Investigation of k-rationality & cutting plane methods in sequentially connected 2D assignment problems

An exploration of properties of the TGOSPA metric

Master's thesis in Engineering mathematics and computational science

Albert Vesterlund

DEPARTMENT OF MATHEMATICAL SCIENCES

Chalmers University of Technology
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Investigation of k-rationality & cutting plane methods in sequentially connected 2D assignment problems

An exploration of properties of the TGOSPA metric

Albert Vesterlund

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Investigation of k-rationality & cutting plane methods in sequentially connected 2D assignment problems
An exploration of properties of the TGOSPA metric
Albert Vesterlund

Supervisor: Ann-Brith Strömberg, Department of Mathematical Sciences
Examiner: Axel Ringh, Department of Mathematical Sciences

Master's Thesis 2025
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Investigation of k-rationality & cutting plane methods in sequentially connected 2D assignment problems
An exploration of properties of the TGOSPA metric
ALBERT VESTERLUND
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Integer linear optimization problems are generally difficult to solve efficiently due to the discrete nature of their solution spaces. A common approach to address these difficulties is to relax the integrality constraints, solve a continuous optimization problem, and utilize different methods to recover near-optimal integer solutions'.

This thesis explores the usage of such approaches in the context of the Time-extended Generalised Optimal Sub-Pattern Assignment (TGOSPA) metric—a performance measure for the evaluation of multiple object tracking algorithms. Specifically, this can be seen as examining a binary sequentially connected 2D assignment problem where the binary requirement has been relaxed. Two primary methods for recovering binary optimal solutions are considered; one based on the concept of k-rationality and the other on using different cutting plane methods derived from fractional Gomory cuts. The results suggests that k-rationality is numerically unsuitable for this specific problem, and that the Gomory-based cutting methods lack performance consistency.

Finally, an outline of several different directions for future research, based on the observations made in this thesis, is presented.

Keywords: Optimization, Sequential assignment problem, Mixed Binary Linear Program, LP-relaxation, TGOSPA, k-rational, Cutting plane methods, Gomory cuts

Acknowledgements

First of all, I would like to thank my supervisor Ann-Brith Strömberg for the great amount of support and ideas during my thesis. The conversations and discussions had during the last few months has lead to results and discoveries I would not have seen otherwise. Without your help this thesis would simply not have been possible. Finally, I would also like to thank all the friends and family who have listened to my ramblings about this thesis and given me their love and support throughout the hardships and setbacks I faced along the way.

Albert Vesterlund, Gothenburg, June 2025

Contents

1

Introduction

In mathematical optimization, there exists multiple different types of problems that can be solved. Notably, problems where the solution is required to be integer are generally significantly more difficult to solve. As such, it is often desired to explore different methods to solve the integer-valued optimization problem in some more efficient manner. A common way of doing this is by removing the requirement of integrality of the solution, allowing continuous values for the variables, and then introducing additional constraints or penalties on the objective value as a measure to try and still obtain an optimal integer solution. A common type of problem often solved this way is assignment problems, where each of a number of agents is to be assigned to do some specific task at a cost, with the goal to minimize the overall cost of completing all tasks. Clearly, any relevant solution to, this problem is integer-valued, however, it is often the case that these types of problems can be solved without the integer requirement, and still have that an optimal solution is always integer. This property is commonly referred to as the problem possesses the integrality property.

One problem which can be compared to an assignment problem is the Generalized Optimal Sub-Pattern Assignment (GOSPA) metric [1], further expanded in [2] to the so called *Time-extended Generalised Optimal Sub-Pattern Assignment* (TGOSPA) metric, supporting time-dependent ground truths and estimates. These metrics are used to compare estimation algorithms by using their estimations of targets and comparing them with the ground truth targets. In short, they assign some produced estimated target to some existing ground truth target, yielding a "cost" of the distance between them. The main difference which makes this problem deviate from a normal assignment problem, is the connection of variables between time steps. As such, the problem does not necessarily possess this nice property that optimal solutions without integrability requirement on variables are always integer valued.

In [3] it was proposed that the extended GOSPA metric, despite showing that the problem is not totally unimodular, still have the property that any (extreme point) optimal solution is always integer valued, regardless of requiring it by constraints. This was later shown to be false in [4], which provided a small example of a configuration where a non-integer solution achieved a better objective value than the best

integer-valued solution.

The goal of this thesis is to explore methods to try and find ways that can be used to solve the problem with removed integrality constraints, while still obtaining an integer solution in the end. The methods explored utilize the properties of k-rationality and k-modularity, and also different versions of more common cutting plane methods. Some relevant theory is presented in Chapter 2, giving details about general optimization, the TGOSPA metric, integrality properties and cutting plane methods. In Chapter 3, the usage of k-rationality is explored analytically and numerically before some brief results on the bounds of k are presented. Then, in Chapter 4, the investigation of efficiency of cutting plane methods is presented together with some comments on the implementational details of such methods. Finally, Chapter 5, presents the conclusions from this thesis together with some proposals for future work.

2

Prerequisite Theory

This chapter contains some brief prerequisite theory required for the general understanding of the work presented. First some general optimization theory is covered, before the outline and properties of the specific problem at hand are described. Finally, the chapter covers the basic theory behind the cutting plane methods examined.

2.1 Mixed-Integer Linear Programs

Given a variable vector $x \in \mathbb{R}^n \times \mathbb{Z}^m$, a constraint matrix $A \in \mathbb{R}^{r\times(n+m)}$, a cost vector $c \in \mathbb{R}^{(n+m)}$, and a constraint vector $b \in \mathbb{R}^r$, a *Mixed-Integer Linear Program* (MILP) is an optimization problem which can be written on the form

$$\text{minimize } c^\intercal x \tag{2.1a}$$
$$\text{subject to } Ax \leq b \tag{2.1b}$$
$$x \in \mathbb{R}^n \times \mathbb{Z}^m. \tag{2.1c}$$

A special case of this program is when all integer variables are instead required to take binary values, that is $x \in \mathbb{R}^n \times \{0,1\}^m$. In that case, the problem is usually referred to as a *Mixed-Binary Linear Program* (MBLP).

Consider the problem (2.1) on its *standard form*, where inequalities have been replaced by equalities, using a vector of slack variables $s \in \mathbb{R}^r : s \geq 0$ such that

$$Ax \leq b \iff (A, I)\begin{pmatrix} x \\ s \end{pmatrix} = b, \tag{2.2}$$

where I denotes the $r \times r$ identity matrix. Since the slack variables are required to be non-negative, the standard form is an equivalent representation. For any linear program, using its standard form and some given optimal solution x^*, it is possible to separate the problem into so-called *basis* and *non-basis* matrices and variables.

For simplicity, and without loss of generality, consider a simple optimization problem on standard form with $A \in \mathbb{R}^{r\times n},\ b \in \mathbb{R}^r,\ x \in \mathbb{R}^n,\ n > r$. Then, the matrix A and variables x can be separated such that

$$Ax = Bx_B + Nx_N = b \tag{2.3}$$

where $B \in \mathbb{R}^{r\times r}$ and $N \in \mathbb{R}^{r\times(n-r)}$ is said to be the *basis* and *non-basis* matrices, respectively. Similarly, $x_B \in \mathbb{R}^r$, $x_N \in \mathbb{R}^{(n-r)}$ are called the basic and non-basic variables. Then, for any non-degenerate basic feasible solution, it holds that $(x_B)_i \neq 0, i = 1, \ldots, r$, and $x_N = 0^{n-r}$. Further, for any linear program with an optimal solution, it holds that there exists a basic feasible solution which is optimal. And in fact, since the basis matrix B is non-singular by design, the optimal basic feasible solution x^* can be described by the parts x_B^*, x_N^* from

$$\begin{aligned} Ax^* =& Bx_B^* + Nx_N^* = b \Rightarrow && (2.4)\\ \Rightarrow\quad & x_B^* + B^{-1}Nx_N = B^{-1}b && (2.5)\\ \Rightarrow\quad & x_B^* = B^{-1}b,\ x_N^* = 0. && (2.6) \end{aligned}$$

2.2 Linear programming relaxation

A *Linear programming relaxation* is the optimization problem where the integrality/binary constraint of a MILP/MBLP has been relaxed. As an example, consider the relaxation of a MBLP. The problem could then be stated as

$$\begin{aligned} z_{\text{LP}}^* = \min_x\ & c^\intercal x && && (2.7a)\\ \text{s.t. } & Ax \leq b && && (2.7b)\\ & 0 \leq x_i \leq 1, && \forall\ i \in \{n+1, \ldots, n+m\} && (2.7c)\\ & x \in \mathbb{R}^{n+m} && && (2.7d) \end{aligned}$$

where $0 \leq x_i \leq 1, \forall\ i \in \{n+1, \ldots, n+m\}$ is the relaxed version of the constraint $x_{\{n+1,\ldots,n+m\}} \in \{0,1\}^m$.

A consequence of relaxing the binary constraints is that the optimal value of the relaxed problem becomes, in general, better than the objective value of the MBLP. That is, $z_{\text{LP}}^* \leq z_{\text{MBLP}}^*$. Additionally, given that the polyhedron defined by $Ax \leq b$ and the constraint (2.7c) is bounded an non-empty, an optimal solution to the relaxed problem (and generally any real-valued LP) lies in an extreme point of the polyhedron. As such, unless the feasible set of solutions to (2.7) is integral, the optimal solution x_{LP}^* is not guaranteed to be binary for the m binary variables in the MBLP.

The reason why one might be interested in performing an LP-relaxation is that while integer programs are generally $\mathcal{NP}$-hard, the relaxed programs can be solved in polynomial time. As a result, solving a problem with real-valued variables, and employing additional tactics to obtain integer solutions, is often much more computationally efficient than solving problems requiring integer-valued variables.

2.3 The TGOSPA metric

A *Multiple-object tracking algorithm* is an algorithm used to create an estimate of multiple objects trajectories over time, i.e., a set of estimates denoted $\mathbf{Y}$, based on

some data, describing the set of ground truth trajectories $\mathbf{X}$. The goal of any such algorithm is then to estimate this ground truth as accurately as possible, without containing errors. As such, it is desired to have some metric to compare algorithms in order to evaluate performance.

Consider some maximum amount of discrete time steps, $t_{\max} > 1$, $t_{\max} \in \mathbb{Z}_+$ and let $T = \{1, \dots, t_{\max}\}$, $n_{\mathbf{X}} := |\mathbf{X}|, n_{\mathbf{Y}} := |\mathbf{Y}|$. For ease of notation, let also $N_{\mathbf{X}} := \{1, \dots, n_{\mathbf{X}}\}$ and $N_{\mathbf{Y}} := \{1, \dots, n_{\mathbf{Y}}\}$. Finally, let $W_{\mathbf{X},\mathbf{Y}}$ be the set of all binary matrices W^t of dimension $N_{\mathbf{XY}} := (n_{\mathbf{X}} + 1) \times (n_{\mathbf{Y}} + 1)$ such that each $w_{ij}^t \in W^t$ satisfies

$$\sum_{i=1}^{n_{\mathbf{X}}+1} w_{ij}^t = 1, \qquad j \in N_{\mathbf{Y}}, \tag{2.8a}$$

$$\sum_{j=1}^{n_{\mathbf{Y}}+1} w_{ij}^t = 1, \qquad i \in N_{\mathbf{X}}, \tag{2.8b}$$

$$w_{n_{\mathbf{X}}+1, n_{\mathbf{Y}}+1}^t = 0, \tag{2.8c}$$

$$w_{ij}^t \in \{0, 1\}, \qquad (ij) \in N_{\mathbf{XY}}. \tag{2.8d}$$

Note that, for each time step t, W^t is essentially the formulation of a $N_{\mathbf{XY}}$ dimensioned assignment problem. As such, these constraints will be related to as *assignment-constraints*.

The *Binary Linear Programming* formulation of the *Time-extended Generalised Optimal Sub-Pattern Assignment* (TGOSPA) problem is then given by the following definition.

Definition 2.1. [2, Def. 1, Lemma 1] Given some metric $d_b(\cdot, \cdot) : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}_+$, scalars $c > 0$, and $1 \leq p < \infty$, let

$$d_p^c(\mathbf{x}, \mathbf{y}) = \begin{cases} \min(c, d_b(x, y)), & \mathbf{x} = \{x\}, \mathbf{y} = \{y\}, \\ 0, & \mathbf{x} = \mathbf{y} = \varnothing, \\ \frac{c}{2^{1/p}}, & \text{otherwise}, \end{cases}$$

and

$$D_{ij}^t = d_p^c(\mathbf{x}_i^t, \mathbf{y}_j^t)^p.$$

Then, the TGOSPA metric (for $\alpha = 2$) between sets $\mathbf{X}$ and $\mathbf{Y}$ is defined as

$$d_p^{(c,\gamma)}(\mathbf{x}, \mathbf{y}) = \min_{w_{ij}^t \in W_{\mathbf{X},\mathbf{Y}}} \left(\sum_{t=1}^{t_{\max}} \sum_{i=1}^{n_{\mathbf{X}}+1} \sum_{j=1}^{n_{\mathbf{Y}}+1} D_{ij}^t w_{ij}^t + \frac{\gamma^p}{2} \sum_{t=1}^{t_{\max}-1} \sum_{i=1}^{n_{\mathbf{X}}} \sum_{j=1}^{n_{\mathbf{Y}}} \left| w_{ij}^t - w_{ij}^{t+1} \right| \right)^{\frac{1}{p}}. \tag{2.9}$$

As per [3, Model (1.7)], letting $\hat{T} = T \setminus \{t_{\max}\}$, the metric can be rewritten as the MBLP

$$d_p^{(c,\gamma)}(\mathbf{x}, \mathbf{y}) = \min_{w_{ij}^t \in W_{\mathbf{X},\mathbf{Y}}} \left(\sum_{t \in T} \sum_{i=1}^{n_{\mathbf{X}}+1} \sum_{j=1}^{n_{\mathbf{Y}}+1} D_{ij}^t w_{ij}^t + \frac{\gamma^p}{2} \sum_{t \in \hat{T}} \sum_{i=1}^{n_{\mathbf{X}}} \sum_{j=1}^{n_{\mathbf{Y}}} h_{ij}^t \right)^{\frac{1}{p}} \tag{2.10a}$$

with the additional constraints on $w_{ij}^t \in W^t$ for $t \in \hat{T}$

$$h_{ij}^t \geq w_{ij}^t - w_{ij}^{t+1} \qquad (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}} \tag{2.10b}$$
$$h_{ij}^t \geq w_{ij}^{t+1} - w_{ij}^t \qquad (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}} \tag{2.10c}$$
$$h_{ij}^t \in \mathbb{R} \qquad (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}}. \tag{2.10d}$$

These additional constraints will be related to as the TGOSPA-constraints. By design, h_{ij}^t is a free variable and behaves as $\left|w_{ij}^t - w_{ij}^{t+1}\right|$ in regard to the minimization context. However, in practice it can only attain the values $\{0,1\}$ in any extreme point of the polyhedron defining the feasible set due to the imposed constraints.

