00000    0.00000   100%  56.2
625s
 88876 20422    1.00000  129  499    3.00000    0.00000   100%  56.7
631s
 89633 20582 infeasible  126         3.00000    0.00000   100%  57.0
635s
 90529 20780    1.00000  151  149    3.00000    0.00000   100%  57.6
641s
 92211 21253 infeasible  131         3.00000    0.00000   100%  57.8
646s
 92550 21349    0.00000  118  786    3.00000    0.00000   100%  57.9
651s
 93039 21405 infeasible  114         3.00000    0.00000   100%  58.1
655s
 93958 21695    0.25817  119  881    3.00000    0.00000   100%  58.4
672s
 94000 21658    0.09648  118  909    3.00000    0.00000   100%  58.5
675s
 94769 21806 infeasible  108         3.00000    0.00000   100%  59.0
681s
 95188 21905    0.00000  108  945    3.00000    0.00000   100%  59.1
685s
 95776 22004    0.00000  102  806    3.00000    0.00000   100%  59.4
690s
 97412 22453 infeasible  118         3.00000    0.00000   100%  59.7
696s
 98430 22664    1.00000  121  671    3.00000    0.00000   100%  59.9
700s
 99670 22881    1.00000  120  152    3.00000    0.00000   100%  60.3
706s
 100817 23193 infeasible  126         3.00000    0.00000   100%  60.4
711s
 101627 23460     cutoff  120         3.00000    0.00000   100%  60.5
728s
 101771 23459    0.54832  127  766    3.00000    0.00000   100%  60.6
731s
 102332 23577    0.09733  129  731    3.00000    0.00000   100%  60.8
735s
 102829 23603    0.84470  112  772    3.00000    0.00000   100%  61.1
740s
 104133 23824    1.00000  111  140    3.00000    0.00000   100%  61.5
746s
 105252 24061 infeasible  111         3.00000    0.00000   100%  61.6
751s
 105724 24026    0.52874  123  804    3.00000    0.00000   100%  61.9
756s
 106690 24170 infeasible  106         3.00000    0.00000   100%  62.2
763s
 106788 24152 infeasible  127         3.00000    0.00000   100%  62.3
765s
 107307 24182    0.83333  128   88    3.00000    0.00000   100%  62.6
770s
 107743 24330 infeasible  110         3.00000    0.00000   100%  62.8
784s
 107779 24290 infeasible  115         3.00000    0.00000   100%  62.8
787s
 108413 24450    1.24541  117  690    3.00000    0.00000   100%  63.0
792s
```

```
 109635 24955 infeasible  122         3.00000    0.00000   100%  63.2
796s
 110748 25317 infeasible  134         3.00000    0.00000   100%  63.3
801s
 111898 25661 infeasible  124         3.00000    0.00000   100%  63.4
805s
 112732 25915 infeasible  134         3.00000    0.00000   100%  63.5
819s
 112770 25881    0.49000  124  787    3.00000    0.00000   100%  63.6
821s
 113261 25959    0.13001  120  768    3.00000    0.00000   100%  63.8
827s
 113749 26013    0.50010  101  769    3.00000    0.00000   100%  64.0
831s
 114529 26155    0.00000  112  906    3.00000    0.00000   100%  64.3
836s
 115513 26387 infeasible  130         3.00000    0.00000   100%  64.6
842s
 115773 26413    0.00000  115  809    3.00000    0.00000   100%  64.7
846s
 116551 26539    0.04348  133  969    3.00000    0.00000   100%  65.0
851s
 117683 26914    1.00000  130  134    3.00000    0.00000   100%  65.0
856s
 118411 27034 infeasible  145         3.00000    0.00000   100%  65.1
868s
 119066 27256 infeasible  123         3.00000    0.00000   100%  65.0
871s
 119858 27362    0.00000  118  913    3.00000    0.00000   100%  65.1
876s
 120895 27652     cutoff  116         3.00000    0.00000   100%  65.3
881s
 121106 27690    0.01083  130  952    3.00000    0.00000   100%  65.5
899s
 121422 27749    0.05000  131  906    3.00000    0.00000   100%  65.4
902s
 121682 27775    0.00000  113  978    3.00000    0.00000   100%  65.6
905s
 122105 27829 infeasible  119         3.00000    0.00000   100%  66.0
911s
 122602 27889    0.06872  112  945    3.00000    0.00000   100%  66.3
917s
 123328 27996    1.05882  130  638    3.00000    0.00000   100%  66.7
922s
 123523 27975    0.00000  122  741    3.00000    0.00000   100%  66.9
925s
 124381 28173    1.20682  125  845    3.00000    0.00000   100%  67.2
930s
 125337 28394    0.51986  132  943    3.00000    0.00000   100%  67.5
936s
 126308 28598 infeasible  115         3.00000    0.00000   100%  67.8
943s
 126535 28593    0.00000  138  451    3.00000    0.00000   100%  68.1
954s
 126635 28568 infeasible  107         3.00000    0.00000   100%  68.2
957s
 126864 28569 infeasible  123         3.00000    0.00000   100%  68.4
960s
 127438 28639 infeasible  120         3.00000    0.00000   100%  68.8
967s
```

