Wageningen University - Department of Social Sciences

Operations Research and Logistics

MSc Thesis

Comparing Solvers

Piecewise Linear- & Logarithmic Approximations

Abstract

Purpose – Reformulations of non-convex non-linear programming problems to mixed-integer linear programming problems require the addition of binary variables. This thesis will test if reformulations with less binary variables can improve solving times.

Design/methodology/approach – A Big-M reformulation with (1) a linear number of binary variables and (2) a logarithmic number of binary variables is tested on various solvers and solving times are compared.

Findings – Computational results indicate that a logarithmic Big-M reformulation is potentially suitable for the GLPK and CBC solvers. Although further testing on larger problems is still required.

Originality/value - To the best of the author's knowledge, there have not been any papers comparing reformulation techniques on various solvers. Practitioners without access to faster commercial solvers could use the logarithmic Big-M reformulation to potentially speed up the solving process when compared to using a linear Big-M reformulation with the GLPK or CBC solvers.

Research limitations/implications - Further testing on larger real-world problems should be performed to verify that the logarithmic Big-M reformulation is suitable to speed up solving times for GLPK and CBC.

Word count: 7,432

March, 2023

Student	Michel Vrieswijk
Registration number	1168460
MSc program	Master Biobased Sciences
Specialisation	Biobased and Circular Economy
Commissioner	Research project
Supervisor	dr. Peter Kirst
Examiner/2nd supervisor	dr.ir. GDH (Frits) Claassen
Thesis code	ORL-80436

WAGENINGEN
UNIVERSITY & RESEARCH

Comparing Solvers:
Piecewise Linear- & Logarithmic Approximations

Master Thesis

Operations Research & Logistics Group
Wageningen University & Research (WUR)
Wageningen, The Netherlands

Student name: Michel Vrieswijk
Student number: 1168460
Supervisor: dr. Peter Kirst
Course code: ORL-80436
Date: March 31, 2023

Abstract

Purpose - Optimization problems containing non-convex non-linearities are often difficult to solve. Reformulations to mixed-integer linear programming (MILP) problems can, in some cases, make the problem faster to solve. However, these reformulations require the addition of binary variables which adds to the computational difficulty. This thesis will investigate if solvers benefit from a reduced number of binary variables.

Design/methodology/approach - The chosen reformulation technique is the Big-M method. Several test problems containing a non-linear constraint are created and reformulated with (1) a linear number of binary variables and (2) a logarithmic number of binary variables. These test problems are then solved using Gurobi, Cplex, Xpress, Glpk, Cbc, and Scip. Solving times are obtained and compared.

Findings - Computational results indicate that the logarithmic Big-M reformulation could be suitable to speed up solver performance for Glpk and Cbc, but additional tests are required. Results from Scip are inconclusive, the solver showed an improved solving time for one test problem. Gurobi, Cplex, and Xpress all performed worse with the logarithmic reformulation. Although, Xpress could not be used on all test problems because the author was limited to the community license.

Originality/value - To the best of the author's knowledge, there have not been any papers comparing reformulation techniques on various solvers. Practitioners who do not have access to faster commercial solvers could use the logarithmic Big-M reformulation to potentially speed up the solving process when compared to using a regular Big-M reformulation with Glpk or Cbc as solvers.

Research limitations/implications - Further testing on larger real-world problems should be performed to verify that the logarithmic Big-M reformulation is suitable to speed up solving times for Glpk and Cbc. Further research should also focus on *locally ideal* reformulation techniques. The Big-M reformulation is relatively weak compared to other methods in the literature which could affect solving times.

Keywords: Piecewise linear approximation; Mixed-integer programming; Solvers; Big-M reformulation; Non-linear functions

Chapter 1

Introduction

Over the last decades, mixed-integer linear programming (MILP) has been extensively studied in theory and practice. Nowadays, it is an indispensable optimization technique with application areas including health care, logistics, chemical engineering among many others (Guignard-Spielberg & Spielberg, 2005; Jünger et al., 2010). The success of MILP can be attributed to the availability of state-of-the-art solvers (e.g., Cplex, Gurobi, Fico Xpress, Scip) as well as the modeling flexibility (Vielma, 2015).

However, in practice, many phenomena are non-linear and require the introduction of non-linear functions into the optimization model. An optimization model containing continuous and integer variables as well as at least one non-linear function can be classified as a mixed-integer non-linear programming (MINLP) problem. The addition of non-linearities, which are often non-convex, causes additional computational difficulties to an already $\mathcal{NP}$-hard problem. Therefore, a major challenge is being able to solve MINLPs in acceptable computational time to global optimality (if a global optimum exists) (Bertsimas et al., 2021; Fard et al., 2021). Hence, in the literature, there are several proposed techniques to deal with these non-linearities.

One of these techniques is piecewise linearization, where non-linear functions are approximated by piecewise linear functions. Then, a MINLP problem can be transformed to an approximate MILP problem with the goal of reducing computational difficulties. Such transformations especially make sense if the existing MINLP problem contains only a few non-linearities. For an overview of linearization and transformation techniques, the reader is referred to Asghari et al. (2022).

One of the problems with piecewise linearizations of non-convex non-linear functions is the requirement to add additional binary variables, which in turn cause additional challenges. Therefore, techniques have been developed in the literature to reduce the number of binary variables when piecewise

linearization techniques are used. From a theoretical point of view, this should lead to a significant computational advantage (Carrion & Arroyo, 2006; Vielma & Nemhauser, 2011; Yildiz & Vielma, 2013).

One of these proposed techniques reduces the number of binary variables from linear to binary logarithmic ($\log_2$). To the best of the author's knowledge, the first paper to investigate the possibility of modeling with a logarithmic number of binary variables is by Watters (1967). Another problem is that due to the complex nature of state-of-the-art solvers it is difficult to, with high accuracy, predict the best formulation technique for a certain problem (Vielma, 2015).

Nonetheless, not the entire scientific community is convinced that reducing the number of binary variables is computationally advantageous over standard solving methods for MILPs. Owen and Mehrotra (2002) critically discuss the efficiency of approaches resulting in fewer binary variables. However, there have been contradicting articles that demonstrate that solving MILPs with a reduced number of binary variables can decrease the computational time required to solve these problems (Adams & Henry, 2012; Li et al., 2013, 2017; Zhao et al., 2021). It is thus unclear why some authors have seen promising results from a reduced number of binary variables, whereas Owen and Mehrotra (2002) have shown computational results supporting their claims that such remodeling techniques for MILPs leads to difficulties. To address part of this research gap, this thesis will examine the impact of solvers on how much time it takes to solve reformulated MILPs with linear binary variables and logarithmic binary variables.

To address the discussed research gap, the following research questions (RQs) have been formulated:

RQ1. Which suitable piecewise linearization techniques exist in the literature?

RQ2. How can the number of binary variables be reduced?

RQ3. What is the effect of reducing the number of binary variables on solving mixed-integer optimization problems with different solvers?

The specific contribution of this thesis will be the impact of reducing the number of binary variables using a specific piecewise linearization technique on solver performance. It is interesting to see if some solvers show greater benefit from this approach than others.

RQ1 and RQ2 will be answered through an extensive systematic literature review. Based on the results, a suitable piecewise linearization technique will be chosen. It is required that the chosen technique can be reformulated with a reduced number of binary variables as well. Because then, in

order to answer RQ3 it is possible to make a fair comparison between a problem with a linear number of binary variables and a reduced number of binary variables. Therefore, several MINLPs will be reformulated to MILPs. These reformulations will be done in Python. During the thesis, two Python scripts will be developed where non-linear functions are approximated using piecewise linear functions. These reformulation techniques will be taken from existing literature. One of these scripts will add a linear number of binary variables, whereas the other will implement a technique using a reduced number of binary variables. Thereafter, the reformulated MILPs will be solved using several commercial and open-source solvers to determine whether some solvers benefit more from an approach with less binary variables than others. To the best of the author's knowledge, there as of yet have not been any studies where the effect of reducing the number of binary variables has been tested on various solvers.

The remainder of this thesis is organized as follows. In Chapter 2 the existing literature on piecewise linear approximations and reformulations techniques is discussed. Chapter 3 is focused on the methodology used to execute the research. Specifically, this section will describe the chosen reformulation techniques in detail for univariate and multivariate functions. In Chapter 4 the research results are presented including computational examples and a discussion. Lastly, Chapter 5 gives conclusions and indicates some directions for further research.

Chapter 2

Literature review

A systematic literature review is undertaken according to the framework proposed by Tranfield et al. (2003). The stages in conducting a systematic review can be summarized as (1) planning, (2) searching, (3) screening, and (4) extraction, synthesis, and reporting. These steps are performed to avoid bias and to guarantee that the acquired literature is relevant (Abidi et al., 2014). For a detailed overview of the steps taken in the systematic literature review, the reader is referred to Appendix A. The next two sections present the outcome of the systematic literature review and form the answers to RQ1 and RQ2.

2.1 Piecewise Linear Approximations

As mentioned before, non-linear functions occur naturally in many application areas, some application areas are cost functions of supply chain problems (Vielma et al., 2010b), production planning (Fourer et al., 1993), operation planning of gas networks (Martin et al., 2006), and appointment scheduling (Jiang et al., 2017), among others.

In the literature, there are various proposed methods to handle non-linear functions through piecewise linear approximation methods. Optimization problems containing non-linearities can be reformulated as MILPs and solved through traditional MILP algorithms. The advantage is that traditional state-of-the-art MILP solvers make use of advanced technology which makes solving MILPs fast and robust (Vielma et al., 2010b). Furthermore, MILP techniques can be used to solve large problems with up to millions of variables (Geißler et al., 2012). The approximation of a non-convex non-linear model through piecewise linear approximation requires the addition of binary variables, continuous variables, and/or constraints to the resulting MILP model (Hu et al., 2013).

2.2 Piecewise Linear Reformulation Techniques

There are numerous reformulation techniques in the literature. Here, some of the most well-known will be discussed. Vielma (2015) gives an overview of advanced formulation techniques for *mixed integer* problems. He emphasizes that due to the complexities of state-of-the-art solvers, it is difficult to predict the best formulation technique for a certain problem with high accuracy.

One well-known mixed-integer formulation for piecewise linear approximations is the incremental cost formulation, first introduced by Markowitz and Manne (1957) and also known as the delta (δ) formulation. This formulation has been studied extensively in for example Croxton et al. (2003), Keha et al. (2004), and Padberg (2000).

Another common method is the convex combination (CC) model, also called the lambda (λ) model, introduced by Dantzig (1960). Some studies that use this formulation are Keha et al. (2004), Lee and Wilson (2001), and Padberg (2000).

An extension of the CC model is the disaggregated convex combination (DCC) model introduced by Meyer (1976) and studied by Croxton et al. (2003), Jeroslow and Lowe (1984), and Sherali (2001). For the CC and DCC logarithmic formulations where the number of variables and/or constraints are significantly reduced exist. Formulations with a logarithmic number

of binary variables will be referred to with the prefix *Log*. For extensive information on these formulations, the reader is referred to Vielma et al. (2010b). It is however important to keep in mind that a reduction in binary variables and constraints does not guarantee a better formulation in the sense of better performance of solvers on such models (Vielma & Nemhauser, 2011).