Further, the metric can be expressed using inequalities and non-negative variables as follows. By introducing $g_{ij}^t := 1 - h_{ij}^t$ and

$$\widetilde{D}_{ij}^t := D_{ij}^t - D_{i,n_{\mathbf{Y}}+1}^t - D_{n_{\mathbf{X}}+1,j}^t \ , \ (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}}$$

the inequality version of the TGOSPA metric can be formulated as

$$z_{\text{TGOSPA}} := d_p^{(c,\gamma)}(\mathbf{x},\mathbf{y}) = \min \left(\sum_{t=1}^{T} \sum_{i=1}^{n_{\mathbf{X}}} \sum_{j=1}^{n_{\mathbf{Y}}} \widetilde{D}_{ij}^t w_{ij}^t + \frac{\gamma^p}{2} \sum_{t=1}^{T-1} \sum_{i=1}^{n_{\mathbf{X}}} \sum_{j=1}^{n_{\mathbf{Y}}} (1 - g_{ij}^t) - \right. \tag{2.11a}$$
$$\left. \sum_{t=1}^{T} \sum_{i=1}^{n_{\mathbf{X}}} D_{i,n_{\mathbf{Y}}+1}^t - \sum_{t=1}^{T} \sum_{j=1}^{n_{\mathbf{Y}}} D_{n_{\mathbf{X}}+1,j}^t \right)^{\frac{1}{p}}, \tag{2.11b}$$

$$\text{s.t} \qquad \sum_{i=1}^{n_{\mathbf{X}}} w_{ij}^t \leq 1, \qquad t \in T, \ j \in N_{\mathbf{Y}}, \tag{2.11c}$$
$$\sum_{j=1}^{n_{\mathbf{Y}}} w_{ij}^t \leq 1, \qquad t \in T, \ i \in N_{\mathbf{X}}, \tag{2.11d}$$
$$w_{ij}^t - w_{ij}^{t+1} + g_{ij}^t \leq 1, \qquad t \in \hat{T}, \ (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}}, \tag{2.11e}$$
$$-w_{ij}^t + w_{ij}^{t+1} + g_{ij}^t \leq 1, \qquad t \in \hat{T}, \ (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}}, \tag{2.11f}$$
$$w_{ij}^t \geq 0, \qquad t \in T, \ (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}}, \tag{2.11g}$$
$$g_{ij}^t \geq 0, \qquad t \in T, \ (i,j) \in N_{\mathbf{X}} \times N_{\mathbf{Y}}. \tag{2.11h}$$

While the variables w_{ij}^t and g_{ij}^t are clearly multidimensional, this does not hinder the usage of any methods requiring the regular optimization structure of $Ax \leq b$. By simply "flattening" these variables, ordering them one after another in a single vector, these variables and their corresponding constraints can be represented on the standard form $Ax \leq b$ and elaborated with accordingly.

2.4 Integrality of solutions to LPs

As mentioned in Section 2.2, it is often desired to solve problems with real-valued variables instead of integer ones. However, since it's not always the case that the

relaxed problems yields an integer solution, it is of interest to identify some property leading to the optimal solution being integer, regardless of the lack of imposed integrality constraints. Two matrix properties related to this are *unimodularity* and *total unimodularity*.

Definition 2.2 (Unimodular matrix)**.** A matrix M is said to be *unimodular* if it is square and has determinant $+1$ or -1.

Definition 2.3 (Totally unimodular matrix)**.** A matrix $M \in \mathbb{R}^{m\times n}$ is said to be *totally unimodular* if, for every square sub-matrix $M' \in \mathbb{R}^{l\times l}$, $l \in \mathbb{Z}_+ : l \leq \min(m, n)$, it holds that $\det(M') \in \{-1, 0, 1\}$.

Now, two things can be noted about the totally unimodularity property. Firstly, a totally unimodular matrix, in comparison to a unimodular matrix, is not required to be square. Second, if some matrix M is said to be totally unimodular, it holds by definition that any sub-matrix (not necessarily square) is also totally unimodular.

For optimization purposes, the property of total unimodularity provides a way to guarantee integrality of solutions by simply observing the polyhedron defined by the problem constraints.

Theorem 2.1. [5, Thm. 4.4] Let A be an $m \times n$ integral matrix. The polyhedron $\{x : Ax \leq b, x \geq 0\}$ is integral for every $b \in \mathbb{Z}^m$ if and only if A is totally unimodular.

That is, if A is totally unimodular and b is integral, the polyhedron defined by the constraints $Ax \leq b, x \geq 0$ has only integer extreme points. Recalling the details about optimal solutions presented in Section 2.2, it is easy to see that regardless of the integrality requirement on x, if an optimal solution exists, there exists at least one integer optimal solution. As such, for a linear optimization problem with a totally unimodular constraint matrix, the integrality requirement can be relaxed without removing the integrality of an optimal solution.

However, for the TGOSPA metric this property is not applicable. While the constraint vector b is indeed integer, it does not hold that the constraint matrix A is totally unimodular. It has been shown in [3] that the matrix A of the TGOSPA model (2.11) has a simple case where a sub-matrix M' has a determinant $\det(M') \notin \{-1, 0, 1\}$. Namely, considering a sub-matrix containing parts from constraints related to the TGOSPA-constraints,

$$w_{ij}^t - w_{ij}^{t+1} + g_{ij}^t \leq 1, \tag{2.12a}$$

$$-w_{ij}^t + w_{ij}^{t+1} + g_{ij}^t \leq 1, \tag{2.12b}$$

it is possible to obtain a sub-matrix

$$M' = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{2.12c}$$

where the first column corresponds to w_{ij}^t and the second to g_{ij}^t. This matrix clearly

has the determinant 2, thus not fulfilling the requirement for total unimodularity of only having a $0, \pm 1$ determinant.

Instead, it might be possible to utilize two generalisations of the totally unimodularity property.

Definition 2.4. [6, Def. 2] A matrix A is said to be *k-rational* if for every non-singular square sub-matrix R of A, kR^{-1} is integral.

Definition 2.5. [6, Def. 4] A matrix A is *totally k-modular* if for all of its square submatrices R, it holds that $\det(R) \in \{0, \pm k^r, r \in \mathbb{N}\}$.

The connections between Definition 2.4 , Definition 2.5 and total unimodularity is further shown and explored in [6].

In [7], multiple connections between matrix structures and properties and the k-modular/k-rational properties are presented and examined. Specifically, it is proven that k-rationality has a similar desired property as total unimodularity.

Theorem 2.2. [7, Thm. 17] The polyhedron defined by $Ax \leq kb, x \geq 0$, where $k \in \mathbb{Z}_+$, is integral for each integral vector b, if and only if A is k-rational.

2.5 Cutting plane methods

A prominent method to find integer solutions to linear programs where some integrality constraint has been relaxed is to use so called *cutting plane methods*. A cutting plane method aims to reduce the feasible solution space by adding additional linear inequalities to cut away parts of the original solution space. As an example, consider the relaxed version of some ILP with a solution space defined by $Ax \leq b$. Let x^*_{LR} and x^*_{ILP} be the optimal solution to the relaxed and original integer problem, respectively. Using some cutting method it is possible to introduce additional constraints

$$Cx \leq d \tag{2.13a}$$

such that

$$Cx^*_{LR} > d, \tag{2.13b}$$
$$Cx^*_{ILP} \leq d \tag{2.13c}$$

are both valid. In other words, it is possible to introduce constraints which remove optimal solutions to the relaxed problem, without removing any integer solutions from the original problem. The result is a new solution space where the relaxed optimum is infeasible, yet the ILP optimum is unaffected. Exactly how C and d are defined depends on the cutting method used, however they all should abide by this property. These types of linear inequalities in (2.13a) are commonly referred to as *cuts*.

Below follows some brief theory behind the different types of cutting plane methods explored in this thesis.

2.5.1 Gomory's fractional cut

Recalling the possible separation of the constraint matrix A into basis and non-basis matrices B and N, respectively, such that

$$Ax = Bx_B + Nx_N = b, \tag{2.14}$$

Gomory's Fractional Cuts is a simple general-purpose cutting plane method based on using the basis matrix corresponding to some found optimal solution x^* to construct cuts. This type of cut was first presented in [8], but can be found in similar notation to this thesis in [5, Sec. 5.2.4].

Also recall that, for an optimal solution, and a corresponding basis B, it holds that

$$x^* = x_B + B^{-1}Nx_N = B^{-1}b. \tag{2.15}$$

Denote $\alpha_{ij} = (B^{-1}N)_{ij},\ \beta_i = (B^{-1}b)_i$ for $i \in I_B,\ j \in I_N$, where I_B is the set of indices corresponding to the optimal basis of the LP relaxation, and I_N the set of non-basis indices. If x^* is non-integer, there exists some $h \in I_B$ such that $\beta_h \notin \mathbb{Z}$. Then, relative to the h:th row of the basis matrix, the equation

$$x_h^* + \sum_{j \in I_N} \alpha_{hj} x_j^* = \beta_h \tag{2.16}$$

holds true since $x_j^* = 0,\ \forall\ j \in I_N$.

For ease of notation, let $f(a) := a - \lfloor a \rfloor \in [0, 1)$, denote the fractional part of any real valued integer a. Now, after some additional argumentation (see [5, Sec. 5.2.4]), assuming that there exists some $h \in I_B$ such that $f(\beta_h) > 0$, *Gomory's Fractional Cut* is given by the inequality

$$\sum_{j \in I_N} f(\alpha_{hj}) x_j \geq f(\beta_h). \tag{2.17}$$

A stronger cut

A stronger version of Gomory's fractional cut is presented in [9, Prop. 3]. Using the same notation as above, the strengthened cut is given by the inequality

$$\sum_{j \in I_N} \min \left\{ f(\alpha_{hj}), f(\beta_h) \frac{1 - f(\alpha_{hj})}{1 - f(\beta_h)} \right\} x_j \geq f(\beta_h). \tag{2.18}$$

By design, it is clear that this cut is at least as strong as Gomory's original fractional cut (2.17), with the potential to be stronger due to the added conditional coefficient. Further, it is a special case of the so called *Gomory's mixed-integer cut*.

2.5.2 Gomory's mixed-integer cut

Specifically for a MILP, it is possible to use a so called *Gomory mixed-integer cut* in order to cut the solution space [5, Sec. 5.3]. This cut is a generalization of the fractional cuts presented above.

Consider a set of indices $I_I \subseteq \{1, \ldots, n\}$ corresponding to the integer variables, where n is the number of variables, and a set of indices $I_C = \{1, \ldots, n\} \setminus I_I$ corresponding to the continuous variables.

Using the notation introduced above, if $I'_B := \{h \in I_B \cap I_I : f(\beta_h) > 0\}$ is non-empty, consider an index $h \in I'_B$ such that $f_0 := f(\beta_h) > 0$ and $f_j := f(\alpha_{hj})\ \forall\ j \in I_N$. Then, the *Gomory mixed-integer cut* is given by the inequality

$$\sum_{\substack{j \in I_N \cap I_I \\ f_j \leq f_0}} \frac{f_j}{f_0} x_j + \sum_{\substack{j \in I_N \cap I_I \\ f_j > f_0}} \frac{1 - f_j}{1 - f_0} x_j + \sum_{\substack{j \in I_N \cap I_C \\ \alpha_{hj} \geq 0}} \frac{\alpha_{hj}}{f_0} x_j - \sum_{\substack{j \in I_N \cap I_C \\ \alpha_{hj} < 0}} \frac{\alpha_{hj}}{1 - f_0} x_j \geq 1. \tag{2.19}$$

2.5.3 Letchford–Lodi cut

In [9] a procedure for strengthening two types of cutting methods are presented. One of these methods is a strengthening of Gomory's fractional cut as presented in section 2.5.1. The cut presented in this section, which will be referred to as the *Letchford–Lodi cut*, uses partitions of indices in order to further tighten the cut.

Using the same notation as introduced in Section2.5.1, for some $h \in I_B$, the Letchford–Lodi cut is proposed as the inequality (2.20) in the following theorem.

Theorem 2.3. [9, Thm. 2] Suppose that $f(\beta_h) > 0$ and let $k \geq 1$ be the unique integer such that $1/(k+1) \leq f(\beta_h) < 1/k$. Partition I_N into classes $Q_0, \ldots, Q_k$ such that

$$Q_0 := \{i \in I_N : f(\alpha_{hi}) \leq f(\beta_h)\}$$

and, for $p = 1, \ldots, k$, let

$$Q_p := \{i \in I_N : f(\beta_h) + (p+1)(1 - \beta_h)/k < f(\alpha_{hi}) \leq \beta_h + p(1 - f(\beta_h))/k\}.$$

Then, the inequality

$$\sum_{i \in Q_0} f(\alpha_{hi}) x_i + \sum_{p=1}^{k} \sum_{i \in Q_p} \left(f(\alpha_{hi}) - \frac{p}{k+1} \right) x_i \geq f(\beta_h) \tag{2.20}$$

is valid for the convex hull of feasible integer solutions.

As stated in [9], the cut defined by (2.20) dominates the fractional cut given by (2.17), but a general relation to the mixed-integer cut (2.19) has not been shown.

3

k-rationality

This chapter presents the investigation of using k-rationality as a method for obtaining binary solutions from the relaxed model. Firstly, Section 3.1 provides an initial analysis of matrix and determinant structures. Then, Section 3.2 details the computational methods and results related to establishing a lower bound for k. Finally, Section 3.3 offers a brief concluding discussion.

All generated data, and associated code, can be found on GitHub, `https://github.com/Vesterlund/MVEX03`.

3.1 Initial analysis

To begin the analysis, the aim is to find a value of k for some *configuration* of ground truths $\mathbf{X}$ and estimates $\mathbf{Y}$. Based on the theorems presented in [7], the most suitable property to explore is seemingly k-rationality and its relation to Theorem 2.2. As a reminder, the definition of k-rationality given in Definition 2.4 states that the goal is to find a value $k \in \mathbb{Z}_+$ such that all elements of kR^{-1} are integer for all square sub-matrices R of A.