```
 127634 28684    0.00000  112  987    3.00000    0.00000   100%  69.0
970s
 128493 28953 infeasible  134         3.00000    0.00000   100%  69.4
977s
 128692 28981 infeasible  116         3.00000    0.00000   100%  69.6
986s
 128714 28934    1.00000  111  880    3.00000    0.00000   100%  69.7
990s
 129077 28921 infeasible  130         3.00000    0.00000   100%  70.1
997s
 129252 28897    1.00000  118  683    3.00000    0.00000   100%  70.4
1000s
 129649 28885    1.00000  115  541    3.00000    0.00000   100%  70.9
1008s
 130226 29077     cutoff  122         3.00000    0.00000   100%  71.0
1011s
 130462 29140    0.00000  118  941    3.00000    0.00000   100%  71.2
1016s
 130768 29226 infeasible  148         3.00000    0.00000   100%  71.3
1020s
 131258 29234    0.00000  107  872    3.00000    0.00000   100%  71.7
1030s
 132022 29333 infeasible  120         3.00000    0.00000   100%  72.1
1037s
 132216 29292    0.00000  122  940    3.00000    0.00000   100%  72.3
1041s
 132478 29323 infeasible  115         3.00000    0.00000   100%  72.5
1051s
 132517 29286     cutoff  115         3.00000    0.00000   100%  72.6
1055s
 132994 29285 infeasible  122         3.00000    0.00000   100%  73.1
1064s
 133231 29306 infeasible  122         3.00000    0.00000   100%  73.4
1068s
 133605 29391 infeasible  128         3.00000    0.00000   100%  73.6
1072s
 134486 29774 infeasible  122         3.00000    0.00000   100%  73.8
1081s
 134557 29723    0.14972  123  903    3.00000    0.00000   100%  73.9
1085s
 134843 29734    0.00000  134  823    3.00000    0.00000   100%  74.1
1090s
 135065 29733    1.00000  122  883    3.00000    0.00000   100%  74.3
1095s
 136039 29925 infeasible  117         3.00000    0.00000   100%  74.8
1103s
 136283 29898    0.00000  125  181    3.00000    0.00000   100%  75.0
1108s
 136524 29947    1.00000  176  167    3.00000    0.00000   100%  75.2
1284s
 136526 29948    1.00000  137  864    3.00000    0.00000   100%  75.2
1285s
 136533 29953    1.00000  144  717    3.00000    0.00000   100%  75.2
1290s
 136538 29957 infeasible   94         3.00000    0.00000   100%  75.3
1295s
 136545 29960    0.00000   97  803    3.00000    0.00000   100%  75.3
1301s
 136557 29960    0.00000   99  480    3.00000    0.00000   100%  75.3
1305s
```