Li et al. (2009) developed a piecewise linear function model with a logarithmic number of binary variables when compared to traditional techniques, based on a Big-M reformulation. However, Vielma et al. (2010a) showed that the representations by Li et al. (2009) are theoretically and computationally worse to standard MILP formulations for piecewise linear functions.

Padberg (2000) showed that the incremental cost method is superior to the CC method in terms of tightness. When a MILP problem is being solved, the solver typically uses branch-and-bound algorithms. Usually, integrality constraints are relaxed (i.e., any discrete variables can be considered as continuous). Solving the relaxed problem is generally easier because standard linear programming (LP) techniques can be used. When the relaxation is solved, assuming a minimization problem, the result constitutes a lower bound for the original problem. In the incremental cost method, the piecewise linear approximated model is so-called *locally ideal*, whereas the CC method is not. The term *locally ideal* refers to the fact that vertices of its LP relaxation satisfy the integrality constraints of the original MINLP problem (Alkhalifa & Mittelmann, 2022).

Another, slightly less, desirable property of a formulation is *sharpness*. A formulation is called *sharp* if the projection of the vertices of the model to the original set of variables is exactly the convex hull of this set (Alkhalifa & Mittelmann, 2022; Sridhar et al., 2013), this is a property that the CC model does have. Note that, a *locally ideal* formulation is always *sharp*.

The last reformulation technique that we discuss here is the Big-M reformulation. In most commonly used textbooks (e.g., Bazaraa et al. (1993), Hillier and Lieberman (1986), and Taha (2003)) only the CC and/or the incremental cost model are introduced. However, the Big-M reformulation focussing on piecewise linear approximations is hardly ever introduced. Most information on Big-M reformulations is focussed on linear relaxation techniques, and although linear relaxation and piecewise linear approximation share some steps, they are not the same (Alkhalifa & Mittelmann, 2022). An advantage of the Big-M reformulation is that it is quite easy to implement and intuitively to understand compared to most other techniques. However, there are drawbacks as well. In the Big-M reformulation, one or more parameters M are added to the formulation. The value for these parameters has to be chosen wisely, if M is too large this can cause loss of precision and numerical instabilities. Whereas, a too low value may cause a reformulation

Table 2.1: Comparison of piecewise linearization methods with m line segments ($m+1$ breakpoints) for a single variable x.

Method	# of binary variables	# of continuous variables	# of constraints	Source
CC	m	$m+1$	$5+m$	(Hu et al., 2013)
Incremental cost	$m-1$	m	$4+2(m-1)$	(Sridhar et al., 2013)
Big-M	m	0	$1+4m$	-
Log CC	$\lceil \log_2(m) \rceil$	$m+1$	$3+2\lceil \log_2(m) \rceil$	(Vielma et al., 2010b)
Log Big-M	$\lceil \log_2(m) \rceil$	0	$4m$	-

to not converge to the same optimum as the original non-linear problem.

The regular Big-M reformulation method uses a linear number of binary variables. But, there is a logarithmic formulation which reduces the number of binary variables as well. First introduced by Ibaraki (1976) and explained in detail by Vielma (2015, Proposition 9.4 on p. 43). Furthermore, in Vielma (2015, Chapter 6.1 on p. 27-28) traditional Big-M formulations are discussed as well.

A common approach in piecewise linear approximation methods is to split up the domain of a non-linear function in m intervals by using $m+1$ breakpoints. Then, in every interval, a linear function can be constructed to approximate part of the non-linear function. In order to express which of the m intervals has to be selected, it is common practice to utilize m binary variables with the additional requirement that they add up to one. However, this approach can slow down branch-and-bound based MILP solvers because this approach potentially leads to unbalanced branch-and-bound trees (Yildiz & Vielma, 2013). The Big-M, DCC, and CC methods make use of this approach. Modeling m alternatives with binary variables logarithmic in m tends to lead to more balanced branch-and-bound trees, although imbalances may still occur (Yildiz & Vielma, 2013).

According to research by Till et al. (2004) complexity of MILP problems vary from $O(d \cdot n^2)$ to $O(2^d \cdot n^3)$, where n denotes the number of constraints and d the number of binary variables. From this, Hu et al. (2013) deduced that reducing constraints and binary variables has a greater impact on computational efficiency than reducing continuous variables. Thus, if a linearization method generates many additional constraints and binary variables, this will have an adverse effect on computational efficiency.

In Table 2.1 the discussed methods along with the number of binary variables, continuous variables and constraints which are needed are shown for a single variable x.

From the literature, we can make some hypotheses regarding the RQs proposed in Chapter 1 on page 1. For RQ1 the piecewise linearization techniques shown in Table 2.1 are all suitable and used throughout the literature, although some formulations have favorable properties (e.g., being *locally ideal*

or *sharp*). With regard to RQ2, reducing the number of binary variables can be done through logarithmic reformulation techniques, which also eliminate the need to add up binary variables to a value of one. The effect of reducing the number of binary variables, RQ3, should based on the literature lead to reduced computational difficulty. This thesis will test this hypothesis using the Big-M and Log Big-M reformulation techniques for different solvers. Big-M is chosen because it is a relatively straightforward technique compared to most other reformulation techniques. Furthermore, due to the limited amount of available time, this technique is deemed to have the highest chance of being implemented without delays.

Chapter 3

Methodology, research design and data description

The methodology chapter is split up into four distinct sections. Firstly, the notation that will be used throughout the rest of this document is introduced in Section 3.1. Secondly, piecewise linear approximations and the Big-M reformulation of univariate functions will be explained in Section 3.2. Thirdly, the Big-M approach for univariate functions is extended to multivariate functions in Section 3.3. Lastly, this chapter will end with Section 3.4 where the Big-M reformulation with a logarithmic number of binary variables is explained.

3.1 Notation

In general, given a vector $x = (x_1, x_2, \ldots, x_n)$, a MINLP problem may be formulated as

$$\begin{aligned} \min_{x} \quad & g(x) \\ \text{s.t.} \quad & f(x) \lesseqgtr b, \\ & x \in \mathbb{R}^{n-k} \times \mathbb{Z}^{k} \end{aligned} \tag{3.1}$$

with $g : \mathbb{R}^n \mapsto \mathbb{R}$, $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, b is a real m-dimensional vector, and $x \in \mathbb{R}^{n-k} \times \mathbb{Z}^k$ meaning that the first $n-k$ variables are continuous and that the last k variables are integers.

Equation (3.1) reduces to a non-linear programming (NLP) model when $k = 0$ and to a MILP model when f and g are linear. Lastly, the model reduces to a LP model when $k = 0$ *and* f and g are linear.

In this thesis, we will use formulation eq. (3.1) with the additional requirement that the objective function (g) is linear. I.e., non-linearities will only be apparent in the constraints (f).

Furthermore, note that a maximization problem can always be converted to a minimization problem and vice versa by multiplying the objective function by -1 ($\min g(x) = -\max -g(x)$).

3.2 Approximations of Univariate Functions

The general idea of reformulating a MINLP as a MILP problem is quite straightforward, every non-linear function gets approximated by a piecewise linear function (or more accurately, a piecewise affine function).

We consider a general univariate non-linear function f. The function f is continuous and x is in the domain $[a_l, a_u] \subseteq \mathbb{R}$. Then, let $\widehat{f}(x)$ be a piecewise linear function of $f(x)$, where the domain $[a_l, a_u]$ is broken up into m intervals by $m+1$ breakpoints. Such that $a_l = a_0 < a_1 < a_2 < \cdots < a_m = a_u$. Now, the function f is evaluated at every breakpoint, and the lines connecting the points $(a_{i-1}, f(a_{i-1}))$ and $(a_i, f(a_i))$ now form the piecewise linear approximation of f. This is graphically represented in Figure 3.1.

We denote the piecewise linear approximation of a non-linear function $f(x)$ on a sub-interval $i = [a_{i-1}, a_i]$ by

$$f_i(x) := \alpha_i x + \beta_i \tag{3.2}$$

$$x \in [a_{i-1}, a_i] \tag{3.3}$$

Where α_i denotes the slope of $f(x)$ on interval i and β_i represents the intercept, which may respectively be calculated by

$$\alpha_i := \frac{f(a_i) - f(a_{i-1})}{a_i - a_{i-1}} \tag{3.4}$$

and

$$\begin{aligned} \beta_i : &= f(a_i) - \frac{f(a_i) - f(a_{i-1})}{a_i - a_{i-1}} a_i \\ &= f(a_i) - \alpha_i \cdot a_i \end{aligned} \tag{3.5}$$

Now, the piecewise linear function $\widehat{f}$ can be written as

$$\widehat{f}(x) := \begin{cases} f_1(x) = \alpha_1 x + \beta_1, & \text{for } x \in [a_0, a_1] \\ f_2(x) = \alpha_2 x + \beta_2, & \text{for } x \in [a_1, a_2] \\ \qquad \vdots & \\ f_m(x) = \alpha_m x + \beta_m, & \text{for } x \in [a_{m-1}, a_m] \end{cases} \tag{3.6}$$

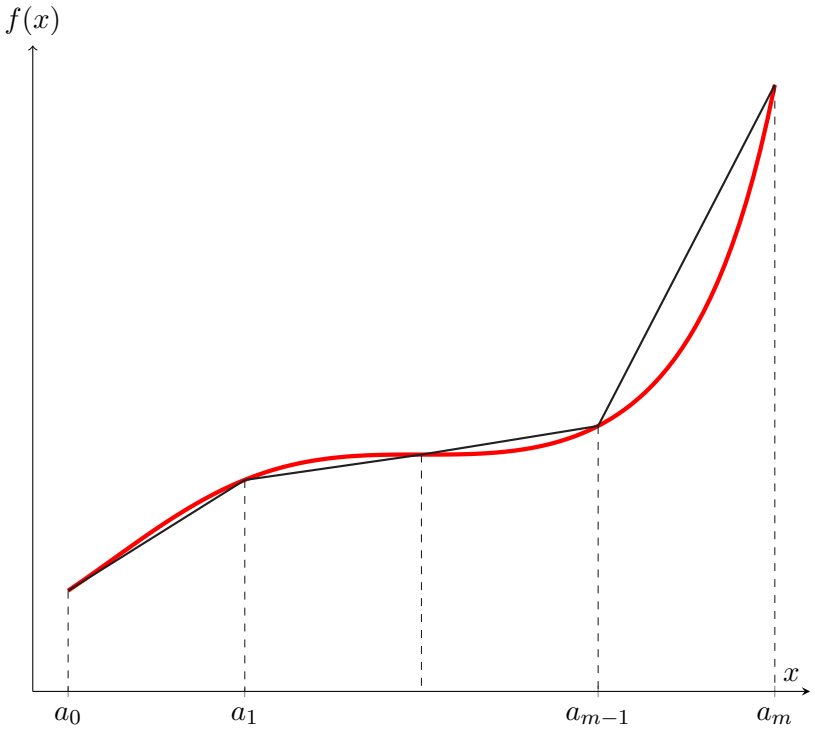


Figure 3.1: Piecewise Linear Approximation $\widehat{f}(x)$ (black) of $f(x)$ (red).

In Figure 3.1 an example of a piecewise linear approximation is given, here the black line depicts the piecewise linear approximation of the red function. The accuracy of a piecewise linear approximation can be increased by adding additional breakpoints. However, this will result in a larger model because additional variables and constraints need to be included in the optimization model. And, hence, this will lead to additional computational difficulty. Therefore, piecewise linear approximation techniques may only benefit functions of few dimensions (Alkhalifa & Mittelmann, 2022; Geißler et al., 2012).