Based on [4, Prop. 4], there exists at least one configuration for which the optimal solution in the relaxed case contains the fraction $\frac{1}{2}$, thus yielding $k \geq 2$ for that specific configuration. This was simply a configuration found by the authors which sufficed to disprove [3, Conj. 2.3.1], and the properties were not explored further. In this case, the size of the configuration is $n_{\mathbf{X}} = 3, n_{\mathbf{Y}} = 2$. For configurations of this size, it is easy to show that not all configurations have fractions as part of their optimal solution. Consider the example of $\mathbf{X} = \{(0,0),(2,4),(0.5,2.5)\}$ and $\mathbf{Y} = \{(3.5,1),(0.5,4.5)\}$ with $c = 2, \gamma = p = 1$. In this case, the relaxed and binary problems have the same optimal objective value of 6.5, with the assignments

$$w^1_{\text{bin}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \; w^2_{\text{bin}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.1a}$$

and

$$w^1_{\text{rel}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \; w^2_{\text{rel}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.1b}$$

clearly showing a binary solution in the relaxed case with an equal objective valueas in the non-relaxed case. While in this example the assignment is the same in both cases, this does not necessarily have to be the case.

In fact, generally it may be that most configurations actually do not have a fractional optimum. However, the bounding polyhedron is still the same for same sized configurations. The only difference is which extreme point of this polyhedron that is optimal, based on the specific values of the configuration. As such, for a value of k to be valid for all values of some sized configuration, one needs to examine the whole defining polyhedra itself.

3.1.1 Analytical approach

When trying to analytically determine the value of k, a starting point for the procedure is by using Cramer's rule, using the fact that the inverse of any square sub-matrix $R \in \mathbb{Z}^{n \times n}$ can be written as

$$R^{-1} = \frac{1}{\det(R)} \text{adj}(R) \tag{3.2}$$

where $\det(R)$ and $\text{adj}(R)$ denotes the determinant and adjugate of R, respectively. By design, the constraint matrix A is integer, and thus R must be as well. A result of this, when combined with the fact that the adjugate matrix is the transpose of the cofactor matrix, it holds that $\text{adj}(R) \in \mathbb{Z}^{n \times n}$ as well. From this, it becomes clear that if R^{-1} is to have any non-integer elements, it must hold that $\det(R) \neq \pm 1$ and that the entries of $\text{adj}(R)$ must also contain some element which cannot be evenly divided by $\det(R)$.

As a starting point, we need to compute the different possible values of $\det(R)$, either generally or for some specific configuration size. Assume that the chosen sub-matrix R consists only of rows and columns that originate from the set of assignment constraints, that is the constraints

$$\sum_{i=1}^{n_X} w^t_{ij} \leq 1, \qquad j \in \{1, \ldots, n_Y\}, \; t \in T, \tag{3.3a}$$

$$\sum_{j=1}^{n_Y} w^t_{ij} \leq 1, \qquad i \in \{1, \ldots, n_X\}, \; t \in T, \tag{3.3b}$$

and the variables w^t_{ij}. Since assignment problems are well-known to have totally unimodular constraint matrices, it holds that $\det(R) \in \{0, \pm 1\}$ for any square sub-matrix of the matrix defined by the constraints (3.3a)—(3.3b). As such, any sub-matrix consisting of only rows and columns present in assignment constraints cannot cause a fractional value in R^{-1}.

Thus, consider the cases where at least one TGOSPA-constraint, i.e., the constraints (2.11e)—(2.11f), are part of the sub-matrix R. For some index tuple $(i, j, t, t+1)$, the sub-matrix consists of parts obtained from rows with the following structure:

$$\begin{array}{cccccc} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c^t & c_{t+1} & c^o \\ \left[\begin{array}{cccccc} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \\ 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{array}\right. & \begin{array}{l} \text{(2.11e)}, \\ \text{(2.11f)}, \\ \text{(2.11c) or (2.11d) for } t, \\ \left.\text{(2.11c) or (2.11d) for } t+1.\right] \end{array} \end{array} \tag{3.4}$$

Here, the columns are reordered according to the variables above, with c^t, c^{t+1}, c^o denoting the set of columns not actively being considered, but corresponding to some variable present in the w_{ij}^t assignment constraints, the w_{ij}^{t+1} assignment constraint, or none of them, respectively. The rows are reordered such that they come in the following order

1. (2.11e),
2. (2.11f),
3. (2.11c) or (2.11d) for t,
4. (2.11c) or (2.11d) for $t+1$.

By examining different combinations of rows and columns for different tuples $(i, j, t, t+1)$ on the structure defined in (3.4), an understanding of the different possible determinants can be made. For simplicity, consider a combination of the rows and columns as presented in (3.4), without any rows/columns for other index values. Since our interest lies in the value of the determinant, and not its sign, operations such as reordering the rows/columns or performing Gaussian elimination on the matrix do not affect the desired outcome.

It can be shown that, for most sub-matrices R constructed to include combinations of rows from (3.4),the determinant is two times the determinant of any one dimension smaller sub-matrix R' of R. As an example, consider the case when the two TGOSPA rows (corresponding to (2.11e)–(2.11f)) and the row relating the assignment constraint for w_{ij}^t (2.11c) (or (2.11d)) are included. For the rest of the cases with similar result, see Appendix A. In this case, the resulting sub-matrix R will, after some Gaussian elimination, take the form

$$\begin{array}{ccccc} g_{ij}^t & w_{ij}^{t+1} & w_{ij}^t & c^t & c^o \\ \left[\begin{array}{ccccc} 2 & 0 & 0 & 0\cdots & 0\cdots \\ 0 & 1 & -1 & 0\cdots & 0\cdots \\ 0 & 0 & 1 & 1\cdots & 0\cdots \end{array}\right]. \end{array} \tag{3.5}$$

In (3.5), the rows not also present in (3.4) have been omitted for clarity, since they have no impact on the results presented below. However, a noteworthy detail is that

the column corresponding to g_{ij}^t will, apart from the entry 2 in the first row, consist of only zeroes.

By the properties of block diagonal matrices, the determinant of this sub-matrix is simply two times the $(n-1) \times (n-1)$ sub-matrix R' obtained by removing the first row and the first column from R. Further, R' is also a sub-matrix of the original constraint matrix A. As such, similar to an induction proof, if it holds that the smaller sub-matrix R' also shares the property that the overall determinant of R will be either zero or $\pm 2^k, k = 0, \ldots, n$, depending on the determinant of its smallest square sub-matrix that consists of only assignment constraints, i.e. constraints from (2.11c) and/or (2.11d).

However, this property does not hold for all constructable sub-matrices. Consider the case where one row related to assignment constraints for each of w_{ij}^t and w_{ij}^{t+1} and a single TGOSPA constraint is included. In this case, a slightly different result is obtained. Instead of having the determinant of R being two times the determinant of the smaller sub-matrix R', they are instead equal (up to sign). As an example, performing Gaussian elimination on the sub-matrix consisting of the assignment constraints and the second TGOSPA constraint yields

$$\begin{array}{c} \begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c^t & c_{t+1} & c^o \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 \cdots & 0 \cdots & 0 \cdots \\ -1 & 1 & 1 & 0 \cdots & 0 \cdots & 0 \cdots \\ 0 & 1 & 0 & 0 \cdots & 1 \cdots & 0 \cdots \end{bmatrix} \end{array} \tag{3.6a}$$

$$\Rightarrow \begin{array}{c} \begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c^t & c_{t+1} & c^o \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 \cdots & 0 \cdots & 0 \cdots \\ 0 & 0 & 1 & 1 \cdots & -1 \cdots & 0 \cdots \\ 0 & 1 & 0 & 0 \cdots & 1 \cdots & 0 \cdots \end{bmatrix} \end{array} \tag{3.6b}$$

$$\Rightarrow \begin{array}{c} \begin{matrix} g_{ij}^t & w_{ij}^t & c^t & w_{ij}^{t+1} & c_{t+1} & c^o \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 \cdots & 0 & -1 \cdots & 0 \cdots \\ 0 & 1 & 1 \cdots & 0 & 0 \cdots & 0 \cdots \\ 0 & 0 & 0 \cdots & 1 & 1 \cdots & 0 \cdots \end{bmatrix} \end{array}. \tag{3.6c}$$

Choosing the other TGOSPA constraint yields a analogous result. While this differs from the cases where the determinant of a sub-matrix R are two times the determinant of the one dimension smaller sub-matrix R', this still plays nicely with the idea of induction-like behaviour presented above.

The last case, not covered by the above reasoning, is the troublesome one. Consider the case where both TGOSPA constraints and the assignment constraints are included in the sub-matrix. As above, performing Gaussian elimination and reordering

rows and columns yields the following structure

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c^t & c_{t+1} & c^o \end{matrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & a\cdots & 0\cdots & 0\cdots \\ 1 & -1 & 1 & 0\cdots & 0\cdots & 0\cdots \\ -1 & 1 & 1 & 0\cdots & 0\cdots & 0\cdots \\ 0 & 1 & 0 & 0\cdots & b\cdots & 0\cdots \end{bmatrix} \tag{3.7a}$$

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c^t & c_{t+1} & c^o \end{matrix}$$
$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & a\cdots & 0\cdots & 0\cdots \\ 0 & 0 & 2 & 0\cdots & 0\cdots & 0\cdots \\ -1 & 1 & 0 & 0\cdots & 0\cdots & 0\cdots \\ 0 & 1 & 0 & 0\cdots & b\cdots & 0\cdots \end{bmatrix} \tag{3.7b}$$

$$\begin{matrix} g_{ij}^t & w_{ij}^t & c^t & w_{ij}^{t+1} & c_{t+1} & c^o \end{matrix}$$
$$\Rightarrow \begin{bmatrix} 2 & 0 & 0\cdots & 0 & 0\cdots & 0\cdots \\ 0 & -1 & 0\cdots & 1 & 0\cdots & 0\cdots \\ 0 & 1 & 1\cdots & 0 & 0\cdots & 0\cdots \\ 0 & 0 & 0\cdots & 1 & 1\cdots & 0\cdots \end{bmatrix}. \tag{3.7c}$$

While this looks similar to the cases presented above, there is one problematic part still remaining. The remaining issue lies in the second row of the latter expression in (3.7), namely

$$\begin{matrix} g_{ij}^t & w_{ij}^t & c^t & w_{ij}^{t+1} & c_{t+1} & c^o \end{matrix}$$
$$\begin{bmatrix} 0 & -1 & 0\cdots & 1 & 0\cdots & 0 \end{bmatrix}. \tag{3.8}$$

The problem is that this row does not occur naturally in the original constraint matrix. Because of this, the corresponding sub-matrix has a determinant of two times the determinant of some smaller matrix which is *not* a sub-matrix of the original constraint matrix. As such, computing this determinant cannot be done by reasoning using induction.

However, by using a Laplace expansion along the troublesome row from the obtained matrix, we can conclude that (up to the sign of the determinant)

$$\left|R\right| = \begin{vmatrix} g_{ij}^t & w_{ij}^t & w_{ij}^{t+1} & c^t & c_{t+1} & c^o \\ 2 & 0 & 0 & 0\cdots & 0\cdots & 0\cdots \\ 0 & -1 & 1 & 0\cdots & 0\cdots & 0\cdots \\ 0 & 1 & 0 & 1\cdots & 0\cdots & 0\cdots \\ 0 & 0 & 1 & 0\cdots & 1\cdots & 0\cdots \end{vmatrix} \tag{3.9a}$$

$$= 2 \cdot \left((-1) \cdot \begin{vmatrix} w_{ij}^{t+1} & c^t & c_{t+1} & c^o \\ 1 & 0\cdots & 0\cdots & 0\cdots \\ 0 & 1\cdots & 0\cdots & 0\cdots \\ 1 & 0\cdots & 1\cdots & 0\cdots \end{vmatrix} - (1) \cdot \begin{vmatrix} w_{ij}^t & c^t & c_{t+1} & c^o \\ -1 & 0\cdots & 0\cdots & 0\cdots \\ 1 & 1\cdots & 0\cdots & 0\cdots \\ 0 & 0\cdots & 1\cdots & 0\cdots \end{vmatrix} \right) \tag{3.9b}$$

$$= 2 \cdot \left(-(1) \cdot \begin{vmatrix} c^t & c_{t+1} & c^o \\ 1\cdots & 0\cdots & 0\cdots \\ 0\cdots & 1\cdots & 0\cdots \end{vmatrix} - (-1) \cdot \begin{vmatrix} c^t & c_{t+1} & c^o \\ 1\cdots & 0\cdots & 0\cdots \\ 0\cdots & 1\cdots & 0\cdots \end{vmatrix} \right) \tag{3.9c}$$

$$= 2 \cdot (-(1) \cdot \det(R_1') - (-1) \cdot \det(R_2')) \,. \tag{3.9d}$$

Here, R_1' and R_2' denote some smaller sub-matrices of R. Again, recall that each determinant contains some suitable amount of rows from other indices $i, j, t, t+1$ in order to make it square. Based on this, and the observations made for the other cases, there is a strong indication that determinants which are not a power of two can be found. This is due to the argument that if it was the case that determinants of sub-matrices could only be zero or a power of two, having the determinants inside (3.9) be differing powers of two, would yield a contradiction. As such, the determinant is not limited to only multiples of 2, and instead lies in $\mathbb{Z}$, making the analytical reasoning harder to perform, and thus a value for k more difficult to determine. Instead, we find it more useful to look at some computational methods to search for values of k for some smaller simulated problem instances.

3.2 Computational value of k

In order to find a value of k using computational methods, the program used to prove [4, Prop. 4] was used as a starting point. This program is implemented in Python, and works on the simple premise of randomly generating estimate targets and ground truth targets,which are then evaluated with the TGOSPA metric until a configuration of estimates and ground truths such that $z^*_{\text{LR}} < z^*_{\text{TGOSPA}}$ and $\exists\, w^t_{ij} \notin \mathbb{Z}$ is identified. That is, the program finds a configuration for which the optimal value, in the case of the integrality constraint being relaxed, is lower than the optimal value of the true TGOSPA metric and has some non-integer variable value as part of the optimal solution.