```
 136608 29967    0.00000  104  795    3.00000    0.00000   100%  75.3
1310s
 136876 29934 infeasible  133         3.00000    0.00000   100%  75.3
1315s
 137299 29921    1.00000  119  605    3.00000    0.00000   100%  75.2
1320s
 137397 29898 infeasible  152         3.00000    0.00000   100%  75.3
1325s
 137515 29886 infeasible  137         3.00000    0.00000   100%  75.2
1331s
 137802 29857     cutoff  133         3.00000    0.00000   100%  75.2
1335s
 138637 29804    0.01125  125  839    3.00000    0.00000   100%  75.2
1340s
 139468 29800    0.00000  118  890    3.00000    0.00000   100%  75.2
1345s
 139957 29629 infeasible  131         3.00000    0.00000   100%  75.4
1350s
 140370 29548    0.00547  117  745    3.00000    0.00000   100%  75.6
1355s
 141967 29583 infeasible  127         3.00000    0.00000   100%  75.5
1360s
 142789 29410    2.00000  127  759    3.00000    0.00000   100%  75.7
1366s
 143994 29442    1.00000  143  112    3.00000    0.00000   100%  75.6
1371s
 145293 29375    0.01000  118  803    3.00000    0.00000   100%  75.6
1377s
 145597 29254    1.00000  119  793    3.00000    0.00000   100%  75.6
1380s
 147110 29293    1.00000  159  815    3.00000    0.00000   100%  75.4
1385s
 147118 29298    1.00000  119  803    3.00000    0.00000   100%  75.4
1390s
 147131 29301    0.00000  108  438    3.00000    0.00000   100%  75.6
1395s
 147141 29305    0.00000  111  350    3.00000    0.00000   100%  75.6
1400s
 147181 29320    0.00000  122  613    3.00000    0.00000   100%  75.6
1405s
 147401 29303 infeasible  124         3.00000    0.00000   100%  75.6
1410s
 147614 29261    1.00000  127  556    3.00000    0.00000   100%  75.6
1415s
 147854 29209 infeasible  119         3.00000    0.00000   100%  75.6
1420s
 148021 29200    0.00000  119  674    3.00000    0.00000   100%  75.7
1425s
 148173 29206 infeasible  155         3.00000    0.00000   100%  75.7
1431s
 148225 29186    1.96000  128  767    3.00000    0.00000   100%  75.7
1436s
 148488 29099     cutoff  129         3.00000    0.00000   100%  75.9
1440s
 150053 28994    1.02068  142  631    3.00000    0.00000   100%  75.6
1445s
 151749 28575     cutoff  143         3.00000    1.00000  66.7%  75.3
1450s
 154051 28101    1.00000  146  660    3.00000    1.00000  66.7%  74.8
1455s
```

```
 156659 27645 infeasible  146         3.00000    1.00000  66.7%  74.3
1460s
 157811 27535    2.00000  145  680    3.00000    1.00000  66.7%  74.1
1465s
 157822 27545    1.00000  114  128    3.00000    1.00000  66.7%  74.2
1471s
 157832 27544    1.00000  117  127    3.00000    1.00000  66.7%  74.2
1475s
 157868 27550 infeasible  121         3.00000    1.00000  66.7%  74.2
1480s
 158031 27529    1.00000  129  624    3.00000    1.00000  66.7%  74.2
1485s
 158631 27491    1.00000  155  117    3.00000    1.00000  66.7%  74.1
1490s
 159694 27437 infeasible  143         3.00000    1.00000  66.7%  74.0
1495s
 159732 27425 infeasible  143         3.00000    1.00000  66.7%  74.0
1501s
 160420 27265    1.00000  142  416    3.00000    1.00000  66.7%  74.0
1505s
 165363 26228 infeasible  159         3.00000    1.00000  66.7%  73.0
1510s
 170354 25089 infeasible  146         3.00000    1.00000  66.7%  72.0
1515s
 173867 23998    1.00000  143  642    3.00000    1.00000  66.7%  71.6
1520s
 178837 23064 infeasible  135         3.00000    1.00000  66.7%  70.9
1525s
 183700 22107    1.00730  156  221    3.00000    1.00000  66.7%  70.3
1530s
 188316 20909     cutoff  171         3.00000    1.00000  66.7%  69.6
1536s
 192341 19939 infeasible  156         3.00000    1.00000  66.7%  69.0
1540s
 196622 18809    1.00000  156  587    3.00000    1.00000  66.7%  68.5
1545s
 200772 17642    2.00000  158   59    3.00000    1.00000  66.7%  68.0
1550s
 204516 16423 infeasible  138         3.00000    1.00000  66.7%  67.5
1555s
 207567 13642    2.00000  143   78    3.00000    2.00000  33.3%  67.1
1560s
 209440 11717 infeasible  154         3.00000    2.00000  33.3%  66.9
1565s
 210851 10153    2.00000  137  510    3.00000    2.00000  33.3%  66.8
1570s