Note that in Figure 3.1 the breakpoints are uniformly spaced. Of course, in practice this can differ, but these techniques are not within the scope of this thesis. In the literature there are some articles that focus on optimizing the locations of break points, the interested reader is referred to Contardo and Ngueveu (2021), Kong and Maravelias (2020), Rebennack and Kallrath (2015), and Ugaz et al. (2019).

After the piecewise linear approximation, we want to add constraints which force x into the correct interval. This is where the Big-M reformulation comes into play. To achieve this, it is necessary to introduce m binary variables, one for every interval i, denoted by $\{y_1, \ldots, y_m\}$ as well as a sufficiently large constant M

$$-(1 - y_i)M + a_{i-1} \leq x \leq a_i + (1 - y_i)M, \qquad \forall i \in \{1, \ldots, m\} \tag{3.7}$$

where

$$\sum_{i=1}^{m} y_i = 1, \tag{3.8}$$

$$y_i \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, m\} \tag{3.9}$$

So, to satisfy eqs. (3.8) and (3.9) only one binary variable, y_i, will become one with the others having a value of zero. Therefore, if $y_i = 1$ eq. (3.7) is reduced to $a_{i-1} \leq x \leq a_i$ and hence x is forced into some interval i. On the other hand, if $y_i \neq 1$ eq. (3.7) reduces to $-M + a_{i-1} \leq x \leq a_i + M$. And thus, depending on the value of M the left-hand-side of eq. (3.7) will become large and negative and the right-hand-side will become large and positive. From this, it is obvious that the value of M should be chosen large enough such that any sub-interval i on the domain $[a_l, a_u]$ is able to become enforced.

Furthermore, it is necessary to ensure that the constraint takes on the right function value with one of the following constraints

$$-(1-y_i)M + b_i \leq f_i(x) \leq b_i + (1-y_i)M, \quad \forall i \in \{1,\dots,m\} \tag{3.10a}$$
$$f_i(x) \leq b_i + (1-y_i)M, \quad \forall i \in \{1,\dots,m\} \tag{3.10b}$$
$$-(1-y_i)M + b_i \leq f_i(x), \quad \forall i \in \{1,\dots,m\} \tag{3.10c}$$

here, eq. (3.10a) is used if the original constraint is an equality constraint; eq. (3.10b) represents a less than or equal to constraint; and eq. (3.10c) represents a greater than or equal to constraint. If in eq. (3.10) some $y_i = 1$ then the expression(s) on the left-hand-side and/or right-hand-side of $f_i(x)$ will become equal to b_i.

For example, if $y_i = 1$ in the case of an equality constraint, eq. (3.10a), will reduce to $b_i \leq f_i(x) \leq b_i$ which is equivalent to $f_i(x) = b_i$.

It is important to choose an appropriate value for the M-parameter(s). In general, one wants to choose M as small as possible while being large enough to avoid an invalid formulation. This is because, the smaller the M-parameter(s) the tighter the reformulation. For guidance on finding the optimal value for the M-parameter, readers are referred to Trespalacios and Grossmann (2015, Section 2.3).

3.3 Approximations of Multivariate Functions

We can extend the formulation from Section 3.2 to multivariate functions. Firstly, consider a non-linear function over n dimensions $f(x_1, x_2, \dots, x_n) :\to \mathbb{R}$ and let us introduce a box, B, defined by

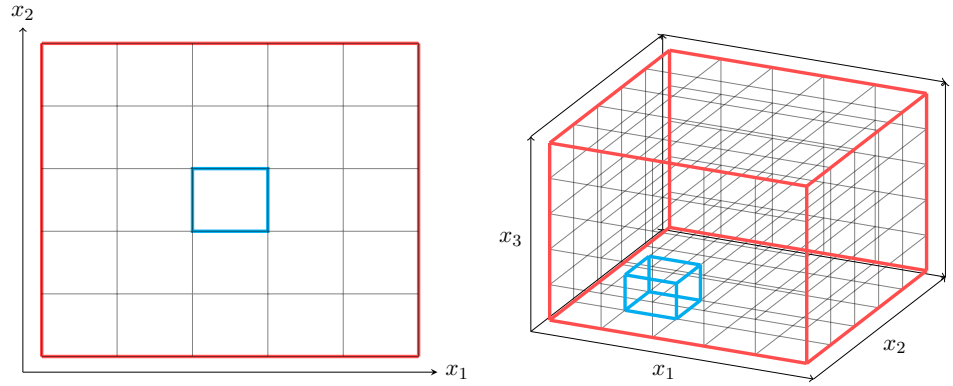
$$B := \{x \in \mathbb{R}^n \mid \underline{x} \leq x \leq \overline{x}\} \tag{3.11}$$

With $\underline{x} \in \mathbb{R}^n$ denoting the lower bounds and with $\overline{x} \in \mathbb{R}^n$ denoting the upper bounds of B. Next, box B is divided into sub-boxes $B_1, \dots, B_m$ such that

$$\bigcup_{i=1}^{m} B_i = B \tag{3.12}$$

Then, the lower bounds of a sub-box B_i are denoted as $\underline{x}_k^i$ $(\forall k \in \{1,\dots,n\})$. And upper bounds of a sub-box B_i are denoted as $\overline{x}_k^i$ $(\forall k \in \{1,\dots,n\})$. Note that in this formulation superscripts are indices, not exponents.

The representation of a box B and sub-boxes B_i can be seen in Figure 3.2. Figure 3.2a shows a possibility for 2-dimensional models, whereas Figure 3.2b



(a) Box B in red. Activated sub-box B_i in blue.

(b) Box B in red. Activated sub-box B_i in blue.

Figure 3.2: Representation of boxes in respectively 2-dimensions (a) and 3-dimensions (b).

shows a possibility in 3-dimensions. Note that, in 1-dimension, the box would look like a line and an activated sub-box would be a line segment.

Figure 3.2 also shows a potential issue, commonly known as *the curse of dimensionality* (Bellman, 1961). In Figure 3.2a there is an orthogonal grid of $5 \cdot 5 = 25$ sub-boxes, when the number of dimensions is increased from 2-dimensional to 3-dimensional as in Figure 3.2b there are already $5 \cdot 5 \cdot 5 = 125$ sub-boxes and so on for even higher dimensions. And these increases have a direct influence on the required number of binary variables, and thus on the computational difficulty to solve a problem in higher dimensions (Feijoo & Meyer, 1988; Misener & Floudas, 2010). In general, the number of boxes that need to be evaluated can be calculated by m^n where $m+1$ breakpoints are used and n dimensions.

Now, to generate a suitable piecewise linear approximation, over every sub-box B_i a local approximation of function f can be calculated. To do this, we will again use affine linear functions similar to the approach in 1 dimension as previously explained in Section 3.2. For every sub-box B_i the lower left corner $(\underline{x}_1^i, \underline{x}_2^i, \ldots, \underline{x}_n^i)^\intercal$ is used as a reference point. Then, the slope into the direction of some x_k can be approximated as follows

$$\alpha_k^i := \frac{f(\overline{x}_k^i, \underline{x}_1^i, \ldots, \underline{x}_{k-1}^i, \underline{x}_{k+1}^i, \ldots, \underline{x}_n^i) - f(\underline{x}_1^i, \ldots, \underline{x}_n^i)}{\overline{x}_k^i - \underline{x}_k^i}, \quad \forall i \in \{1, \ldots, m\} \quad \forall k \in \{1, \ldots, n\} \tag{3.13}$$

and, then the approximation of function f over the domain of sub-box B_i is

given by

$$f_i(x_1, x_2, \ldots, x_n) = f(\underline{x}_1^i, \underline{x}_2^i, \ldots, \underline{x}_n^i) + \alpha_1^i(x_1 - \underline{x}_1) + \alpha_2^i(x_2 - \underline{x}_2) + \cdots + \alpha_n^i(x_n - \underline{x}_n) \quad (3.14)$$

now, similar to the case in 1 dimension it is necessary to introduce m binary variables, one for every interval, $\{y_1, \ldots, y_m\}$ so that we can ensure that a point x is in the right sub-box B_i.

$$-(1 - y_i)M + \underline{x}_k^i \leq x_k \leq \overline{x}_k^i + (1 - y_i)M, \qquad \forall i \in \{1, \ldots, m\} \quad \forall k \in \{1, \ldots, n\} \quad (3.15)$$

where

$$\sum_{i=1}^{m} y_i = 1, \quad (3.16)$$

$$y_i \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, m\} \quad (3.17)$$

so, there is only one binary variable for which $y_i = 1$ holds, such that only one sub-box B_i is activated resulting in $\underline{x}_k^i \leq x_k \leq \overline{x}_k^i$ ($\forall k \in \{1, \ldots n\}$).

Now, to approximate the non-linear function, we introduce a new artificial variable z. And, just like before, it is necessary to ensure that the constraint takes on the correct function value through one of the following constraints.

$$-(1 - y_i)M \leq z - f_i(x_1, x_2, \ldots, x_n) \leq (1 - y_i)M, \qquad \forall i \in \{1, \ldots, m\} \quad z = b_i \quad (3.18a)$$

$$z - f_i(x_1, x_2, \ldots, x_n) \leq (1 - y_i)M, \qquad \forall i \in \{1, \ldots, m\} \quad z \geq b_i \quad (3.18b)$$

$$-(1 - y_i)M \leq z - f_i(x_1, x_2, \ldots, x_n), \qquad \forall i \in \{1, \ldots, m\} \quad z \leq b_i \quad (3.18c)$$

Here, eq. (3.18a) is used if the original constraint is an equality constraint; eq. (3.18b) represents a greater than or equal to constraint; and eq. (3.18c) represents a less than or equal to constraint.

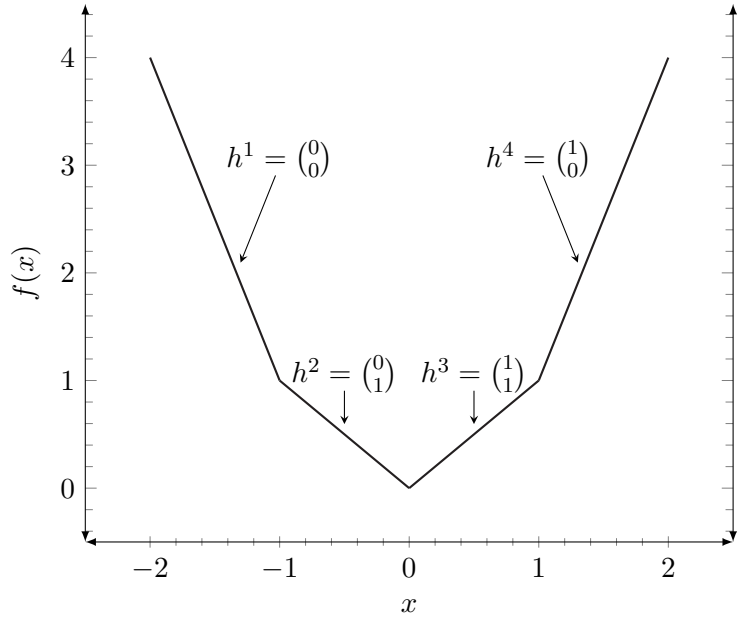


Figure 3.3: Piecewise linear approximation of $f(x) = x^2$ using $\lceil \log_2(m) \rceil$ binary variables.