Our modified program works in a similar fashion as the original, generating random configurations and solving the true and relaxed metrics, but instead of terminating once a solution such that $z^*_{\text{LR}} < z^*_{\text{TGOSPA}}$ is found, our modified program keeps generating configurations until some iteration threshold is reached. Then, during the execution, every generated configuration which has a lower optimal value (than that of the true metric) for the relaxed model and contains some fractional variable in its solution is saved, and a count is kept of the times a specific fractions occur during the whole runtime. This program is then run for several different sets of *parameters* $(T, n_{\mathbf{X}}, n_{\mathbf{Y}})$, representing the number of time steps, the number of ground truth targets and the number of estimate targets, respectively.

3.2.1 Numerical bound for k

As mentioned above, running this program yields a list of example configurations where fractional solutions occur, some of which are optimal. However, since this is a computational approach using random simulations, it is possible that some

fractional solutions which have a configuration for which they are optimal may not be generated in the number of instances run. As such, the result obtained may not be able to determine an exact value for k. However, for any k to be valid, it needs to hold by definition that the fractional solutions found can at least be multiplied by k to result in an integer. As such, k needs to be *at least* the *least common multiple* of the denominator in the fractions found.

As mentioned in the beginning of Section 3.1, in [4] an example configuration where the coefficients caused the optimal solution to contain the fraction $\frac{1}{2}$ was identified. As such, for the specific set of parameters $(T, n_{\mathbf{X}}, n_{\mathbf{Y}}) = (2, 3, 2)$ used, $k \geq 2$ becomes the lower bound. By extending the method used, as mentioned in Section 3.2, considering different sets of parameters, more instances possessing optimal solutions with fractional variables were found. Further, these fractional values were not limited to only $\frac{1}{2}$, but could attain a wide range of different fractional values. The different possible fractional values obtained in an optimal solution seem to depend on the configuration size. As an example, consider the set of parameters $(T, n_{\mathbf{X}}, n_{\mathbf{Y}}) = (5, 6, 4)$. In this case, depending on the specific configuration generated, the optimal solution consisted of multiples of either $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, or $\frac{1}{6}$. Another example is the set of parameters $(T, n_{\mathbf{X}}, n_{\mathbf{Y}}) = (7, 7, 7)$, where different configurations could have multiples of some fraction $\frac{1}{l}$, $l = 2, \ldots, 10$, as part of the optimal solution.

From this, for the existence of a value of k for a specific configuration, multiplying k with any found fraction must result in an integer. In other words, for a specific parameter set, any value k is bounded from below by the least common multiple of the denominators of found fractions which are a part of some optimal solution. Utilizing this, it is possible to gain some insight in the feasibility of using methods relating to k for efficient problem solving.

In Table 3.1, the computed lower bound for k, based on the fractions generated as part of some optimal solution, is presented. More details about the exact fractional values found, together with bounds for more sets of parameters, can be found in Appendix B.

$(T, n_{\mathbf{X}}, n_{\mathbf{Y}})$	Lower bound for k
$5 \times 6 \times 4$	12
$7 \times 7 \times 7$	$2.772 \cdot 10^4$
$8 \times 8 \times 8$	$1.225 \cdot 10^7$
$9 \times 9 \times 9$	$5.084 \cdot 10^{15}$
$10 \times 10 \times 10$	$1.526 \cdot 10^{29}$

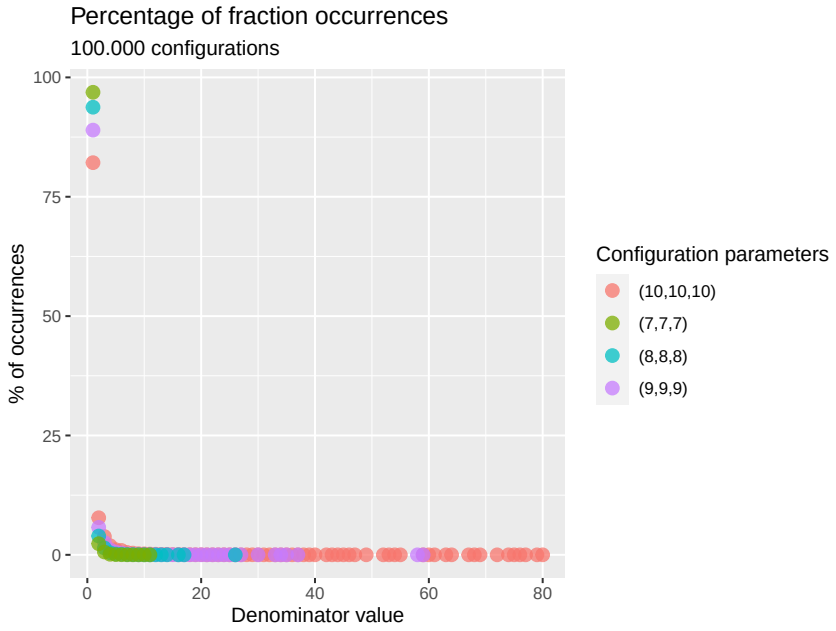
Table 3.1: Examples of computed lower bounds for k.

A clear observation from Table 3.1 is that the lower bound for k grows *very* quickly with the sizes of the parameters. The lowest possible value of k is therefore growing much faster than the size of the problem, suggesting scalability issues using methods

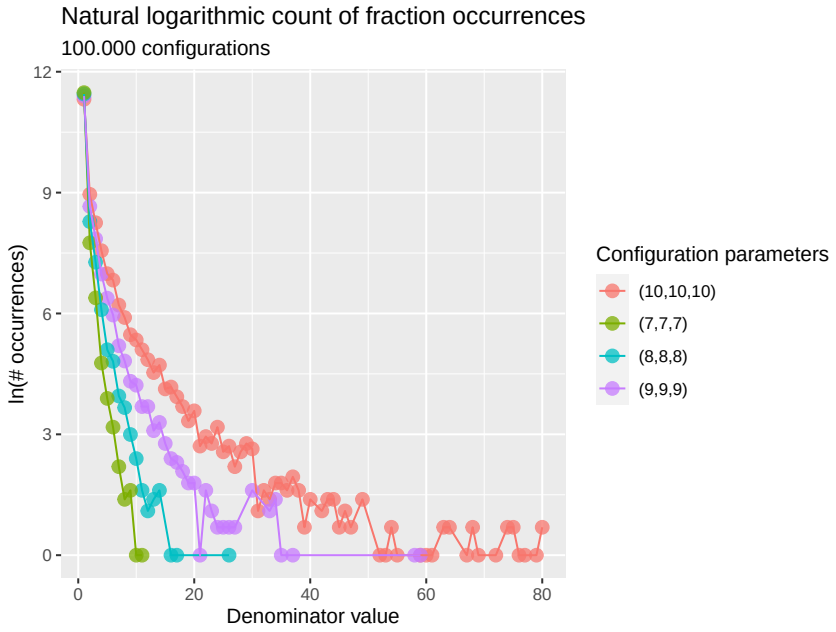
utilizing k. It can already be seen in the $(10, 10, 10)$ case, where the lower bound is in the order of magnitude of 10^{29}, a quite large number which is inconvenient for fast and precise calculations. Together, these observations provide a motivation to not explore methods relating to k-rational or k-modular properties when trying to efficiently cut the solution space. This is mainly due to the numerical and time-complexity issues likely to occur from working with the large numbers required to represent k.

3.2.2 Possible fractional parts

A side-note discovered from the computations required to find a lower bound of k, is that it is likely that the fractions which occur are connected to the specific parameter values $(T, n_{\mathbf{X}}, n_{\mathbf{Y}})$ used. During the data generation for bounds on k, every instance for which a solution contained some fractional variable value were saved and counted. From this data, some interesting observations have been made. First of all, each configuration seems to contain only multiples of some specific fraction, for example, if the fraction is $\frac{1}{4}$, the only values occurring seems to be $\frac{1}{4}, \frac{2}{4}(=\frac{1}{2}), \frac{3}{4},$ and $\frac{4}{4}(=1)$. In addition to this, fractions with a smaller denominator (i.e., a fraction closer to 1) seem to be much more common than fractions with a larger denominator. In fact, based on the data generated, for all parameter sizes chosen, there seems to be a rapid decrease in number of occurrences as the denominators of fractions grow larger. Secondly, the largest denominator of a found fraction seems to be related to the size of the parameters $(T, n_{\mathbf{X}}, n_{\mathbf{Y}})$. That is, as parameter sizes get smaller, the largest occurring denominator gets smaller as well. These observations are illustrated in Figure 3.1.



(a) Percentage

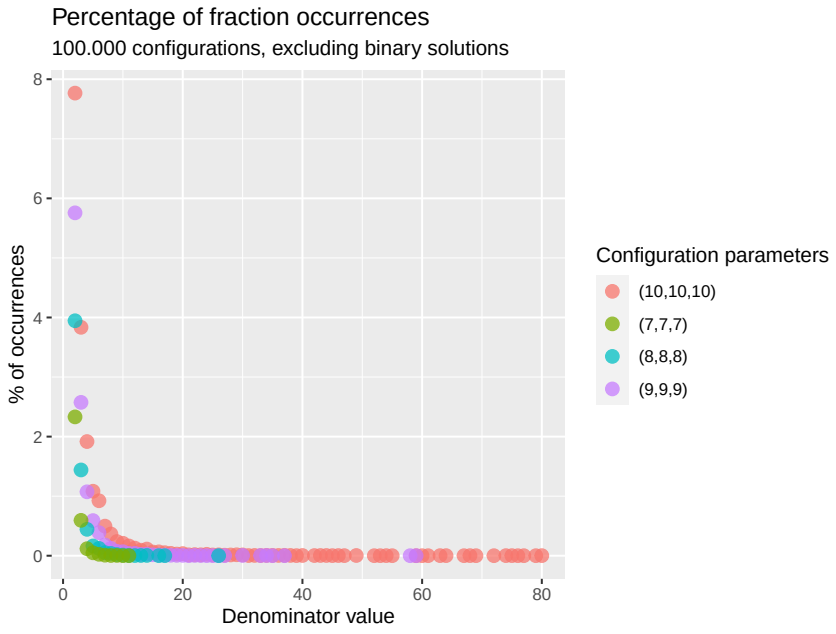


(b) Natural log-count

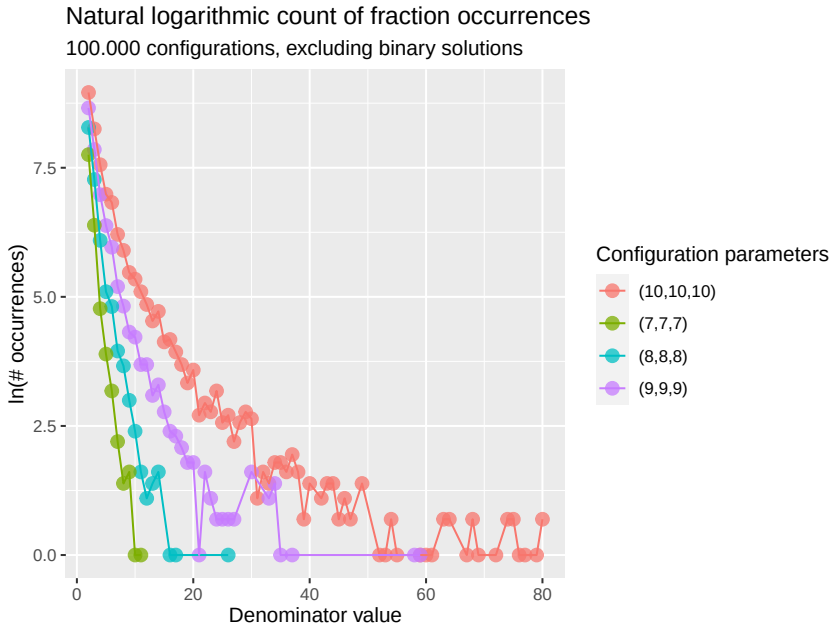
Figure 3.1: Plots showing the occurrence of fractional solution denominators in generated configurations for various sets of parameter values.

Further, in Figure 3.1, the percentage of occurrences of the binary solutions (corresponding to a denominator value of 1 in the plots)constitute a clear majority for all the parameter sizes tested. Since the binary solutions quite heavily dominate the percentage plot, Figure 3.2 shows the percentages for the non-binary solutions. Further, while the larger fractions such as $\frac{1}{2}$ and $\frac{1}{3}$ can be considered quite "common" whenever fractions occur, any smaller fractions quickly diminish in observations.

However, as can be seen from the natural log-count, when the parameter values increases, more unique fractions occur and the instances of larger fractions occurring becomes more common.



(a) Percentage



(b) Natural log-count

Figure 3.2: Plots showing the occurrence of fractional solution denominators, excluding binary solutions, in generated configurations for various sets of parameter values.

3.3 Discussion

Determining the exact value of k for a given configuration proves to be unexpectedly complex. Initially, it was assumed that introducing time dependence, given the problem's resemblance to a series of time-connected assignment problems, would have only a minimal impact. However, both the initial analysis and the numerical results clearly show that this assumption was incorrect. The addition of a time dependency on each variable significantly increases the complexity, transforming the problem into something far more intricate than a standard assignment problem. In addition, not only is k non-trivial to determine, its value is also potentially very large for even small sizes of problem instances.

To expand on the reasoning why k being large becomes a problem, the main reason is due to the computational aspect of solving real instances of the problem. Considering that for the relatively small parameter size of $(10, 10, 10)$, the lower bound of k is already in the order of 10^{29}. For some perspective, this number is roughly 10^{10} times larger than the maximum unsigned 64-bit integer. As such, for most computational implementations, one would need to take extra precautions to handle such large numbers, let alone to deal with the multiplication of it with other entities such as vectors, matrices, or floating point numbers to avoid numerical issues. This would result in additional complexity in the implementation of the program, but likely also additional runtime from the need for handling special implementations of integer classes. As such, the initially proposed method of using k-rationality and/or k-modularity becomes seemingly uninteresting due to the complexity associated with the number k, even if the value of k would be easy to calculate.

Note that this does not mean that finding k is impossible, it's quite likely the opposite. From the side-note results from the computations where fractions were generated, there is a strong suggestion that there in fact exists a k and that its value is bounded based on the parameters of the problem. However, since this thesis focuses on identifying methods that could be useful for obtaining integer solutions when solving a relaxed version of the problem, if the methods using k are likely unreasonable, the value of k itself is uninteresting. As such, not much research and focus was put toward finding k after the computational results suggesting it to be inconveniently large. The data pointing toward k being bounded is the fact that the number of occurrences for each fraction seems to be decreasing as the denominator of a fraction increases in general, and that the number of distinct fractions showing up is fewer for smaller parameter values, but increases as the size of the parameters increase. As an example, for the $(T, n_{\mathbf{X}}, n_{\mathbf{Y}}) = (10, 10, 10)$ case, it can be seen that most fractions $\frac{1}{l}$, $l \in 2, 3, \ldots, 80$ do show up at least once, but the closer to $\frac{1}{2}$ the fractions are, the more often they appear.