Cutting planes:
  Gomory: 2
  Cover: 27
  Implied bound: 20
  Clique: 47
  MIR: 8
  StrongCG: 3
  Flow cover: 50
  GUB cover: 11
  Inf proof: 66
  Zero half: 4

Explored 211389 nodes (14241416 simplex iterations) in 1572.23 seconds
```

```
Thread count was 8 (of 8 available processors)

Solution count 8: 3 4 8 ... 87

Optimal solution found (tolerance 3.00e-02)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap
0.0000%
```

Figure B.4.: Gurobi log of problem 30_2-2-96 with MIPStart

C. Computational results of the customized branch-and-bound algorithm

The columns of Table C.1 are defined as follows

δ - parameter defined in Section 4.2.2 determining how often the second optimization problem is solved

Problem - problem in the usual notation

BB Nodes - number of nodes explored provided by the customized branch-and-bound algorithm, non considering the nodes within the solve calls at each node

Nodes - number of explored nodes during the Gurobi MIP solving process

Optim calls - total number of optimization calls over all nodes

Time - computational time in s, applies for all of the three columns

	δ=0			**δ=4**			**Gurobi**	
Problem	**BB Nodes**	**Optim calls**	**Time**	**BB Nodes**	**Optim calls**	**Time**	**Nodes**	**Time**
10_4-1-11	25	50	3.435	104	130	11.462	1	0.15
10_4-1-12	19	38	3.115	117	146	17.557	1	0.15
10_4-1-13	17	34	6.584	117	146	29.246	65	0.22
10_4-1-14	23	46	3.892	222	277	31.540	1	0.31
10_4-1-15	16	32	2.213	107	133	11.739	1	0.21
10_4-1-16	22	44	2.777	154	192	13.195	48	0.20

	δ=0			**δ=4**			**Gurobi**	
Problem	**BB Nodes**	**Optim calls**	**Time**	**BB Nodes**	**Optim calls**	**Time**	**Nodes**	**Time**
10_4-1-17	48	56	7.174	207	258	32.364	178	0.29
10_4-1-18	56	112	9.005	232	290	21.936	1	0.13
10_4-1-19	16	32	4.211	121	151	26.113	43	0.29
10_4-1-20	12	24	1.965	106	162	12.463	1	0.16
15_4-1-11	27	54	74.003	408	510	233.049	255	0.84
15_4-1-12	147	294	202.108	779	973	3044.674	2098	3.34
15_4-1-13	79	158	157.804	858	1072	2278.826	1125	2.26
15_4-1-14	76	152	83.478	464	580	2198.329	251	0.66
15_4-1-15	30	60	1768.049	122	152	2748.781	126	0.81
15_4-1-16	58	118	106.561	774	967	2788.792	333	0.76
15_4-1-17	69	138	48.933	460	575	314	539	1.34
15_4-1-18	20	40	9.111	367	458	190.308	1	0.73
15_4-1-19	37	74	25.237	312	390	471.718	93	0.65
15_4-1-20	38	76	27.254	343	428	287.763	284	0.73

Table C.1.: Computational results Heuristic

D. Computational results of the sequence-to-sequence network

The columns of Table D.1 are defined as follows

aTSS - adjusted test set size, in number of tuples of sequences; the test set is 30% of the data set; the aTSS is obtained by removing calculated sequences of length $\neq n$ from the test set

μ(ICC) - mean value of number of incorrect color codes over all sequences in aTSS as defined in (5.17)