3.4 Reducing the Number of Binary Variables

To reduce the number of binary variables from linear to binary logarithmic, the approach outlined in Vielma (2015) will be followed. Approaches for univariate functions and multivariate functions are virtually the same. As explained in the previous sections, the traditional Big-M reformulation technique requires the addition of m binary variables. However, it is possible to avoid using m binary variables since already $\lceil \log_2(m) \rceil$ variables are sufficient to represent m distinct possibilities.

Example 3.4.1. *In Figure 3.3 the piecewise linear approximation of* $f(x) = x^2$ *is represented on the domain* $[-2, 2]$*. The original non-linear function is clearly broken up into four distinct linear pieces* ($f_1(x), f_2(x), f_3(x),$ *and* $f_4(x)$)*, which indicates that it should be possible to use* $\lceil \log_2(4) \rceil = 2$ *binary variables to express this situation.*

This is indicated by the four h-vectors. Each h-vector is comprised out of two binary parameters ($h^i = \binom{h^i_1}{h^i_2}$)*. These can be used to indicate if a constraint should be enforced or not, which is explained hereafter.*

More formal, for $i = 1, \ldots, m$ we choose vectors $h^i \in \{0,1\}^{\lceil \log_2(m) \rceil}$ such that $h^i \neq h^j$ for $i \neq j$. Then, we can reformulate the earlier constraints from Section 3.2 by replacing all occurrences of $(1 - y_i)M$ with

$$\left(\sum_{j=1}^{\lceil \log_2(m) \rceil} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j \right) M, \qquad \forall i \in \{1, \ldots, m\} \tag{3.19}$$

and by formulating the binary variables as

$$y \in \{0,1\}^{\lceil \log_2(m) \rceil} \tag{3.20}$$

Furthermore, it is obvious that the constraint $\sum_{i=1}^{m} y_i = 1$ from the original formulation needs to be removed. Potentially leading to a more balanced branch-and-bound tree.

Hence, when some $h^i = y$ the part between parentheses will reduce to 0 and the constraint is enforced. But when $h^i \neq y$ the expression on the right-hand-side becomes large, depending on the value of M and the constraint is not enforced.

In this thesis, vectors h^i are formatted according to Gray code. Gray code is a way to format numbers using a binary encoding scheme. Gray code works in such a way that only one bit in the group changes from the number before and after it, see Table 3.1 for an example. The main benefit of Gray code over standard binary encoding is when a reformulation technique like the CC model is used, where only two adjacent λ's are allowed to be nonzero. Thus, regular binary encoding could as well be used for the Big-M approach. A more detailed discussion on Gray code in combination with the CC model is given in Wei (2020).

Table 3.1: Differences between decimal, binary and Gray code.

Decimal	Binary	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Chapter 4

Results

The results chapter is structured as follows. Firstly, Section 4.1 will describe how the models are evaluated. Secondly, Sections 4.2 to 4.4 will describe the test problem for the 1-dimensional case, 2-dimensional case, and 4-dimensional case, respectively. For all sections, the model using a linear number of binary variables is shown first and the model using a logarithmic number of binary variables is given thereafter. Thirdly, computational results will be given before lastly moving into a discussion on the obtained results.

4.1 Evaluation Procedures

In all evaluations six different solvers are compared. The release version and the used licenses can be found in Table 4.1. All solvers were installed in separate virtual environments with Python version 3.8.13 as the programming language and Pyomo version 6.4.2 used as the modelling language.

Table 4.1: Setup for Computational Study

Solver	Release Version	License	Source
Cplex	22.1	Academic	IBM® (n.d.)
Gurobi	9.5.2	Academic WLS	Gurobi Optimization, LLC (n.d.)
Xpress	8.14.2	Community License	Fair Isaac Corporation (n.d.)
Glpk	5.0	GNU General Public License 3	GNU (n.d.)
Scip	8.0.2	ZIB Academic License	Zuse Institute Berlin (ZIB) (n.d.)
Cbc	2.10.8	Eclipse Public License 2	Forrest and Lougee-Heimer (n.d.)

The reformulated MILPs were solved on an Apple MacBook Air running on macOS Monterey version 12.6.2, with a 1.6 GHz Dual-Core Intel Core i5 processor with 8 GB of 1,600 MHz DDR3 memory. Solvers were not restricted in the number of available threads. So, potentially up to 4 threads could be used.

The 4-dimensional test problem was taken from the MINLPLib library, a library of Mixed-Integer and Continuous Non-Linear Programming Instances (Bussieck et al., 2003). The problem can be found under the name 'chance'.

To obtain comparative solving times for every solver, the models were solved using the following optional Pyomo command: `report_timing=True`.

4.2 The 1-Dimensional Problem

4.2.1 Linear number of Binary Variables

For the 1-dimensional case, the following simple test problem was formulated. Note that this is a convex problem and binary variables are in principle not needed. However, in this thesis, this problem is used to see how the Big-M approach with binary variables works.

$$\begin{aligned} -\min_{x} \quad & -x \\ \text{s.t.} \quad & x^2 \leq 2, \\ & x \geq -1, \\ & x \in [-2, 2] \end{aligned} \tag{4.1}$$

From eq. (4.1) we can see that there is one non-linear constraint, namely $x^2 \leq 2$. The Big-M approach is used to generate a piecewise linear approximation over four intervals (and thus five breakpoints). This results in the following:

$$a_0 = -2, \quad a_1 = -1, \quad a_2 = 0, \quad a_3 = 1, \quad a_4 = 2$$

Now, for every sub-interval i the slope α_i and the intercept β_i can be calculated from eqs. (3.4) and (3.5). And then the piecewise linear function $\widehat{f}$ is

$$\widehat{f}(x) = \begin{cases} f_1(x) = -3x - 2, & \text{for } x \in [-2, -1] \\ f_2(x) = -x, & \text{for } x \in [-1, 0] \\ f_3(x) = x, & \text{for } x \in [0, 1] \\ f_4(x) = 3x - 2, & \text{for } x \in [1, 2] \end{cases}$$

Then, according to eqs. (3.7) to (3.9) we get the following

$$
\begin{aligned}
-(1-y_1)M + a_0 \leq x \leq a_1 + (1-y_1)M,\\
-(1-y_2)M + a_1 \leq x \leq a_2 + (1-y_2)M,\\
-(1-y_3)M + a_2 \leq x \leq a_3 + (1-y_3)M,\\
-(1-y_4)M + a_3 \leq x \leq a_4 + (1-y_4)M
\end{aligned}
$$

where

$$
\begin{aligned}
y_1 + y_2 + y_3 + y_4 = 1,\\
y_1, y_2, y_3, y_4 \in \{0, 1\}
\end{aligned}
$$

The non-linear constraint is of the less than or equal to form with $b_i = 2, \quad \forall i \in \{1, 2, 3, 4\}$, thus using eq. (3.10b) gives

$$
\begin{aligned}
f_1(x) \leq 2 + (1-y_1)M,\\
f_2(x) \leq 2 + (1-y_2)M,\\
f_3(x) \leq 2 + (1-y_3)M,\\
f_4(x) \leq 2 + (1-y_4)M
\end{aligned}
$$

Finally, the implicit formulation of the 1-dimensional reformulated problem then looks like

$$
\begin{aligned}
-\min_x \quad & -x && && && (4.2a)\\
\text{s.t.} \quad & x \geq -1, && && && (4.2b)\\
& f_i(x) = \alpha_i x + \beta_i, && x \in [a_{i-1}, a_i], && \forall i \in \{1,2,3,4\}, && (4.2c)\\
& -(1-y_i)M + a_{i-1} \leq x, && && \forall i \in \{1,2,3,4\}, && (4.2d)\\
& x \leq a_i + (1-y_i)M, && && \forall i \in \{1,2,3,4\}, && (4.2e)\\
& f_i(x) \leq b_i + (1-y_i)M, && && \forall i \in \{1,2,3,4\}, && (4.2f)\\
& \sum_{i=1}^{4} y_i = 1, && && && (4.2g)\\
& y_i \in \{0,1\}, && && \forall i \in \{1,2,3,4\} && (4.2h)
\end{aligned}
$$

A graphical representation of eq. (4.2) is given in Figure 4.1. There, the black lines represent the piecewise linear approximation, the red line is the original non-linear function, the green lines represent the maximum value of the constraints. Then the area shaded in blue represents the feasible

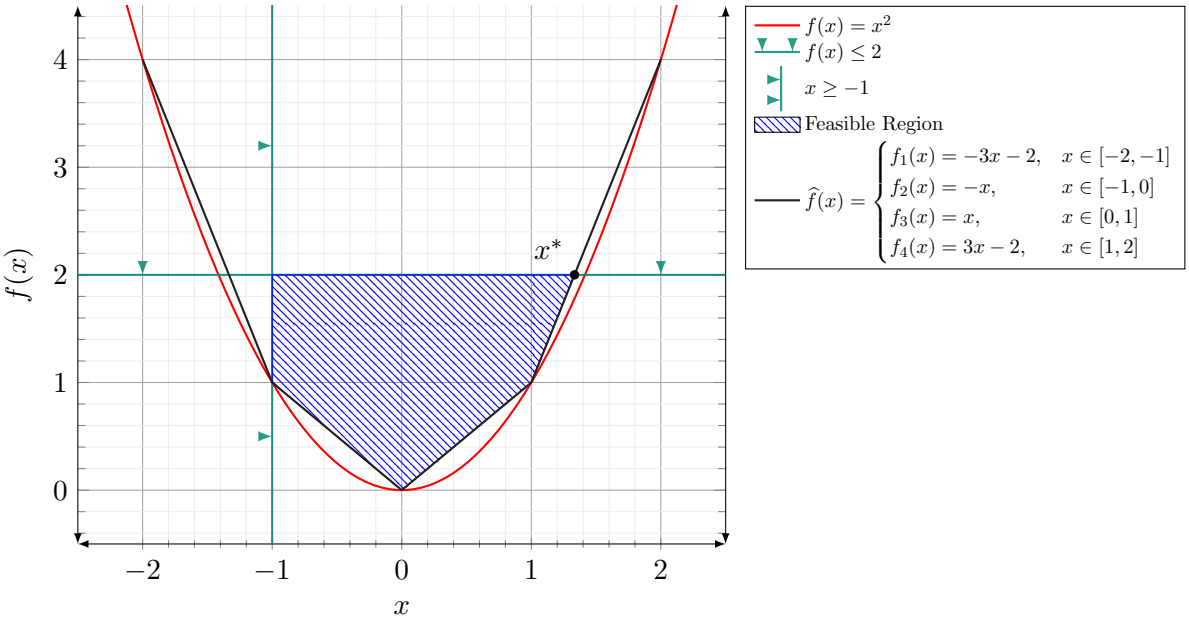


Figure 4.1: Graphical representation of the 1-dimensional test case.

region of the approximation and the point x^* is the optimal solution for this problem.

Figure 4.1 shows that the optimal solution for the approximated problem is lower than the actual solution for the original non-linear problem. According to Rebennack (2016) the piecewise-linear approximation in Figure 4.1 is the tightest (in the sense of minimizing the maximal distance) overestimator with the maximal absolute function value difference at the mid-point of each interval.