Assuming that the smallest fraction that can appear in a solution is $\frac{1}{l}$ for some $l \in \mathbb{Z}_+$ depending on the parameters, I believe that it is possible to find at least one configuration for which the optimal solution is fractional for each of the fractions $\frac{1}{2}, \ldots, \frac{1}{l}$ if the program were to run infinitely. However, since this is not the main

focus, this theory was not explored further.

4

Cutting methods

This chapter contains the details regarding the investigation of using cutting plane methods in order to obtain binary solutions in the relaxed model. First, some implementational details regarding the examined methods are presented in Section 4.1. Then, in Section 4.2, the results obtained from the implementation of the cutting methods are presented, before the concluding discussion and analysis is presented in Section 4.3.

The data used for the analysis, and associated code, can be found on GitHub, `https://github.com/Vesterlund/MVEX03`.

4.1 Implementational details

The implementation of (2.11) was made in Julia using the JuMP [10] package as an interface for working with the Gurobi optimizer. This program was designed in such a way that it should be possible to solve the true problem without any relaxation, and compare the optimal solution obtained with the solution to the relaxed version together with any potentially added cuts.

The program starts by solving the instance of the model where the integrality requirement of the variables $w_{ij}^t, g_{ij}^{t'},\ t \in T, t' \in \hat{T}$, have been relaxed. Then, if the solution is non-integer, the program attempts to create a cut using some cutting plane method to generate and introduce an inequality to the model. This new inequality should be a constraint such that the previous optimal solution is no longer feasible, yet no binary solution is cut away from the solution space, as described in Section 2.5. Solving the new model with the added cut yields a new optimal solution. If if happens that this solution is also not integral, it can again be used to generate a cut. This process is then repeated until an binary solution has been found, some iteration limit has been reached, or numerical errors occur. The topic of numerical errors will be discussed further in Section 4.3.1.

Summarized, assuming that the relaxed model has a non-integer optimal solution, the program logic is as follows:

1. Solve the relaxed model
2. Generate a cut based on the solution found
3. Add the cut (i.e., affine inequality constraint) to the existing model
4. Solve the new model
5. If the solution is integer, or some other stopping criterion is met, stop. Otherwise, repeat from step 2.

In order to examine the efficiency of the cutting methods, they were run on the configurations generated in the process of finding a computational value for k in Section 3.2. Thus, for each set of parameters, the cutting plane methods could be compared across a wide range of configurations and instances, each with different fractional components as part of their optimal solution. The result from this procedure is presented in Section 4.2.

Note on runtime
A small note on the runtime of the program comes from the usage of JuMP as an interface to the Gurobi optimizer. Since Gurobi is the software actually solving the model constructed using JuMP, this has a quite severe effect on the runtime of each iteration of adding cuts. JuMP is simply a way of constructing a model using Julia code, and as such, anytime the model is updated and re-optimized, it is simply re-sent from the JuMP interface to the Gurobi optimizer, without making use of any previously known information about the problem (unless explicitly specified). A result of this is the fact that, after adding a cut to a solved model, the optimizer is essentially solving—from its point of view—a completely new and unknown problem. Without making suitable adjustments, this means that runtime measurements will be deceiving when compared against a model solved only once. Thus, for the context of method performance, only the number of iterations needed will be considered. Investigation of the timed performance of these methods is something that could be examined further as proposed in Chapter 5.

4.1.1 Basis cutting methods

The cutting plane methods utilizing the basis and non-basis matrices, are all implemented in a similar fashion. Gomory's fractional cut, the strengthened Gomory's fractional cut, Gomory's mixed-integer cut, and the Letchford–Lodi cut all utilize these matrices in a similar, albeit slightly different, way. From the solved problems, the optimal solution x^* can be used to construct the basis matrix B and the non-basis counterpart N. Then, using a common interface for cutting plane methods, these matrices are used to compute the relevant values required for each method.

However, due to the nature of requiring the computation of the inverse B^{-1} in the construction of the cutting inequalities, all of the methods utilizing the basis matrix

are expected to suffer to some degree from numerical instabilities. The general instability of inverse computations is well known in applications of numerical linear algebra, and can cause wrong or misleading results. To avoid this, one could use an exact representation of each fraction, using Julia's built-in `Rational` type. However, computations using exact representations in that way are significantly slower as the numerator and denominator in a fraction becomes larger. Instead, one simple way is to do the computations with some specific precision large enough to not majorly affect the result. While there are more sophisticated ways to deal with these numerical issues, for the purposes of this thesis the method of rounding was considered decent enough.

As for the selection of which $h \in I_B$ to base the cuts upon, the index can simply be chosen as $h \in I_B$ such that$f(\beta_h) \geq f(\beta_j)\ \forall\ j \in I_B \setminus \{h\}$. Recalling that $f(\beta_h)$ is the fractional part of β_h, this simply means that h is the index of the basic variable in the optimal solution that is violating the relaxed integrality requirement the most. However, it is also possible to use other ways to select h considering other factors as well. One such way is to use so called *lexicographic ordering*.

4.1.2 Lexicographical ordering

In general, lexicographic ordering refers to picking elements of a set in some predetermined (sorted) order based on properties of its elements. A simple example is picking letters from a set in alphabetical order. In the case of cutting methods, it can be used as a way to decide which index $h \in I_B$ is to be selected for the construction of the cuts, in order to to improve the efficiency of the cutting plane methods. For example, it might be desirable to prioritize selecting indices for variables w_{ij}^t which originate from the assignment problem part of (2.7), and leaving the TGOSPA variables g_{ij}^t together with the introduced slack variables to be defined as a result of restrictions placed on other variables.

As such, a version of lexicographical ordering for index selection of $h \in I_B$ was made based on the idea of prioritizing the assignment part of the problem. It consists of ordering the variables such that the prioritized order for index selection is given by $w_{ij}^t \to s_w \to g_{ij}^t \to s_g$, where s_w and s_g represent slack variables corresponding to assignment- and TGOSPA-constraints, respectively. The argument for this ordering being useful comes from the fact that the variables of the problem are very tightly coupled to each other in both the assignments of each time-step (index i and j) and between the time-steps (index t). Thus, prioritizing the variables for each of the separate time-steps would result in each time-step consisting of integer variables, and the connections between them be sorted out by proxy. The resulting order were to prioritize cuts for the w_{ij}^t variables first, then the associated slack variables s_w, before finally considering the variables introduced by the TGOSPA-constraints, g_{ij}^t and s_g. Then, within each selection group, the variable with the largest fractional violation is again selected first, as mentioned at the end of Section 4.1.1. A comparison between the performance of different cutting methods using the standard ordering versus the lexicographical ordering is presented in Section 4.2.2.

4.2 Numerical results

As mentioned in Section 4.1.1, the cutting methods evaluated are Gomory's fractional cut, its strengthened variant, Gomory's mixed-integer cut and the Letchford–Lodi cut. These cutting plane methods were tested on the instances generated when finding the bound for k described in Section 3.2. The data presented here is selected to illustrate the overall behaviour of these methods as observed from a more complete set of data. Some additional tables and plots are available in Appendix C, and the full data is available on GitHub, `https://github.com/Vesterlund/MVEX03`.

As a first example, in Figure 4.1 the performance of the cutting methods is shown for the parameters $(T, n_{\mathbf{X}}, n_{\mathbf{Y}}) = (7, 7, 7)$, in an instance where the optimal solution contains multiples of the fraction $\frac{1}{4}$.

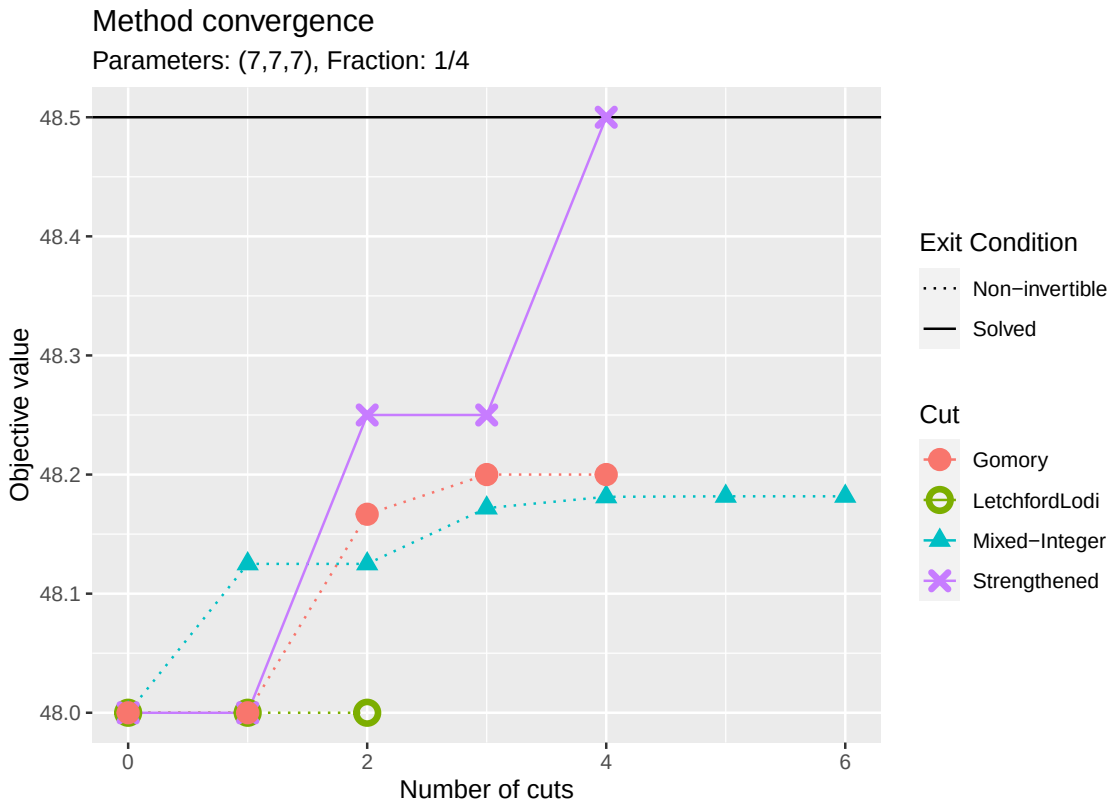


Figure 4.1: Plot showing cutting plane methods' performance on a $(7, 7, 7)$ parameter set for a configuration with $\frac{1}{4}$ fractional solution. The solid black line indicate the optimal value for the binary solution, and the linetype of each cutting plane method denotes the reason for termination.

From Figure 4.1, a few observations can be made. Firstly, the only cutting method that solves—meaning to attain the correct integer optimal solution—this configuration is the strengthened version of Gomory's cut. All other methods run into numerical issues before finding an optimal binary solution. Notably, the standard Gomory cut finds solutions whose objective values are equal (or very close) to the

optimal objective value in the original problem. However, these near optimal solutions contain some fractional part which results in additional cuts being added and numerical issues arising before the binary solution is found. Conversely, the Letchford–Lodi method gets instantly stuck and runs into issues without getting any closer to the original objective. While for this instance only one method is able to solve the example to integrality, this is not generally true.

For example, consider a different configuration for the same parameters $(7, 7, 7)$, this one with multiples of the fraction $\frac{1}{6}$ as part of its optimal solution. From Figure 4.2, it is clear that all but the Mixed-Integer cut solves this problem. The only difference between the successful methods is the number of cuts needed to find the integer optimum. The Letchford–Lodi cut is the fastest with only three cuts needed, followed by the strengthened and regular Gomory cuts, at five and twelve added cuts each, respectively. The Mixed-Integer cut seems to struggle to achieve any improvement, at least on a similar level as the other methods.

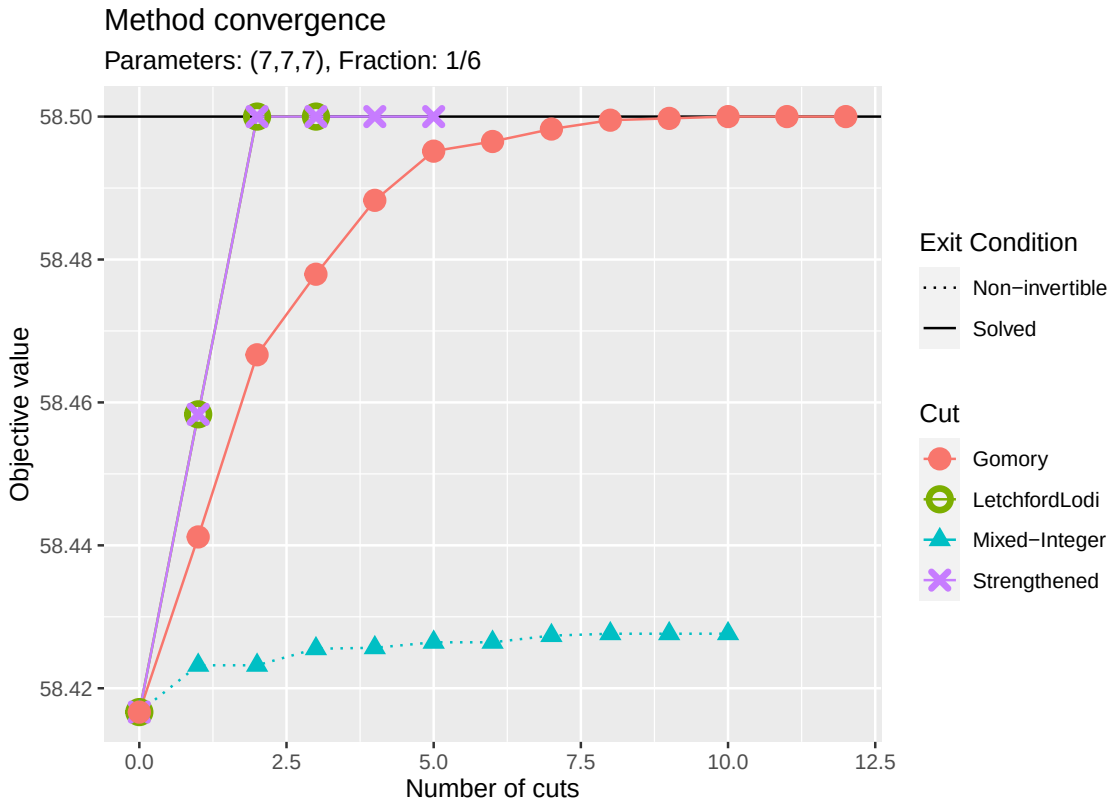


Figure 4.2: Plot showing cutting plane method performance on a $(7, 7, 7)$ parameter set for a configuration with $\frac{1}{6}$ fractional solution. The solid black line indicate the optimal value for the binary solution, and the linetype of each cutting plane method denotes the reason for termination.