μ(CPCC) - mean value of number of correctly placed color codes

P - number of permutations, sequences with NICC$=0$, share in brackets

$\mu(d_{NN})$ - mean value of distance defined in (5.18), defined for permutations and thus calculated for sequences with ICC$=0$

RS - number of removed sequences; tuples of sequences, where the length of the calculated sequence is not equal to n are removed from the set before analysing

OS - number of optimal solutions, $\mu(d_{NN})=0$

TT - training time of the neural network in s

The mean values in Table D.1 are rounded up to two decimal points and to three decimal points for **P**, respectively.

Data set	**aTSS**	**μ(ICC)**	**μ(CPCC)**	**P**	$\mu(d_{NN})$	**OS**	**RS**	**TT**
NN-10_4-1-min	267	3.01	5.52	16(0.06)	7	0	0	516

Data set	aTSS	μ(ICC)	μ(CPCC)	P	$\mu(d_{NN})$	OS	RS	TT
NN-10_4-1-max	1235	2.06	6.12	285(0.23)	6.05	20	5	1800
NN-10_4-2	3360	1.48	6.02	1302(0.39)	5.33	142	0	5227
NN-15_4-1-min	300	3.51	8.22	13(0.04)	15.08	0	0	711
NN-15_4-1-max	1498	3.44	8.39	99(0.07)	13.8	1	1	3028
NN-15_4-2	3350	2	9.65	790(0.24)	7.38	45	10	7178
NN-20_4-1-min	298	6.56	10.38	2(0.01)	53	0	2	924
NN-20_4-1-max	1475	3.06	12.39	141(0.1)	13.82	2	25	4043
NN-20_4-2	3324	2.91	12.99	393(0.12)	12.68	9	36	9453
NN-30_4-1-min	153	20.35	13.27	0(0)	NaN	0	147	1512
NN-30_4-1-max	1498	4.34	19.59	43(0.03)	36.33	0	2	6615
NN-30_4-2	3358	4.26	20.93	173(0.05)	25.28	0	2	15 352
NN-30_8-1-min	168	17.55	11.55	0(0)	NaN	0	132	1666
NN-30_8-1-max	1491	7.53	14.81	4(0)	66	0	9	6997
NN-30_8-2	3323	8.72	15.46	1(0)	70	0	37	17 613
NN-40_4-1-min	0	NaN	NaN	0(0)	NaN	0	300	2209
NN-40_4-1-max	708	7.95	23.12	1(0)	70	0	792	10 056
NN-40_4-2	3107	5.07	29.71	94(0.03)	37.17	1	253	26 851
NN-40_8-1-min	7	33.14	13.71	0(0)	NaN	0	293	1247
NN-40_8-1-max	2759	10.95	18.25	1(0)	132	0	123	9957
NN-40_8-2	3356	8.47	22.18	1(0)	38	0	4	26 138
NN-50_4-1-min	0	NaN	NaN	0(0)	NaN	0	300	2773
NN-50_4-1-max	548	8.16	27.95	4(0.01)	164	0	952	14 184
NN-50_4-2	3360	6.5	32.79	34(0.01)	94.76	0	0	36 625
NN-50_8-1-min	0	NaN	NaN	0(0)	NaN	0	300	2653
NN-50_8-1-max	212	12.76	22.98	0(0)	NaN	0	1288	16 585

Data set	**aTSS**	μ**(ICC)**	μ**(CPCC)**	**P**	$\mu(d_{NN})$	**OS**	**RS**	**TT**
NN-50_8-2	3360	9.28	27.79	2(0)	137	0	0	37 006

Table D.1.: Results for the neural network

The columns of Table D.2 are defined as in Table D.1. We use the following parameter settings

NHU - number of hidden units

EmDim - embedding dimension of the word embedding layer in the encoder

configuration	aTSS	μ(ICC)	μ(CPCC)	P	$\mu(d_{NN})$	OS	RS	TT
				NN-20_4-2				
NHU= 20	1500	4.88	11.53	34(0.02)	34	0	0	3261
NHU= 200	1500	3.18	11.43	129(0.09)	19.57	0	0	5987
EmDim= 2 NHU= 50	1500	3.39	12.83	105(0.07)	18.06	0	0	4654
				NN-40_4-2				
NHU= 20	165	10.04	21.4	0(0)	NaN	0	1235	7494
NHU= 200	0	NaN	NaN	0(0)	NaN	0	1500	19 360

Table D.2.: Results for parameter configurations