4.2.2 Logarithmic Number of Binary Variables

In order to reduce the number of binary variables from linear to binary logarithmic the approach outlined in Section 3.4 on page 14 is followed. Applying eqs. (3.19) and (3.20) to eq. (4.2) results in the following MILP model

$$-\min_{x} \quad -x \tag{4.3a}$$

$$\text{s.t.} \quad x \geq -1, \tag{4.3b}$$

$$f_i(x) = \alpha_i x + \beta_i, \quad x \in [a_{i-1}, a_i], \quad \forall i \in \{1,2,3,4\}, \tag{4.3c}$$

$$-\left(\sum_{j=1}^{2} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j\right) M + a_{i-1} \leq x, \quad \forall i \in \{1,2,3,4\}, \tag{4.3d}$$

$$x \leq a_i + \left(\sum_{j=1}^{2} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j\right) M, \quad \forall i \in \{1,2,3,4\}, \tag{4.3e}$$

$$f_i(x) \leq b_i + \left(\sum_{j=1}^{2} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j\right) M, \quad \forall i \in \{1,2,3,4\}, \tag{4.3f}$$

$$y_j \in \{0,1\}, \quad \forall j \in \{1,2\} \tag{4.3g}$$

4.3 The 2-Dimensional Problem

4.3.1 Linear Number of Binary Variables

The 2-dimensional problem is a simplification of the 'chance' problem used in the 4-dimensional case. The 2-dimensional problem is formulated as follows:

Consider a simple feed-mix problem with the data given in Table 4.2. The goal is to determine a mix with minimal cost per ton satisfying the protein requirement.

Table 4.2: Data for 2-dimensional feed-mix problem.

	Barley	Oats	Requirement
Protein content (%)	12.0	11.9	11.3
Protein variance ($\%^2$)	0.28	0.19	-
Cost per ton ($)	24.55	26.75	-

Let a_k (k=1,2) denote the protein content. Let b denote the specified minimum protein level. Let c_k (k=1,2) denote the cost per ton of respectively

barley (k=1) and oats (k=2). Furthermore, let x_k (k=1,2) be the fraction of the feed-mix composed of each of the raw materials.

In the deterministic case, the protein constraint can be written as

$$f(x) = \sum_{k=1}^{2} a_k x_k \geq b \tag{4.4}$$

However, the situation above does not hold when the protein content of the raw materials is constant for a particular batch, but is subject to variation for different batches. If that is the case, we are dealing with a probabilistic case and the problem becomes so-called chance constrained. If we require that the probability of achieving a certain protein content (equal to or larger) in the feed-mix should not be lower than a given percentage, then constraint eq. (4.4) can be rewritten as

$$P[f(x) \geq b] \geq 1 - \epsilon \tag{4.5}$$

where $1-\epsilon$ stands for the required minimum probability. Hence, ϵ represents the proportion of cases for which a protein content less than b is allowed to occur.

Assuming that the distribution of protein content for respectively barley and oats is normal and independent. In this case $f(x)$ from eq. (4.4) represents the expected protein content, but in the probabilistic case we need to account for the variances, denoted by $\sigma^2(\widetilde{a}_k)$. Then, the expected protein content of any given feed-mix, eq. (4.5), takes the following form

$$\sum_{k=1}^{2} a_k x_k + \varphi \sqrt{\sum_{k=1}^{2} \sigma^2(\widetilde{a}_k) x_k^2} \geq b \tag{4.6}$$

where φ represents the normal deviate of the required probability, which can be calculated or looked up in statistical tables. If the required probability, $1 - \epsilon$, is 95%, we have $\varphi = -1.645$. A detailed proof of why eq. (4.6) is a valid formulation can be found in Bracken and McCormick (1968, p. 95-97). Applying this to our feed-mix problem, assuming a normal distribution of protein content with means and variances equal to the values in Table 4.2.

Now, it is possible to write this as a chance-constrained NLP model.

$$\min_{x} \quad \sum_{k=1}^{2} c_k x_k \tag{4.7a}$$

$$\text{s.t.} \quad \sum_{k=1}^{2} a_k x_k + \varphi\sqrt{\sum_{k=1}^{2} \sigma^2(\widetilde{a}_k) x_k^2} \geq b, \tag{4.7b}$$

$$\sum_{k=1}^{2} x_k = 1, \tag{4.7c}$$

$$x_k \geq 0, \quad k = 1, 2 \tag{4.7d}$$

The following example shows the piecewise linear approximation approach of a 2-dimensional non-linear equation over a single sub-box.

Example 4.3.1. *We consider the piecewise linear approximation of eq. (4.7b) over sub-box* $B_1 = [0, 0.1] \times [0, 0.1]$*. So, we want to calculate* $f_1(x_1, x_2)$*. Explicitly writing out the left-hand-side of eq. (4.7b) gives*

$$f(x_1, x_2) = a_1 x_1 + a_2 x_2 + \varphi\sqrt{\sigma^2(\widetilde{a}_1) x_1^2 + \sigma^2(\widetilde{a}_2) x_2^2}$$

Next we want to calculate the slope according to eq. (3.13). The slope in the direction of x_1 *is calculated as follows.*

$$\begin{aligned} f(\overline{x}_1^1, \underline{x}_2^1) &= 12(0.1) + 11.9(0) - 1.645\sqrt{0.28(0.1)^2 + 0.19(0)^2} \\ &\approx 1.1130 \\ f(\underline{x}_1^1, \underline{x}_2^1) &= 12(0) + 11.9(0) - 1.645\sqrt{0.28(0)^2 + 0.19(0)^2} \\ &= 0 \end{aligned}$$

Filling in eq. (3.13) results in

$$\alpha_1^1 = \frac{f(\overline{x}_1^1, \underline{x}_2^1) - f(\underline{x}_1^1, \underline{x}_2^1)}{\overline{x}_1^1 - \underline{x}_1^1} \approx \frac{1.1130 - 0}{0.1 - 0} \approx 11.130$$

Performing the same steps for the slope in the direction of x_2 *yields* $\alpha_2^1 \approx 11.183$*.*

Now, all the required information to approximate f *over* B_1 *is available so eq. (3.14) can be filled in:*

$$\begin{aligned} f_1(x_1, x_2) &= f(\underline{x}_1^1, \underline{x}_2^1) + \alpha_1^1(x_1 - \underline{x}_1) + \alpha_2^1(x_2 - \underline{x}_2) \\ &\approx 0 + 11.130(x_1 - 0) + 11.183(x_2 - 0) \\ &\approx 11.130x_1 + 11.183x_2 \end{aligned}$$

Hence, eq. (4.7b) over B_1 is approximated by $f_1(x_1, x_2) \approx 11.130x_1 + 11.83x_2$.

Repeating the steps in the example above for every sub-box B_i gives the necessary results to reformulate eq. (4.7). Following the subsequent steps from the methodology outlined by eqs. (3.15) to (3.18) results in the following MILP model.

$$\min_x \quad \sum_{k=1}^{2} c_k x_k \tag{4.8a}$$

$$\text{s.t.} \quad z - f_i(x_1, x_2) \leq (1 - y_i)M \qquad \forall i \in \{1, \dots, m\}, \tag{4.8b}$$

$$z \geq b, \tag{4.8c}$$

$$-(1 - y_i)M + \underline{x}_k^i \leq x_k \qquad \forall i \in \{1, \dots, m\}, \; k = 1, 2, \tag{4.8d}$$

$$x_k \leq \overline{x}_k^i + (1 - y_i)M \qquad \forall i \in \{1, \dots, m\}, \; k = 1, 2, \tag{4.8e}$$

$$\sum_{i=1}^{m} y_i = 1, \tag{4.8f}$$

$$y_i \in \{0, 1\}, \qquad \forall i \in \{1, \dots, m\}, \tag{4.8g}$$

$$\sum_{k=1}^{2} x_k = 1, \tag{4.8h}$$

$$x_k \geq 0, \qquad k = 1, 2 \tag{4.8i}$$

In the formulation above we can see that eq. (4.7b) is replaced by eqs. (4.8b) and (4.8c) to guarantee that the constraint takes on the right function value. To ensure that x_k is in the right sub-box eqs. (4.8d) and (4.8e) are used.

4.3.2 Logarithmic Number of Binary Variables

To convert MILP problem eq. (4.8) to the logarithmic reformulation we apply eqs. (3.19) and (3.20) and we remove eq. (4.8f) resulting in the following MILP model

$$\min_{x} \quad \sum_{k=1}^{2} c_k x_k$$

s.t.

$$z - f_i(x_1, x_2) \leq \left(\sum_{j=1}^{\lceil \log_2(m) \rceil} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j \right) M \quad \forall i \in \{1, \ldots, m\},$$

$$z \geq b,$$

$$- \left(\sum_{j=1}^{\lceil \log_2(m) \rceil} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j \right) M + \underline{x}_k^i \leq x_k \qquad \forall i \in \{1, \ldots, m\},$$

$$k = 1, 2,$$

$$x_k \leq \overline{x}_k^i + \left(\sum_{j=1}^{\lceil \log_2(m) \rceil} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j \right) M \qquad \forall i \in \{1, \ldots, m\},$$

$$k = 1, 2,$$

$$y_j \in \{0, 1\}, \qquad \forall j \in \{1, \ldots, \lceil \log_2(m) \rceil\},$$

$$\sum_{k=1}^{2} x_k = 1,$$

$$x_k \geq 0, \qquad k = 1, 2$$

4.4 The 4-Dimensional Problem

4.4.1 Linear Number of Binary Variables

The 4-dimensional test problem was first published in van de Panne and Popp (1963). Since the 2-dimensional problem is a reduced version of the 4-dimensional one, this is a feed-mix problem as well. Table 4.3 shows the protein content, protein variance, fat content, and cost per ton for the raw materials. For the protein content and fat content requirements are given as well. The goal is to generate a feed-mix satisfying the required contents of respectively protein and fat while minimizing the costs. It is assumed that the protein content is chance-constrained while the fat content is deterministic.

Formulating the NLP model is similar to the process for 2-dimensions. Let $k = 1, 2, 3, 4$ denote the set of raw ingredients (1 = barley, 2 = oats, 3 = sesame flakes, 4 = groundnut meal). Let a_{1k} represent the protein content

Table 4.3: Data for 4-dimensional feed-mix problem.