Now, looking at it more broadly, Table 4.1 gives an overview of the performance of each method for a few different configurations using the $(7, 7, 7)$ parameters. Each column in the table represents a configuration with some multiple of a fraction as part of its relaxed optimal solution. Each column in the table shows a configuration

where the optimal relaxed solution contained multiples of the fraction on the first row (i.e. the $\frac{1}{3}$ column represents a configuration with values $\frac{l}{3}$, $l \in \{0, \ldots, 3\}$ in the optimal solution) On the next line, the optimal value of the binary constrained model is shown. Then, for each of the five methods, we report the final objective value (first line) and the number of added cuts used to attain that value (second line). While the fractions are presented in the table, do note that it may be the specific generated configuration mattering more for the performance of the methods, than the optimal solution of this configuration containing specific fractional values. It may be the case that configurations for the same parameters, with the same fractional part of their optimal solution, show different performances as well. This is further discussed in Section 4.3.

Fraction:	**1/2**	**1/3**	**1/4**	**1/5**	**1/6**	**1/7**	**1/8**	**1/9**	**1/10**
Binary objective	60.00	47.00	48.50	31.75	58.50	51.00	55.00	46.50	51.00
Gomory	59.72 *61**	47.00 1	48.20 *5**	31.75 2	58.50 12	51.00 4	54.67 *3**	46.50 *78**	50.87 *101†*
Letchford–Lodi	59.81 *59**	47.00 1	48.00 *3**	31.75 2	58.50 3	51.00 5	54.75 *9**	46.50 5	50.75 *4**
Mixed-Integer	59.50 *5**	46.67 *4**	48.18 *7**	31.70 *14**	58.43 *11**	50.94 *30**	54.59 *9**	46.40 *14**	50.76 *24**
Strengthened Gomory	59.75 *5**	47.00 1	48.50 4	31.75 2	58.50 5	51.00 6	54.80 *65**	46.50 *7**	50.86 *35**

Table 4.1: For the $(7, 7, 7)$ parameters, a number of configurations, characterized by the smallest fraction in an optimal solution to the relaxed problem, the table lists the binary optimal objective value and—for each of the cutting-plane methods—the closest to optimal value found and the corresponding number of cuts added. The markings * and † indicate that—for cases for which the binary optimal value was not reached—the termination criterion used is numerical issues and iteration limit, respectively.

As can be seen, the performance is quite similar for all the methods with regard to the number of times they solve the problem and number of required cuts to do so. The only outlier is the Mixed-Integer cut which does not solve any of the instances presented.

Consider now a set of larger parameter values, $(T, n_{\mathbf{X}}, n_{\mathbf{Y}}) = (10, 10, 10)$. In Figure 4.3, the performance on a configuration with multiples of the fraction $\frac{1}{5}$ as part of its optimal solution is illustrated.

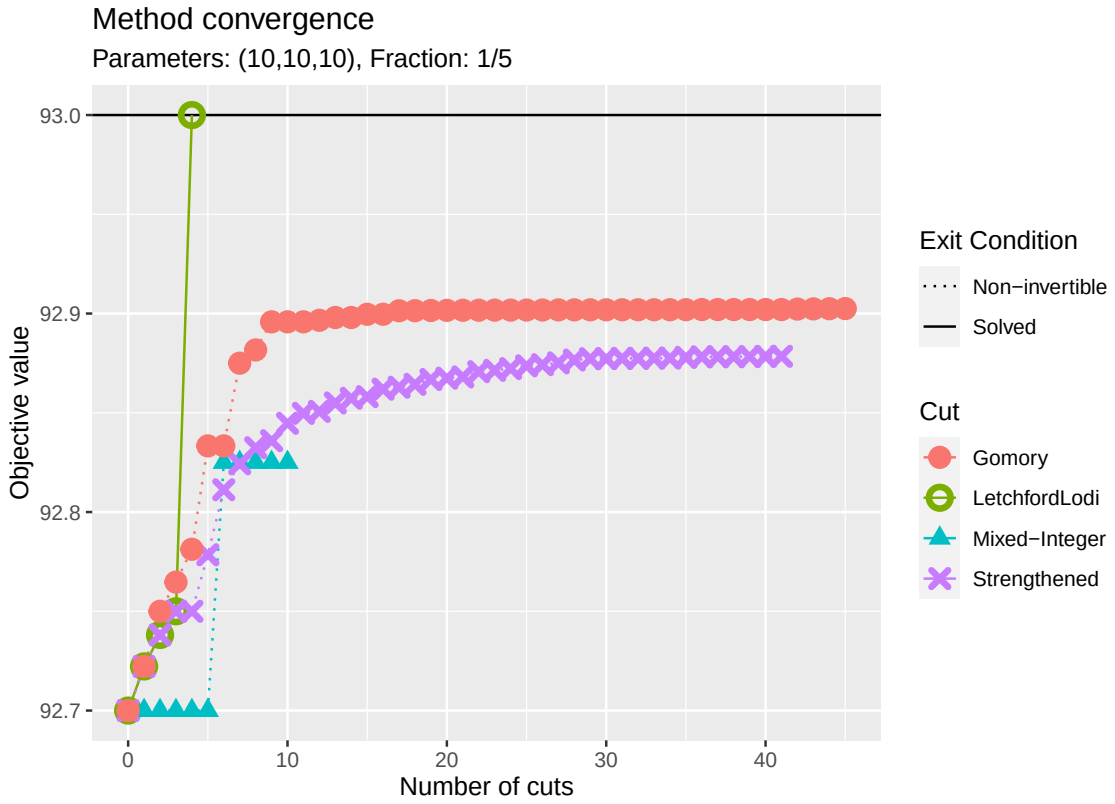
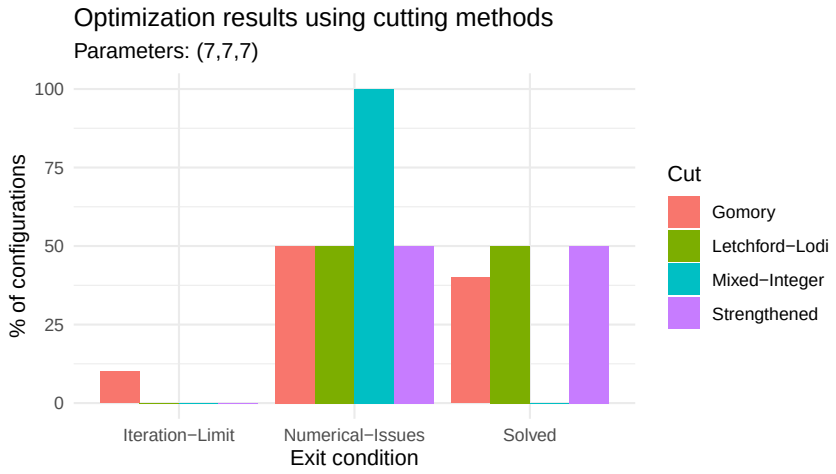
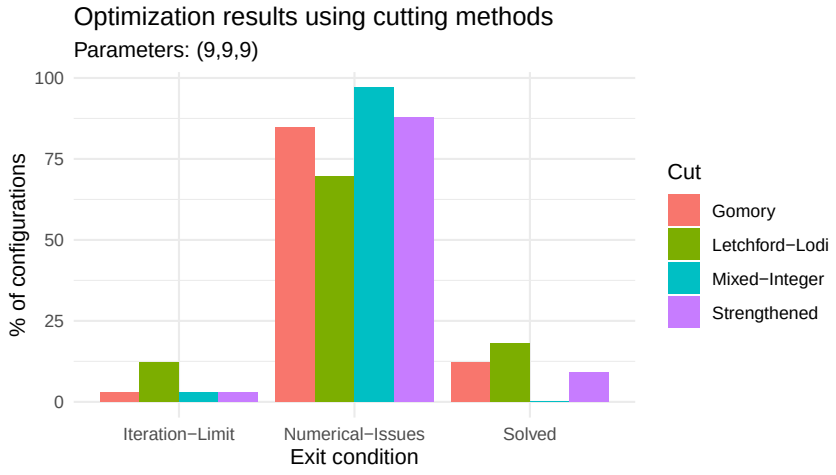


Figure 4.3: Plot showing cutting plane method performance on a $(10, 10, 10)$ parameter set for a configuration with $\frac{1}{5}$ fractional solution. The solid black line indicates the optimal value for the binary solution, and the linetype of each cutting plane method denotes the reason for termination.

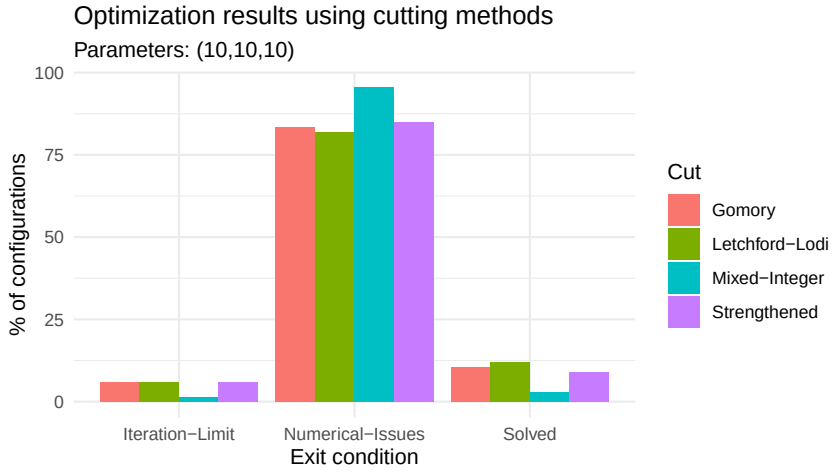
In this case, only Letchford–Lodi manages to find the optimal binary solution, while all the other methods fall short. However, when examining multiple different configuration values, there seems to be a general trend of the methods not being able to solve the problem. Specifically, as the parameter values grow larger, less generated configurations are solved and more configurations end up in numerical issues and/or iteration limits. Looking at Figure 4.4, we can see a comparison between the cutting methods performance in the case of $(7, 7, 7)$, $(9, 9, 9)$, and $(10, 10, 10)$ parameter values, respectively.



(a) $(7, 7, 7)$



(b) $(9, 9, 9)$



(c) $(10, 10, 10)$

Figure 4.4: Plot showing the percentage of configurations reaching iteration-limit, having numerical issues, and being solved, respectively, for different sets of parameter values.

From these plots it is clear that the number of solved configurations decreases as the parameter sizes increase. The number of instances being limited by number of iterations also increases slightly, but the dominating reason for a method not being able to solve an instance is that it runs into numerical issues before reaching the iteration limit..

4.2.1 Numerical stability

During the testing of the different cutting plane methods, some unexpected numerical issues began to occur. After some investigation, it seemed that the implementations of the cuts are quite sensitive to numerical instabilities. Namely, changing the way computations are done or the precision of rounding computations have a significant impact on the result and performance of these methods. This imprecision likely arises from the fact that the implementation of these methods relies on the computationally calculated inverse of the basis matrix, which introduces rounding errors and perturbs the numerical accuracy.

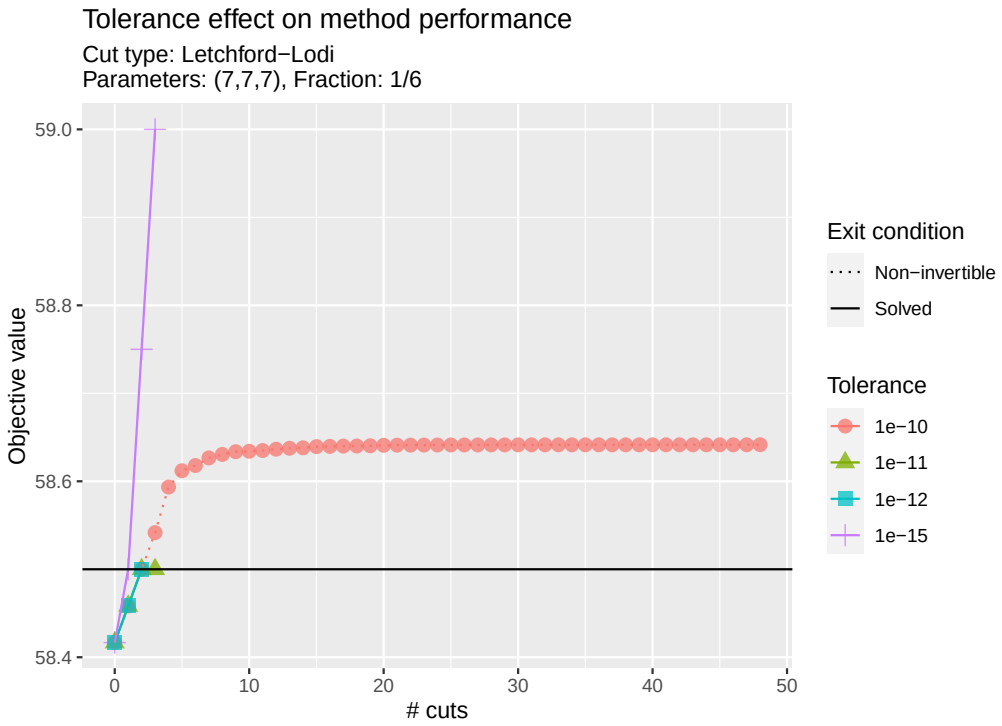


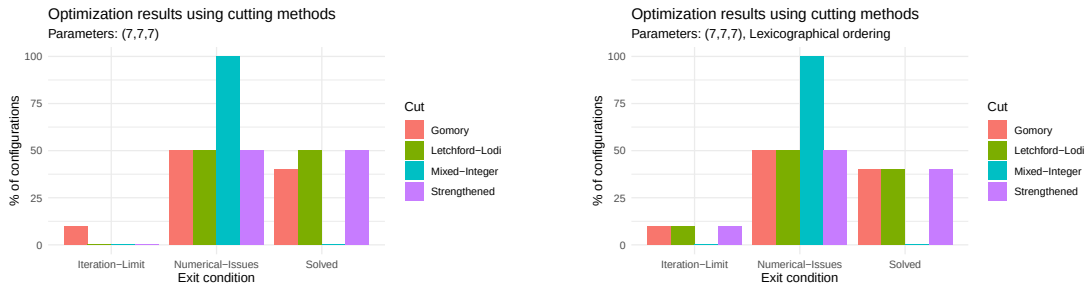
Figure 4.5: Plot showing an example of the effect of rounding on method performance. The solid black line indicates the optimal value for the binary solution.

As an example, consider the performance of the Letchford–Lodi cut for different rounding precisions when doing calculations and computing the basis inverse. Figure 4.5, shows that the performance of the method depends a fair bit on the numerical precision used when performing computations. Notably, for a tolerance of 10^{-10} and 10^{-15} it actually results in the method missing the optimal binary solution, while still being able to find it for 10^{-11} and 10^{-12}. As such, the imprecisions caused

by different computational tolerances can have a large enough effect such that the optimal binary solution becomes infeasible.

4.2.2 Lexicographical ordering

Figure 4.6 shows a comparison of using the regular order, i.e., picking the variable with the largest fractional value, to using the lexicographical ordering of prioritizing variables relating to w_{ij}^t.

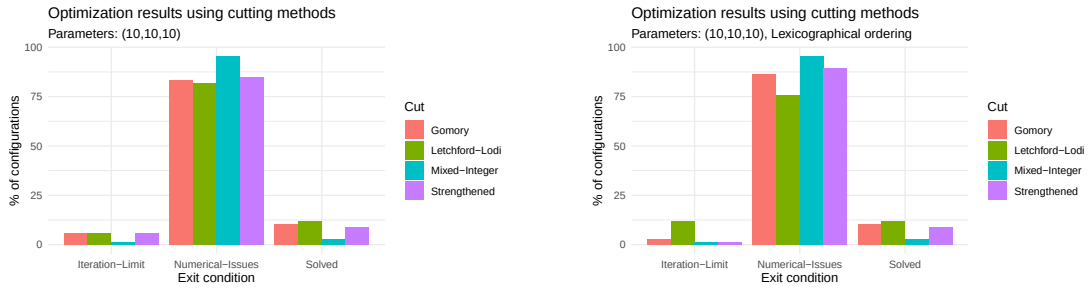


(a) Regular order **(b)** Lexicographical order

Figure 4.6: Comparison of orderings in percentage reaching iteration-limit, having numerical issues, and being solved, for configurations in the $(7, 7, 7)$ parameter set.

From this figure it is clear that this lexicographical ordering does not have any large effect on the outcome of the cutting methods. In fact, it seems to make it slightly slower, having slightly less configurations solved and more configurations reaching the iteration limit. Interestingly, it does not seem to affect the cases where the numerical issues occur, but only the cases where the methods could already solve the problem.

Further, Figure 4.7 shows a similar result for the $(10, 10, 10)$ parameter case.



(a) Regular order **(b)** Lexicographical order

Figure 4.7: Comparison of orderings in percentage reaching iteration-limit, having numerical issues, and being solved, for configurations in the $(10, 10, 10)$ parameter set.

In this case the lexicographical ordering again makes the Letchford–Lodi method hit the iteration limit more often, this time only in cases for which the regular ordering

run into numerical issues. For the Gomory and strengthened Gomory cut methods the lexicographical ordering conversely cause more cases to run into numerical issues instead. However, for all of these configurations, the number of solved cases are unchanged between the different orderings.

4.3 Discussion

Overall, comparing the cutting methods we find that Letchford–Lodi generally performs the best of the methods compared. The strengthened version of Gomory's cut and the regular version are seemingly comparable in their performance, while the mixed-integer cut performs quite bad. However, as the parameter values increase, the differences between Letchford–Lodi, strengthened and regular Gomory's become less distinguishable.

Since all the methods are based on the same underlying theory of basis matrices and Gomory's cut, it is expected for the methods to perform somewhat similarily. However, the mixed-integer methods do not seem to work at all, even for smaller parameter values where the other cutting methods usually perform quite well. There are a few reasons as to why this would be the case. Firstly, this could simply be a case of implementation where, as will be discussed in Section 4.3.1, the numerical properties of this cut is much more sensitive to details in its implementation than the other cuts. Another, perhaps more likely reason, comes from the properties of the problem itself. By the constraints imposed on the TGOSPA metric, the variables g_{ij}^t, which are not required to be integer by the problem formulation, are in practice always restricted to be integer by the constraints and the values of the other variables involved. As such, treating it as a continuous variable in the implementation might actually result in poor performance of this method. A potential improvement would then be to treat g_{ij}^t as integer as well in regard to this cut, since according to [5, Remark 5.20] this would dominate the regular Gomory fractional cut.

As mentioned, the observed results are quite expected. Since the cutting methods are based on the same theory, the instances for which the methods converge to a binary solution are unsurprisingly largely overlapping. The differences are mainly in the number of cuts needed to be added before such a solution is found, or numerical problems occur. From [9] we know that the Letchford–Lodi cut dominates the strengthened version of the Gomory cut, which in turn dominates the simple Gomory cut. As a result of this it is reasonable to assume that the dominating cuts converge more quickly to whatever result the methods find in the end, but this has not been shown to be true in general. However, a small consequence of this seems to be that the dominating cuts tend to perform ever so slightly better in general, perhaps due to this faster convergence avoiding some numerical issues that may occur. However, the result is in general that the cuts perform roughly the same, and that the performance of each of them seem to worsen as the dimension of the solution space increases with increasing parameter sizes.

A note on the fractional values' part of a solution is that they may or may not play

a part in the efficiency of the cuts. While the methods perform worse for larger parameter sizes, and thus larger dimensional solution spaces, it could be that they simply perform worse for fractions with larger denominators (and thus smaller fractional values). As can be seen in the results, and again will be mentioned in Section 4.3.1, the methods all seem to struggle with numerical issues. As a result of this, it may simply be that the cutting methods are good at solving the instances where the fractions are relatively large, which they all are for the smaller parameter sizes, and perform poorly when the fractions are smaller, which become more common as parameter size increases. Thus, by running the tests on one configuration for each fraction each, as done in this project, it may be that the worsening performance is skewed by the fact that fractions that appear for larger parameter sizes do not exist for the smaller cases. This would lead to methods seemingly performing poorly as parameter values grow larger, instead of concluding that they simply perform worse when the optimal solution contains small fractional variables. If this connection between smaller fractions and poor performance is substantial, it could be worthwhile to connect the performance of the cutting plane methods to the bounds found for k in the previous chapter. However, it could also be that the value of the fractional part of the optimal solution simply plays no role and only the parameter values matter, which increase the dimension of the solution space. This has not been studied as a part of the thesis, but is something that could be explored in future work related to this problem.

4.3.1 Numerical issues

We know that a key part of these cutting methods is that we can obtain a basis matrix B which is square and invertible. By design, this matrix should always be obtainable as long as the problem is solvable. However, during the implementation of these cuts it became clear that sometimes the resulting basis matrix from the Gurobi optimizer was non-square, even when specifying the solver to use the simplex method. Thus, additional cuts could not be created due to B lacking an inverse. After some additional analysis, this seemed to always occur in cases where the last few added cuts had given no non-negligible improvement towards the binary optimal objective value, but only potentially relatively small improvements. The conclusion drawn from this is that some of the cuts added, in combination with numerical precision errors, cause the constraints of the model to be almost linearly dependent, resulting in a strange behaviour when solving the model.

In order to investigate this a bit further, we also compared differing levels of numerical precision when performing computations. As it turned out, depending on the method, simply changing the level of precision from 10^{-15} to 10^{-16}, thus allowing an extra decimal for computations, were sometimes enough to cause methods to solve configurations which previously had numerical issues, or vice versa. As such, it seems that for this implementation these cuts are all quite sensitive to small perturbations in the coefficients, causing the cuts to run into issues when being used with an non-exact representation. But finding which levels of precision worked was also not clear. As can be seen in Figure 4.5, for the same configuration and method,

but using different numerical precisions, the results can vary a fair bit. The problematic part is that the precision working best for a specific method varies between configurations. While this was not explored in detail, it becomes troublesome as one considers implementing the methods in practice. It is easy to determine if the found optimal binary solution is correct when knowing the true answer from solving the binary model, but unhelpful if the solver simply finds an binary solution without knowing if it is the correct one or simply a product of numerical instability. Since it is a possibility that the method finds a non-optimal binary solution due to numerical issues, it becomes hard to verify the results from using these methods.

Now, that is not to say these methods are bad or incorrect, it simply means that one should take precaution during implementation and usage to prevent these issues from occurring. However, exactly how such measures should be implemented is outside of the scope of this thesis.

4.3.2 Lexicographical ordering

From the results presented in Section 4.2.2, it is quite clear that the prioritized order of w_{ij}^t and slack variables associated with the assignment constraints before g_{ij}^t and the slack variables of the TGOSPA constraints is not helpful for getting better solutions. The only real effect it seems to have is to slow down convergence such that it more often hits the imposed iteration limit instead of either solving the problem or running into numerical issues. It may be that the prioritized order is wrong or sub-optimal; there is an argument to choosing g_{ij}^t first due to the TGOSPA constraints being the constraints introducing fractional solutions, however, it is also quite likely that it's more complicated than that. Given that the variables are coupled by the constraints of the problem, some more sophisticated logic may be needed in order to construct a good ordering, perhaps specific for each configuration. This argument comes from the fact that if one variable is fixed, it also fixes many other by proxy due to most of the assignment constraints limiting to having at most one non-zero variable among the variables included in the constraint. As such, the conclusion that one type of variable is "generally more important" may be irrelevant since the simple lexicographical ordering did not improve the results.

There exists some theory (see [5, Sec. 5.2.5]) motivating why a good lexicographical ordering can improve the convergence of Gomory's fractional cut. However, in this case either a better order is needed or it may simple be of less use for this specific problem.

5

Conclusion & future work

As a conclusion, we find that trying to find a value k in order to use the properties of k-rationality is difficult. Further, if one was to find an exact value for k, it would be such large that it would be unlikely to provide any advantage in methods trying to find optimal integer solutions to the relaxed model.

The investigation of the cutting methods based on Gomory's cutting methods showed that, while these methods might be helpful in finding optimal integer solutions, they generally run into either numerical issues or converge quite slowly. These numerical issues likely arise from the implementation of these methods and as such shows that one requires to take consideration when implementing them for problem solving. Additionally, trying to use a lexicographical ordering to improve results needs more thought behind the ordering in order to improve the convergence speed.

For future works in relation to this problem, there are a number of different topics that could be investigated. These are presented below in no particular order.

Timing cutting methods
As briefly mentioned in Section 4.1, the methods were not timed due to the interaction between JuMP and Gurobi. However, it would of course be of interest to see how much faster it is to solve the relaxed problem and iteratively adding these cuts to the TGOSPA problem, than it is to simply solve the binary model using, e.g., a branch-and-bound algorithm. In order to do this, one would need to focus a bit more on the implementation of the model, making sure that adding an additional cut makes use of previously known information about the solution space from solving the model. Finding out if these cutting methods are significantly faster could help in getting a good bound of the optimal objective when computation of the exact objective is slow.

Finding the true value of k
While this thesis concludes that the value of k is of no interest in regard to computations and usage of k-rationality, one could be interested in find the value of k itself for some specific set of parameter values (or in general) anyway. Finding the value of k would likely be a more analytical work, studying the structure and underlying polyhedra of the standardized formulation $Ax \leq b$, as expressed in (2.7), but one

could possibly make use of some computational methods as well to try and brute force some better bound.

Other ways to reduce solution space
As this thesis mainly focuses on cutting methods based on Gomory's fractional cuts and the basis matrix B, there are of course other methods to reduce the solution space. It may very well be that there exists other methods which are more suited to the specific symmetric structure that occurs in the standardized A matrix of the problem, and/or the integrality of the problem.

Better lexicographical ordering
As stated in Section 4.2.2, the specific lexicographical ordering tested turned out not to be so useful. However, this order was chosen without giving too much thought into the details of it. By studying more about lexicographical ordering in general, and using such knowledge to examine the problem and construct a better ordering principle, it might be possible to get more use out of the methods studied in this thesis and to achieve better results.

Chvátal–Gomory cuts
At a late stage during this thesis, the prospect of using *Chvátal–Gomory cuts* (CG-cuts) as a cutting method was discovered to be possible, due to their close relation to Gomory's fractional cuts. Due to this late discovery, the prospect of using CG-cuts to solve this problem has fair potential for further exploration. Namely, one might be able to avoid most numerical issues arising from using basis matrices by instead rewriting any fractional Gomory cuts on their equivalent CG-form, or by removing the requirement of numerically computing the inverse of the basis. It may also be that these methods—if one is able to easily find the required weighting to use them—have a better performance on the overall problem than basis-based cutting methods.

Numerical precision for cutting methods
As noted in Section 4.2.1, the cutting methods studied are quite heavily affected by the numerical precision used for computations. As such, it might be of interest to investigate the general implementation of these types of cuts and how to minimize or eliminate the effects of numerical imprecisions.

Cutting methods related to fractions
Since the performance of the cutting methods get worse as the parameter sizes increases, an interesting question is whether this is due to the increase in dimension of the solution space or due to the smaller fractions potentially causing increasing convergence issues. Now, while these properties might be related, it may be that the methods actually perform better for the solutions where the fractional part is larger compared to the case when smaller fractional parts occur.

Heuristic for non-solved instances
If it is difficult to find and implement methods which can solve the relaxed problem

to integrality without numerical issues or a large number of iterations, it may be desirable to implement a heuristic to obtain an upper bound on the optimal objective value. In those cases, if there exists a decent heuristic, one can at least know in what, hopefully narrow, range the optimal objective lies for the binary model. One could use the obtained objective value for the relaxed problem as a lower bound, the heuristic objective for the upper bound, and thus get a decent idea of the performance of an algorithm without having to find an exact optimal solution.

Bibliography

[1] A. S. Rahmathullah, Á. F. García-Fernández, and L. Svensson, "Generalized optimal sub-pattern assignment metric," in *2017 20th International Conference on Information Fusion (Fusion)*, 2017, pp. 1–8. DOI: 10.23919/ICIF.2017.8009645.