	Barley	Oats	Sesame Flakes	Groundnut meal	Requirement
Protein content (%)	12.0	11.9	41.8	52.1	21.0
Protein variance ($\%^2$)	0.28	0.19	20.5	0.62	-
Fat content (%)	2.3	5.6	11.1	1.3	5.0
Cost per ton ($)	24.55	26.75	39.00	40.50	-

for each of the raw materials per ton in weight percentage, and let a_{2k} then denote the fat content per ton in weight percentage. Protein variance is denoted by $\sigma^2(\widetilde{a}_{1k})$. Then, b_1 denotes the minimum protein content of the feed-mix whereas b_2 denotes the minimum fat content. Now, c_k denotes the cost per ton of each raw material. Finally, let x_k be the fraction of the feed-mix composed of each of the raw materials. Then, the NLP model is formulated as

$$\min_{x} \quad \sum_{k=1}^{4} c_k x_k \tag{4.9a}$$

$$\text{s.t.} \quad \sum_{k=1}^{4} a_{1k} x_k + \varphi \sqrt{\sum_{k=1}^{4} \sigma^2(\widetilde{a}_{1k}) x_k^2} \geq b_1, \tag{4.9b}$$

$$\sum_{k=1}^{4} a_{2k} x_k \geq b_2, \tag{4.9c}$$

$$\sum_{k=1}^{4} x_k = 1, \tag{4.9d}$$

$$x_k \geq 0, \quad k = 1, 2, 3, 4 \tag{4.9e}$$

Now, the Big-M reformulation can be applied, analogously to the 2-dimensional case. Resulting in the following MILP

$$\min_{x} \quad \sum_{k=1}^{4} c_k x_k \tag{4.10a}$$

$$\text{s.t.} \quad z - f_i(x_1, x_2, x_3, x_4) \leq (1 - y_i)M \qquad \forall i \in \{1, \ldots, m\}, \tag{4.10b}$$

$$z \geq b_1, \tag{4.10c}$$

$$-(1 - y_i)M + \underline{x}_k^i \leq x_k \qquad \forall i \in \{1, \ldots, m\}, \quad k = 1, 2, 3, 4, \tag{4.10d}$$

$$x_k \leq \overline{x}_k^i + (1 - y_i)M \qquad \forall i \in \{1, \ldots, m\}, \quad k = 1, 2, 3, 4, \tag{4.10e}$$

$$\sum_{i=1}^{m} y_i = 1, \tag{4.10f}$$

$$y_i \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, m\}, \tag{4.10g}$$

$$\sum_{k=1}^{4} a_{2k} x_k \geq b_2, \tag{4.10h}$$

$$\sum_{k=1}^{4} x_k = 1, \tag{4.10i}$$

$$x_k \geq 0, \qquad k = 1, 2, 3, 4 \tag{4.10j}$$

4.4.2 Logarithmic Number of Binary Variables

Analogously to the 2-dimensional model, the 4-dimensional one can be reformulated using a logarithmic number of binary variables. The result is shown below.

$$\min_{x} \quad \sum_{k=1}^{4} c_k x_k$$

s.t.

$$z - f_i(x_1, x_2, x_3, x_4) \leq \left(\sum_{j=1}^{\lceil \log_2(m) \rceil} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j \right) M \quad \forall i \in \{1, \ldots, m\},$$

$$z \geq b_1,$$

$$- \left(\sum_{j=1}^{\lceil \log_2(m) \rceil} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j \right) M + \underline{x}_k^i \leq x_k \quad \forall i \in \{1, \ldots, m\}, \; k = 1, 2, 3, 4,$$

$$x_k \leq \overline{x}_k^i + \left(\sum_{j=1}^{\lceil \log_2(m) \rceil} h_j^i - \sum_{j:h_j^i=1} y_j + \sum_{j:h_j^i=0} y_j \right) M \quad \forall i \in \{1, \ldots, m\}, \; k = 1, 2, 3, 4,$$

$$y_j \in \{0, 1\}, \quad \forall j \in \{1, \ldots, \lceil \log_2(m) \rceil\},$$

$$\sum_{k=1}^{4} a_{2k} x_k \geq b_2,$$

$$\sum_{k=1}^{4} x_k = 1,$$

$$x_k \geq 0, \quad k = 1, 2, 3, 4$$

4.5 Computational Results

An overview of the problems that were solved are given in Table 4.4. The first column shows the problem name, for instance '1D-Lin' stands for '1-dimensional linear number of binary variables' and '1D-Log' stands for '1-dimensional logarithmic number of binary variables' et cetera. Furthermore, for the 2-dimensional problem we have situations 'A' and 'B', referring to the same initial problem, but with differences in the number of binary variables.

Objective function values and values for the optimal point are vertically centered where applicable, since changing the model from linear to logarithmic does not have any effect on the function values. 'Obj (exact)' represents the objective function value for the original non-linear problem and 'Obj (approx)' illustrates the objective function value for the reformulated Big-M

problem. Furthermore, the values corresponding to the optimal point are based on the approximated objective function values; not the exact ones.

Note that the number of binary variables for the 1-dimensional problem are increased from 4 to 1,000 in the linear situation and hence from 2 to 10 in the logarithmic case, compared to Section 4.2. This is done to obtain results where a time difference between the various solvers can be recognized, since additional breakpoints increase computational difficulty.

Table 4.4: Overview of Optimization Problems

Problem	# Binary variables	Obj (exact)	Obj (approx)	x_1	x_2	x_3	x_4
1D-Lin	1,000	1.41	1.41	1.41	-	-	-
1D-Log	10				-	-	-
2D-Lin-A	256	25.11	25.05	0.77	0.23	-	-
2D-Log-A	8					-	-
2D-Lin-B	1,024	25.11	25.04	0.78	0.22	-	-
2D-Log-B	10					-	-
4D-Lin	4,096	29.89	29.86	0.64	0.00	0.31	0.05
4D-Log	12						

In Table 4.4 it can be seen that all approximations closely approximate the exact objective value of the NLPs. For the 1-dimensional problem, the approximation has the same objective value as the original problem. For the other problems, the approximated objective values are slightly lower than those of the exact NLPs.

Table 4.5 shows the solving times for every used solver per problem in seconds.

Table 4.5: Solving times for the tested problems in seconds.

Problem	Solver					
	GLPK	CPLEX	GUROBI	XPRESS	SCIP	CBC
1D-Lin	24.68	0.38	0.29	2.09	8.57	2.77
1D-Log	1.17	3.98	1.45	4.94	3.46	11.97
2D-Lin-A	1.17	0.51	0.25	0.26	0.41	1.14
2D-Log-A	0.31	0.95	0.63	0.73	0.65	2.10
2D-Lin-B	69.87	1.56	1.08	-*	0.53	3.31
2D-Log-B	6.71	9.21	4.77	-*	8.48	22.81
4D-Lin	14,160.10	20.66	23.28	-*	57.48	1,448.81
4D-Log	236.23	43.76	45.48	-*	330.65	1,167.82

* community license does not allow more than 5,000 rows and columns.

In Table 4.5 it can be seen that when a linear number of binary variables are used in the 1-dimensional case GLPK is by far the slowest solver with 24.68 seconds, followed by SCIP at 8.57 seconds. Both XPRESS and CBC perform

in the same order of magnitude at 2.09 and 2.77 seconds, respectively. The fastest solvers for this case are Cplex and Gurobi at respectively 0.38 and 0.29 seconds. When the number of binary variables are reduced, both Glpk and Scip are faster than before. However, the logarithmic approach slows down the other solvers. Figure 4.2a graphically shows the computational results for the 1-dimensional case.

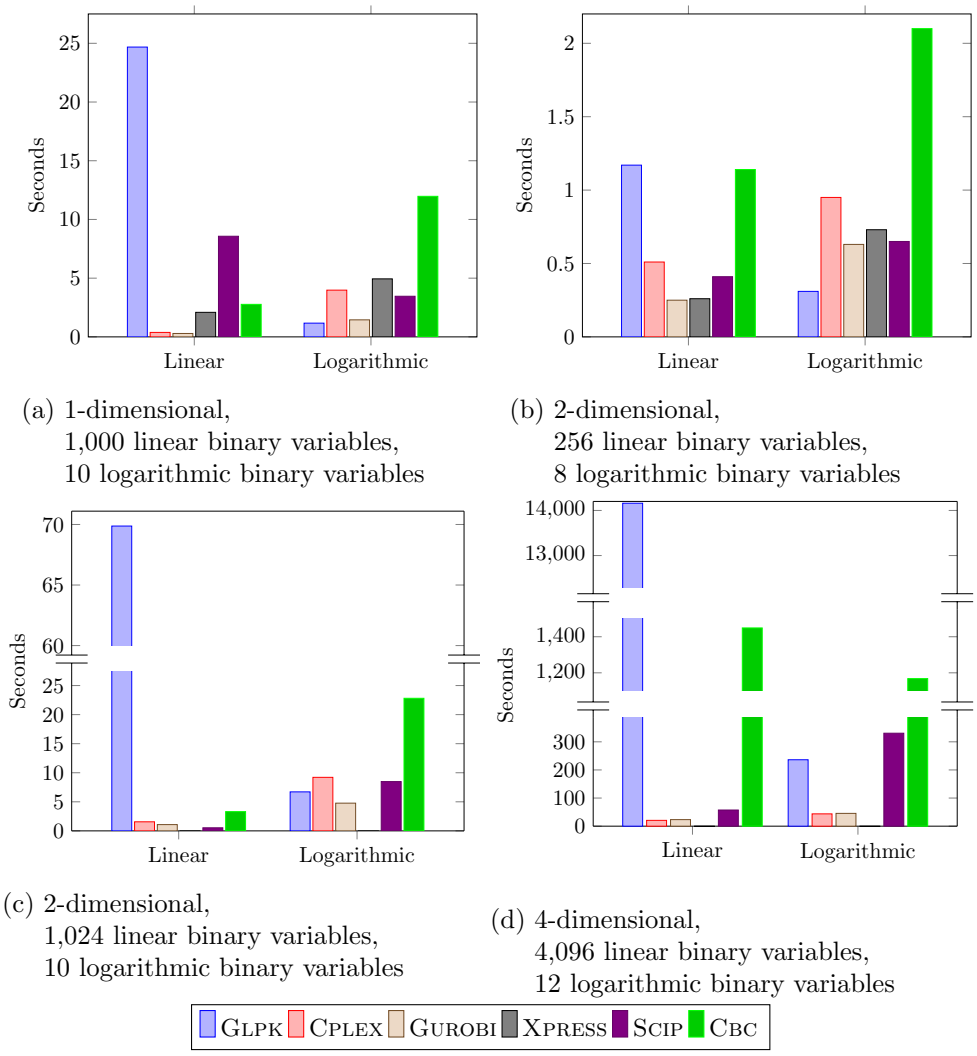
For the 2-dimensional cases, all solvers were able to solve the smaller linear case in less than 2 seconds (2D-Lin-A). Gurobi and Xpress are the fastest solvers in this case with solving times of 0.25 and 0.26 seconds, respectively. What is remarkable is that, when the number of binary variables are reduced, all solvers except for Glpk show slower solving times. The solving time for Glpk is reduced to 0.31 seconds in the logarithmic situation this is comparable to the solving times of Gurobi and Xpress, some of the fastest commercial solvers and even faster than Cplex. Hence, the approach shows very promising results for the Glpk solver. Although, when the number of binary variables are increased, for instance in 2D-Lin-B, Glpk still performs better with the logarithmic formulation but is outperformed by every tested solver using the linear formulation.

Furthermore, Scip is only faster using the logarithmic formulation for the 1-dimensional problem. For the other problems, using Scip yields slower computational results with the logarithmic formulation when compared to the linear formulation.

In the case of the 4-dimensional problem using a linear number of binary variables it can be seen that Glpk is significantly slower than every other tested solver. Glpk took 3 hours and 56 minutes to solve this problem, compared to just over 20 seconds for the fastest solver of this problem, Cplex. Cbc shows faster computational results for the logarithmic 4-dimensional problem, although the logarithmic computational results for Cbc show that it is consistently slower than the other solvers, including Glpk. Another observation is that Glpk was able to solve the logarithmic formulation a factor 60 quicker than the linear formulation.

So, to summarise, only Glpk acts as expected from the literature review. Cbc showed a computational reduction for the largest tested problem which can be seen as a promising result but requires further investigation.

A graphical representation of the results for these problems are given in Figure 4.2. Note that, for Figures 4.2c and 4.2d there are breaks in the y-axis.



(a) 1-dimensional,
1,000 linear binary variables,
10 logarithmic binary variables

(b) 2-dimensional,
256 linear binary variables,
8 logarithmic binary variables

(c) 2-dimensional,
1,024 linear binary variables,
10 logarithmic binary variables

(d) 4-dimensional,
4,096 linear binary variables,
12 logarithmic binary variables

Figure 4.2: Computational results for linear and logarithmic reformulations.

4.6 Discussion

Big-M reformulations are not necessarily *locally ideal* or *sharp*. In Vielma et al. (2010a) it is mentioned that the convex combination model, which is sharp but not locally ideal and has one of the worst computational performances of the methods studied in Vielma et al. (2010b), was able to outperform the formulations of Li et al. (2009), one of which is based on a Big-M reformulation. Furthermore, the reformulations of Li et al. (2009) were significantly outperformed by each of the six formulations tested in Vielma et al. (2010b). Therefore, in general, Big-M reformulations may be relatively weak compared to other known reformulations. And thus, solvers

like Gurobi, Cplex and Xpress may already make use of reformulation techniques when a problem with a linear number of binary variables is being solved, but this cannot be stated with absolute certainty.

In Chapter 2 it was hypothesized that reducing the number of binary variables should be able to lead to reduced computational difficulty. The results have shown that for the Glpk solver this has consistently been the case. However, for the other solvers, this cannot be concluded based on this thesis. This may be due to, as mentioned above, the relatively weak Big-M reformulation. Another reason may be that the reformulation only reduces the number of binary variables and not the number of constraints as well.

The results on the Glpk solver are promising to users of this solver. Since, out of all tested solvers, only Glpk and Cbc are truly open source with a public license (and presumably less sophisticated than the commercial solvers) and the logarithmic Big-M reformulation with Glpk outperformed the linear and logarithmic reformulations when the Cbc solver was used in every instance except for the 2-dimensional B case. So, for users that do not have access to expensive state-of-the-art solvers, the results are useful. Moreover, the Big-M reformulation is relatively easy to implement compared to most other reformulation techniques.

As mentioned in Chapter 2, according to Hu et al. (2013) and Till et al. (2004) reducing constraints and binary variables have the greatest impact on computational efficiency. The current Big-M reformulation only reduces binary variables, not constraints. A method which reduces constraints and binary variables (e.g., logarithmic convex combination model (Vielma & Nemhauser, 2011)) may provide different results in computational efforts for different solvers. Maybe such a method can be used to reduce solving times for state-of-the-art solvers like Gurobi, Cplex, and Xpress.

Another reason why Gurobi, Cplex and Xpress may perform better on larger problems is that they are capable of using multithreading when solving the branch-and-bound tree, in contrast to Glpk and Scip. Cbc is able to use multithreading as well, but this has to be specified by the user.

Chapter 5

Conclusions, limitations and future research

RQ1, was concerned with the suitable piecewise linearization techniques. In Chapter 2 we have seen that the convex combination, incremental cost, Big-M, Log convex combination and Log Big-M are among the available suitable piecewise linearization techniques. The results indicate that the Big-M reformulation can be used to reasonably well approximate the original NLP. It should however be noted that all tested problems in this thesis were relatively small. Therefore, tests on larger real-life problems should be performed in order to confirm this.

RQ2 was about reducing binary variables. In the literature review, it was stated that logarithmic reformulation techniques are capable of reducing the number of binary variables. An additional benefit was that logarithmic techniques do not require the sum of binary variables to add up to one, tending to lead to more balanced branch-and-bound trees. The resulted models prove that the logarithmic Big-M reformulation indeed successfully reduces the number of binary variables when compared to a linear Big-M reformulation.

The main objective of this thesis was to find an answer to RQ3. *"What is the effect of reducing the number of binary variables on solving mixed-integer optimization problems with different solvers?"*. We can state, that for the Big-M reformulation with a linear and logarithmic number of binary variables the results indicate that for the GLPK solver this is a suitable technique to reduce solving times. However, for the other tested solvers XPRESS, GUROBI, CPLEX, CBC, and SCIP this is not necessarily the case. CPLEX, GUROBI and XPRESS did not show improved computational times. Although, it should be noted that with XPRESS it was not possible to solve models with more than 5,000 rows and columns, since the author only had

access to the community license. Results for the Cbc and Scip solver are inconclusive, Cbc showed reduced computational time in the 4-dimensional problem, but tests on larger problems should be performed in order to determine if the Log Big-M approach is suitable for Cbc. Scip only showed a reduction in computational time when solving the 1-dimensional problem.

For modelers that currently do not have access to an expensive state-of-the-art solver and that want to solve NLP problems using a MILP solver, the logarithmic Big-M reformulation technique in combination with the Glpk solver can be used to potentially reduce solving times compared to the linear Big-M reformulation used in this study.

Further research could be focussed on reformulation techniques which are *sharp* and/or *locally ideal*. For instance, the effect of the convex combination and the logarithmic convex combination model on solver efficiency could be tested. These formulations are generally stronger than the Big-M reformulation used in this study, and therefore it is assumed that they are computationally superior. Furthermore, the test problems that were used in this study were relatively small compared to most real-life models. Tests with larger models should be done to confirm that the logarithmic Big-M reformulation technique produces computationally better results than the linear Big-M reformulation technique for the Glpk solver and possibly the Cbc and Scip solvers.

Bibliography

Abidi, H., de Leeuw, S., & Klumpp, M. (2014). Humanitarian supply chain performance management: a systematic literature review. *Supply Chain Management: An International Journal*, *19*, 592–608. https://doi.org/10.1108/SCM-09-2013-0349

Adams, W. P., & Henry, S. M. (2012). Base-2 Expansions for Linearizing Products of Functions of Discrete Variables. *Operations Research*, *60*(6), 1477–1490. https://doi.org/10.1287/opre.1120.1106

Alkhalifa, L., & Mittelmann, H. (2022). New Algorithm to Solve Mixed Integer Quadratically Constrained Quadratic Programming Problems Using Piecewise Linear Approximation. *Mathematics*, *10*(2). https://doi.org/10.3390/math10020198

Asghari, M., Fathollahi-Fard, A. M., Al-E-Hashem, S. M. M., & Dulebenets, M. A. (2022). Transformation and Linearization Techniques in Optimization: A State-of-the-Art Survey. *Mathematics*, *10*. https://doi.org/10.3390/math10020283

Bazaraa, M., Sherali, H., & Shetty, C. (1993). *Nonlinear Programming Theory and Algorithms*. John Wiley.

Bellman, R. E. (1961). *Adaptive Control Processes: A Guided Tour* [First introduction of curse of dimensionality].

Bertsimas, D., Dunn, J., & Wang, Y. (2021). Near-optimal Nonlinear Regression Trees. *Oper. Res. Lett.*, *49*(2), 201–206. https://doi.org/10.1016/j.orl.2021.01.002

Bracken, J., & McCormick, G. (1968). *Selected Applications of Nonlinear Programming*. Wiley. https://apps.dtic.mil/sti/pdfs/AD0679037.pdf

Bussieck, M. R., Drud, A. S., & Meeraus, A. (2003). MINLPLib–A Collection of Test Models for Mixed-Integer Nonlinear Programming. *INFORMS J. on Computing*, *15*(1), 114–119. https://doi.org/10.1287/ijoc.15.1.114.15159

Carrion, M., & Arroyo, J. (2006). A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, *21*(3), 1371–1378. https://doi.org/10.1109/TPWRS.2006.876672

Contardo, C., & Ngueveu, S. U. (2021). *On the approximation of separable non-convex optimization programs to an arbitrary numerical precision* [preprint]. https://hal.science/hal-03336022

Croxton, K. L., Gendron, B., & Magnanti, T. L. (2003). A Comparison of Mixed-Integer Programming Models for Nonconvex Piecewise Linear Cost Minimization Problems. *Management Science*, *49*(9), 1268–1273. Retrieved March 1, 2023, from http://www.jstor.org/stable/4134040

Dantzig, G. B. (1960). On the Significance of Solving Linear Programming Problems with Some Integer Variables. *Econometrica*, *28*(1), 30–44. Retrieved March 1, 2023, from http://www.jstor.org/stable/1905292

Fair Isaac Corporation. (n.d.). *FICO® Xpress Optimization*. Retrieved September 29, 2022, from https://www.fico.com/en/products/fico-xpress-optimization

Fard, A. M. F., Woodward, L., & Akhrif, O. (2021). Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept. *J. Ind. Inf. Integr.*, *24*, 100233. https://doi.org/10.1016/j.jii.2021.100233

Feijoo, B., & Meyer, R. R. (1988). Piecewise-Linear Approximation Methods for Nonseparable Convex Optimization. *Management Science*, *34*(3), 411–419. Retrieved February 24, 2023, from http://www.jstor.org/stable/2632053

Forrest, J., & Lougee-Heimer, R. (n.d.). *CBC User Guide*. Retrieved October 26, 2022, from https://www.coin-or.org/Cbc/

Fourer, R., Gray, D. M., & Kernighan, B. W. (1993). *AMPL : A Modeling Language for Mathematical Programming* (1. ed.). The Scientific Press.

Geißler, B., Martin, A., Morsi, A., & Schewe, L. (2012). Using Piecewise Linear Functions for Solving MINLPs. https://doi.org/10.1007/978-1-4614-1927-3_10

Giménez, C., & Tachizawa, E. (2012). Extending sustainability to suppliers: A systematic literature review. *Supply Chain Management: An International Journal*, *17*. https://doi.org/10.1108/13598541211258591

GNU. (n.d.). *GLPK (GNU Linear Programming Kit)*. Retrieved September 29, 2022, from https://www.gnu.org/software/glpk

Guignard-Spielberg, M., & Spielberg, K. (2005). Preface. *Ann. Oper. Res.*, *139*(1), 17–20. https://doi.org/10.1007/s10479-005-3441-2

Gurobi Optimization, LLC. (n.d.). *Gurobi Optimizer*. Retrieved September 29, 2022, from https://www.gurobi.com/products/gurobi-optimizer

Harzing, A. (2007). *Publish or Perish* (Version 8). https://harzing.com/resources/publish-or-perish

Hillier, F. S., & Lieberman, G. J. (1986). *Introduction to Operations Research* (4th ed.). Holden-Day, Inc.

Hu, Y.-C., Lin, M.-H., Carlsson, J. G., Ge, D., Shi, J., & Tsai, J.-F. (2013). A Review of Piecewise Linearization Methods. *Mathematical Problems in Engineering*, *2013*, 101376. https://doi.org/10.1155/2013/101376

Ibaraki, T. (1976). Integer programming formulation of combinatorial optimization problems. *Discret. Math.*, *16*(1), 39–52. https://doi.org/10.1016/0012-365X(76)90091-1

IBM®. (n.d.). *IBM CPLEX Optimizer*. Retrieved September 29, 2022, from https://www.ibm.com/nl-en/analytics/cplex-optimizer

Jeroslow, R. G., & Lowe, J. K. (1984). Modelling with integer variables. In B. Korte & K. Ritter (Eds.), *Mathematical programming at oberwolfach ii* (pp. 167–184). Springer Berlin Heidelberg. https://doi.org/10.1007/BFb0121015

Jiang, R., Shen, S., & Zhang, Y. (2017). Integer Programming Approaches for Appointment Scheduling with Random No-Shows and Service Durations. *Oper. Res.*, *65*(6), 1638–1656. https://doi.org/10.1287/opre.2017.1656

Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., & Wolsey, L. A. (Eds.). (2010). *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. Springer. https://doi.org/10.1007/978-3-540-68279-0

Keha, A. B., de Farias, I. R., & Nemhauser, G. L. (2004). Models for representing piecewise linear cost functions. *Oper. Res. Lett.*, *32*(1), 44–48. https://doi.org/https://doi.org/10.1016/S0167-6377(03)00059-2

Kong, L., & Maravelias, C. T. (2020). On the Derivation of Continuous Piecewise Linear Approximating Functions. *INFORMS J. Comput.*, *32*(3), 531–546. https://doi.org/10.1287/ijoc.2019.0949

Lee, J., & Wilson, D. (2001). Polyhedral methods for piecewise-linear functions I: the lambda method. *Discrete Applied Mathematics*, *108*(3), 269–285. https://doi.org/https://doi.org/10.1016/S0166-218X(00)00216-X

Li, H.-L., Huang, Y.-H., & Fang, S.-C. (2013). A Logarithmic Method for Reducing Binary Variables and Inequality Constraints in Solving Task Assignment Problems. *INFORMS Journal on Computing*, *25*(4), 643–653. https://doi.org/10.1287/ijoc.1120.0527

Li, H.-L., Huang, Y.-H., & Fang, S.-C. (2017). Linear Reformulation of Polynomial Discrete Programming for Fast Computation. *INFORMS Journal on Computing*, *29*(1), 108–122. https://doi.org/10.1287/ijoc.2016.0716

Li, H., Lu, H., Huang, C., & Hu, N. (2009). A Superior Representation Method for Piecewise Linear Functions. *INFORMS J. Comput.*, *21*(2), 314–321. https://doi.org/10.1287/ijoc.1080.0294

Markowitz, H. M., & Manne, A. S. (1957). On the Solution of Discrete Programming Problems. *Econometrica*, *25*(1), 84–110. Retrieved March 1, 2023, from http://www.jstor.org/stable/1907744

Martin, A., Möller, M., & Moritz, S. (2006). Mixed Integer Models for the Stationary Case of Gas Network Optimization. *Mathematical Programming*, *105*(2), 563–582.

Meyer, R. (1976). Mixed integer minimization models for piecewise-linear functions of a single variable. *Discrete Mathematics*, *16*(2), 163–171. https://doi.org/https://doi.org/10.1016/0012-365X(76)90145-X

Misener, R., & Floudas, C. (2010). Piecewise-Linear Approximations of Multidimensional Functions. *Journal of Optimization Theory and Applications*, *145*, 120–147. https://doi.org/10.1007/s10957-009-9626-0

Owen, J. H., & Mehrotra, S. (2002). On the Value of Binary Expansions for General Mixed-Integer Linear Programs. *Oper. Res.*, *50*(5), 810–819. https://doi.org/10.1287/opre.50.5.810.370

Padberg, M. (2000). Approximating separable nonlinear functions via mixed zero-one programs. *Oper. Res. Lett.*, *27*(1), 1–5. https://doi.org/https://doi.org/10.1016/S0167-6377(00)00028-6

Rebennack, S. (2016). Computing tight bounds via piecewise linear functions through the example of circle cutting problems. *Math. Methods Oper. Res.*, *84*(1), 3–57. https://doi.org/10.1007/s00186-016-0546-0

Rebennack, S., & Kallrath, J. (2015). Continuous Piecewise Linear Delta-Approximations for Bivariate and Multivariate Functions. *J. Optim. Theory Appl.*, *167*(1), 102–117. https://doi.org/10.1007/s10957-014-0688-2

Sherali, H. D. (2001). On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. *Oper. Res. Lett.*, *28*(4), 155–160. https://doi.org/10.1016/S0167-6377(01)00063-3

Sridhar, S., Linderoth, J. T., & Luedtke, J. R. (2013). Locally ideal formulations for piecewise linear functions with indicator variables. *Oper. Res. Lett.*, *41*(6), 627–632. https://doi.org/10.1016/j.orl.2013.08.010

Taha, H. (2003). *Operations Research: An Introduction* (7th ed.). Pearson Education. https://wur.on.worldcat.org/oclc/899030725

Till, J., Engell, S., Panek, S., & Stursberg, O. (2004). Applied hybrid system optimization: An empirical investigation of complexity [Analysis and Design of Hybrid Systems]. *Control Engineering Practice*, *12*(10), 1291–1303. https://doi.org/10.1016/j.conengprac.2004.04.003

Tranfield, D., Denyer, D., & Smart, P. (2003). Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review. *British Journal of Management*, *14*, 207–222. https://doi.org/10.1111/1467-8551.00375

Trespalacios, F., & Grossmann, I. E. (2015). Improved Big-M reformulation for generalized disjunctive programs. *Comput. Chem. Eng.*, *76*, 98–103. https://doi.org/10.1016/j.compchemeng.2015.02.013

Ugaz, C., Han, L., & Lim, A. (2019). Knot Locating in Piecewise Linear Approximation. *arXiv: Optimization and Control*.

van de Panne, C., & Popp, W. (1963). Minimum-Cost Cattle Feed under Probabilistic Protein Constraints. *Management Science*, *9*(3), 405–430. Retrieved March 18, 2023, from http://www.jstor.org/stable/2627114

Vielma, J. P. (2015). Mixed Integer Linear Programming Formulation Techniques. *SIAM Rev.*, *57*(1), 3–57. https://doi.org/10.1137/130915303

Vielma, J. P., Ahmed, S., & Nemhauser, G. L. (2010a). A Note on "A Superior Representation Method for Piecewise Linear Functions". *INFORMS J. Comput.*, *22*(3), 493–497. https://doi.org/10.1287/ijoc.1100.0379

Vielma, J. P., Ahmed, S., & Nemhauser, G. L. (2010b). Mixed-Integer Models for Nonseparable Piecewise-Linear Optimization: Unifying Framework and Extensions. *Operations Research*, *58*(2), 303–315. https://doi.org/10.1287/opre.1090.0721

Vielma, J. P., & Nemhauser, G. L. (2011). Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Math. Program.*, *128*(1-2), 49–72. https://doi.org/10.1007/s10107-009-0295-4

Watters, L. J. (1967). Letter to the Editor - Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems. *Oper. Res.*, *15*(6), 1171–1174. https://doi.org/10.1287/opre.15.6.1171

Wei, W. (2020). Tutorials on Advanced Optimization Methods. *arXiv: Optimization and Control*. https://arxiv.org/pdf/2007.13545.pdf

Yildiz, S., & Vielma, J. P. (2013). Incremental and encoding formulations for Mixed Integer Programming. *Oper. Res. Lett.*, *41*(6), 654–658. https://doi.org/10.1016/j.orl.2013.09.004

Zhao, Z., Cheng, C., & Yan, L. (2021). An efficient and accurate Mixed-integer linear programming model for long-term operations of large-scale hydropower systems. *IET Renewable Power Generation*, *15*(6), 1178–1190. https://doi.org/https://doi.org/10.1049/rpg2.12098

Zuse Institute Berlin (ZIB). (n.d.). *SCIP Optimization Suite*. Retrieved October 26, 2022, from https://www.scipopt.org/

Appendix A

Systematic Literature Review

A.1 Approach to the Literature Review

A.1.1 Planning

This literature review will focus on finding answers for RQ1 as well as leading to hypotheses for RQ2 and RQ3 proposed in Chapter 1.

A.1.2 Searching

Based on the RQs several key terms were identified, 'piecewise linear approximations', 'piecewise linear functions', 'logarithmic reformulation', 'nonlinear programming' and 'mixed-integer linear programming'. Furthermore, the boolean operators AND, OR, and NOT were used to focus the search.

Where applicable snowballing is included, snowballing is the use of different keywords, not given above, to find additional literature. The time period of publications included has been set to 1976 to the present, the year 1976 has been chosen since that is when the logarithmic Big-M reformulation was first introduced (Ibaraki, 1976). The search results are shown in Table A.1.

The publications found on Google Scholar were exported to a Microsoft Excel file using Publish or Perish (Harzing, 2007). Publications found in respectively the Scopus and Web of Science databases were exported using integrated tools from their respective webpages.

Table A.1: Protocol for database search

Database	Scope	Date of search	Number of publications
Scopus	Title, abstract, and keyword	26/01/2023	27
Google Scholar	Anywhere in the article	26/01/2023	802
Web of Science	Title, abstract, and keyword	26/01/2023	175
Total			**1,004**

A.1.3 Screening

To objectively determine if publications should be included or excluded the following criteria were determined:

- Inclusion aspects: only peer reviewed articles, only articles related to (non)linear programming. Moreover, 5 book chapters were included because these were often cited in the relevant articles.
- Exclusion aspects: non-English articles, conference proceedings, editorial opinions, articles focused on specific applications (e.g., assignment problems, gas network optimization, supply chain planning, et cetera), case studies, and articles focussing on signomial functions.

The searching phase resulted in 1,004 articles that included the defined key terms. The next step in the process was screening the articles, see Figure A.1. After this duplicates were removed. Then, based on the exclusion and inclusion criteria articles not fitting the scope of this research were removed. For the remaining 100 articles, the abstract was read to see if the articles really met the inclusion criteria. This resulted in 46 publications which met all the inclusion and exclusion aspects.

A.1.4 Extraction, synthesis, and Reporting

The remaining publications were categorized according to publication period, methodology, and research contribution. This is summarized in Table A.2.

A.2 Results from the Literature Review

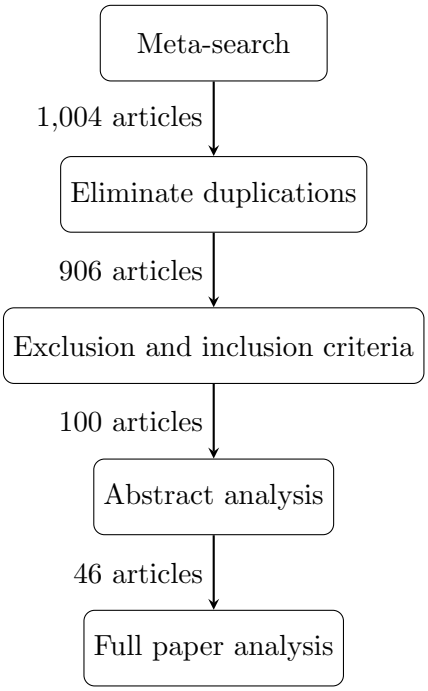


Figure A.1: Paper screening methodology (modified from Abidi et al., 2014; Giménez & Tachizawa, 2012).

Table A.2: Summary statistics (based on Abidi et al., 2014).

	All Journals
Number of articles	46
Author nationalities	
Europe	33
USA	28
Other	46
Publication period	
1976 - 1990	3
1990 - 2000	3
2000 - 2010	13
2010 - 2023	33
Methodology	
Review	2
Survey	1
Mathematical programming	46
Research Contribution	
Application	3
Model	3
Theory	42
Benchmark library	1