[2] Á. F. García-Fernández, A. S. Rahmathullah, and L. Svensson, "A metric on the space of finite sets of trajectories for evaluation of multi-target tracking algorithms," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3917–3928, 2020. DOI: 10.1109/TSP.2020.3005309.

[3] C. Carlsson and H. Ekelund, "Sequentially connected 2D assignment problems solved using Lagrangian dual methods," M.Sc. thesis, Chalmers University of Technology, 2022.

[4] V. Nevelius Wernholm and A. Wärnsäter, "Efficient evaluation of target tracking using entropic optimal transport," M.Sc. thesis, Chalmers University of Technology, 2024.

[5] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming*. Springer, 2014.

[6] G. M. Appa, "k-Integrality, an extension of total unimodularity," *Operations Research Letters*, vol. 13, no. 3, pp. 159–163, 1993, ISSN: 0167-6377. DOI: 10.1016/0167-6377(93)90005-2.

[7] G. Appa and B. Kotnyek, "Rational and integral k-regular matrices," *Discrete Mathematics*, vol. 275, no. 1, pp. 1–15, 2004. DOI: https://doi.org/10.1016/S0012-365X(03)00095-5.

[8] R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, no. 5, pp. 275–278, 1958.

[9] A. N. Letchford and A. Lodi, "Strengthening Chvátal–Gomory cuts and Gomory fractional cuts," *Operations Research Letters*, vol. 30, no. 2, pp. 74–82, 2002, ISSN: 0167-6377. DOI: 10.1016/S0167-6377(02)00112-8.

[10] M. Lubin, O. Dowson, J. Dias Garcia, J. Huchette, B. Legat, and J. P. Vielma, "JuMP 1.0: Recent improvements to a modeling language for mathematical optimization," *Mathematical Programming Computation*, 2023. DOI: 10.1007/s12532-023-00239-3.

A

Analysis of possible determinant values

This section gives an overview of the additional determinants of sub-matrix cases not shown in Section 3.1.1.

Recall the structure of a sub-matrix consisting of constraints in regard to the index tuple $(i, j, t, t+1)$:

$$\begin{array}{c} \begin{matrix} w_{ij}^{t} & w_{ij}^{t+1} & g_{ij}^{t} & c_t & c_{t+1} & c_0 \end{matrix} \\ \begin{bmatrix} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \\ 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix}. \end{array} \tag{A.1}$$

As in Section 3.1.1, c_t, c_{t+1}, and c_0 denote some set of columns not actively being considered, but are a part of the constraints. Again, the rows are ordered such that they come in the following order

1. (2.11e),
2. (2.11f),
3. (2.11c) or (2.11d) for t,
4. (2.11c) or (2.11d) for $t+1$.

Below are the combinations of rows explored not presented in Section 3.1.1. While the sub-matrices presented are supposed to be square, only the rows actively examined are displayed for ease of presentation.

Both TGOSPA-constraints

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} g_{ij}^t & w_{ij}^t & w_{ij}^{t+1} & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 0 & 0 & 0\dots & 0\dots & 0\dots \\ 0 & -1 & 1 & 0\dots & 0\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is two times some smaller determinant.

First assignment constraint & first TGOSPA-constraint

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & -1\dots & 0\dots & 0\dots \\ 0 & -1 & 1 & 1\dots & 0\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is one times some smaller determinant.

First assignment constraint & second TGOSPA-constraint

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \\ 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \\ 0 & 1 & 1 & 1\dots & 0\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is one times some smaller determinant.

First assignment constraint & both TGOSPA-constraints

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} g_{ij}^t & w_{ij}^t & w_{ij}^{t+1} & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \\ 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 0 & 0 & 0\dots & 0\dots & 0\dots \\ 0 & 1 & -1 & 0\dots & 0\dots & 0\dots \\ 0 & 0 & 1 & 1\dots & 0\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is two times some smaller determinant.

Second assignment constraint & first TGOSPA-constraint

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 1 & 0\dots & 1\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is one times some smaller determinant.

Second assignment constraint & second TGOSPA-constraint

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} -1 & 0 & 1 & 0\dots & -1\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is one times some smaller determinant.

Second assignment constraint & both TGOSPA-constraints

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} g_{ij}^t & w_{ij}^t & w_{ij}^{t+1} & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 0 & 0 & 0\dots & 0\dots & 0\dots \\ 0 & 1 & -1 & 0\dots & 0\dots & 0\dots \\ 0 & 0 & 1 & 0\dots & 1\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is two times some smaller determinant.

both assignment constraints

The sub-matrix is, by design of the assignment problem constraints, totally unimodular.

both assignment constraints & first TGOSPA-constraint

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} g_{ij}^t & w_{ij}^t & w_{ij}^{t+1} & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} 1 & -1 & 1 & 0\dots & 0\dots & 0\dots \\ 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & -1\dots & 1\dots & 0\dots \\ 0 & 1 & 0 & 1\dots & 0\dots & 0\dots \\ 0 & 0 & 1 & 0\dots & 1\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is one times some smaller determinant.

both assignment constraints & second TGOSPA-constraint

$$\begin{matrix} w_{ij}^t & w_{ij}^{t+1} & g_{ij}^t & c_t & c_{t+1} & c_0 \end{matrix} \quad \begin{matrix} g_{ij}^t & w_{ij}^t & w_{ij}^{t+1} & c_t & c_{t+1} & c_0 \end{matrix}$$
$$\begin{bmatrix} -1 & 1 & 1 & 0\dots & 0\dots & 0\dots \\ 1 & 0 & 0 & 1\dots & 0\dots & 0\dots \\ 0 & 1 & 0 & 0\dots & 1\dots & 0\dots \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 1\dots & -1\dots & 0\dots \\ 0 & 1 & 0 & 1\dots & 0\dots & 0\dots \\ 0 & 0 & 1 & 0\dots & 1\dots & 0\dots \end{bmatrix}$$

Block diagonal matrix structure occurs where the determinant (up to sign) is one times some smaller determinant.

B

Computed fractions and bounds for k

In Table B.1, the computed fractions (or a multiple thereof) found in some optimal solution for specific parameter values are presented. Do note that since these fractions are found from randomly generated data, the list may be incomplete. In Table B.2, the corresponding computed lower bounds for the value of k are presented.

Parameter values	Fractions
(2,2,3)	$\frac{1}{2}$
(2,10,10)	$\frac{1}{2}, \frac{1}{3}$
(3,3,3)	$\frac{1}{2}$
(3,4,3)	$\frac{1}{2}$
(4,4,4)	$\frac{1}{2}$
(4,5,10)	$\frac{1}{2}, \frac{1}{3}$
(5,5,5)	$\frac{1}{2}, \frac{1}{3}$
(5,6,4)	$\frac{1}{2}, \ldots, \frac{1}{4}, \frac{1}{6}$
(6,4,5)	$\frac{1}{2}, \ldots, \frac{1}{5}$
(6,5,5)	$\frac{1}{2}, \ldots, \frac{1}{5}$
(6,6,6)	$\frac{1}{2}, \ldots, \frac{1}{5}, \frac{1}{7}$
(6,7,8)	$\frac{1}{2}, \ldots, \frac{1}{11}$
(7,7,7)	$\frac{1}{2}, \ldots, \frac{1}{11}$
(8,8,8)	$\frac{1}{2}, \ldots, \frac{1}{14}, \frac{1}{16}, \ldots, \frac{1}{17}, \frac{1}{26}$
(9,9,9)	$\frac{1}{2}, \ldots, \frac{1}{27}, \frac{1}{30}, \frac{1}{33}, \ldots, \frac{1}{35}, \frac{1}{37}, \frac{1}{58}, \frac{1}{59}$
(10,3,3)	$\frac{1}{2}, \frac{1}{4}$
(10,5,5)	$\frac{1}{2}, \ldots, \frac{1}{8}$
(10,10,10)	$\frac{1}{2}, \ldots, \frac{1}{40}, \frac{1}{42}, \ldots, \frac{1}{47}, \frac{1}{49}, \frac{1}{52}, \ldots, \frac{1}{55}, \frac{1}{59}, \ldots, \frac{1}{61}, \frac{1}{63}, \ldots, \frac{1}{64}, \frac{1}{67}, \ldots, \frac{1}{69}, \frac{1}{72}, \frac{1}{74}, \ldots, \frac{1}{77}, \frac{1}{79}, \frac{1}{80}$

Table B.1: The computationally found fractions occurring in at least one optimal solution of 100.000 generated configurations for a variety of sets of parameter values.

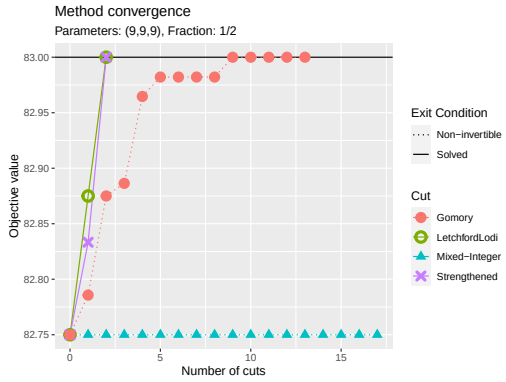
$(T, n_{\mathbf{X}}, n_{\mathbf{Y}})$	k Lower bound
$2 \times 2 \times 3$	2
$2 \times 10 \times 10$	6
$3 \times 3 \times 3$	2
$3 \times 4 \times 3$	2
$4 \times 4 \times 4$	2
$4 \times 5 \times 10$	6
$5 \times 5 \times 5$	6
$5 \times 6 \times 4$	12
$6 \times 4 \times 5$	60
$6 \times 5 \times 5$	60
$6 \times 6 \times 6$	420
$6 \times 7 \times 8$	$2.772 \cdot 10^4$
$7 \times 7 \times 7$	$2.772 \cdot 10^4$
$8 \times 8 \times 8$	$1.225 \cdot 10^7$
$9 \times 9 \times 9$	$5.084 \cdot 10^{15}$
$10 \times 3 \times 3$	12
$10 \times 5 \times 5$	840
$10 \times 10 \times 10$	$1.526 \cdot 10^{29}$

Table B.2: The computed lower bound on k for a variety of sets of parameter values.

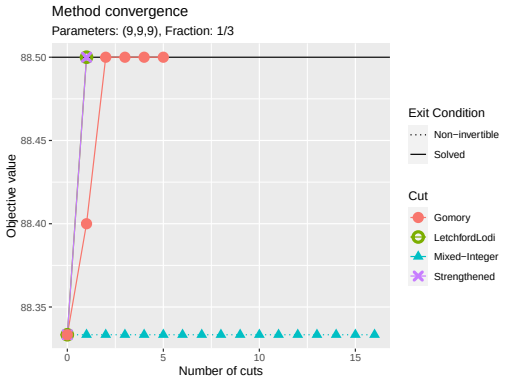
C

Additional numerical results for cutting methods

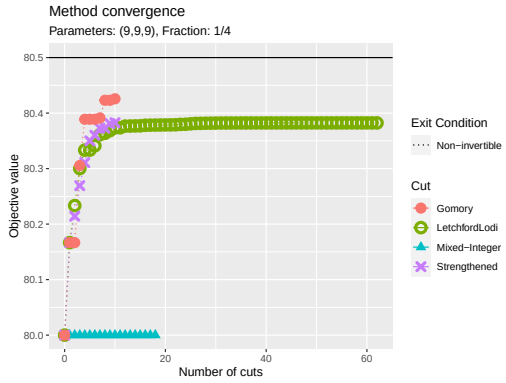
Below are some additional plots and tables from the results of applying the cutting plane methods on different configurations. Additional plots and tables can be generated from the data available on GitHub, `https://github.com/Vesterlund/MVEX03`.



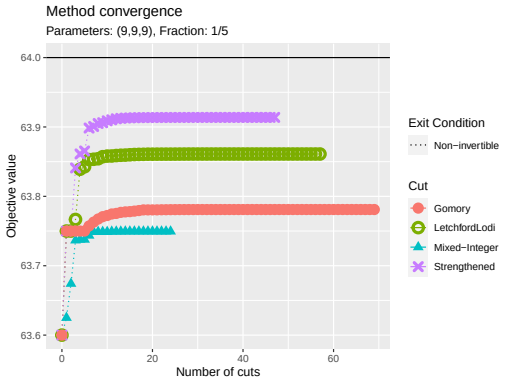
(a) Configuration with fraction $\frac{1}{2}$



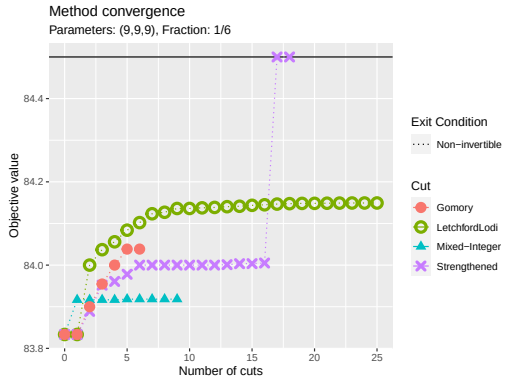
(b) Configuration with fraction $\frac{1}{3}$



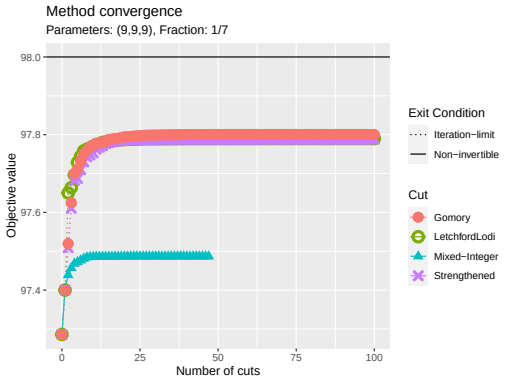
(c) Configuration with fraction $\frac{1}{4}$



(d) Configuration with fraction $\frac{1}{5}$

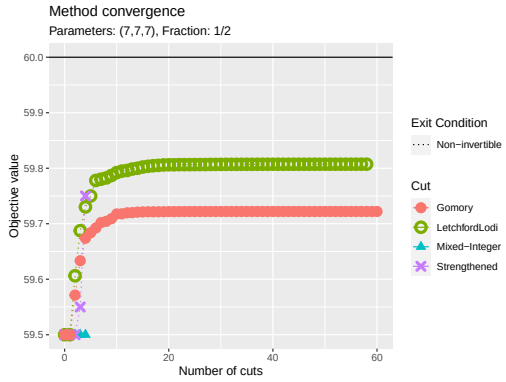


(e) Configuration with fraction $\frac{1}{6}$

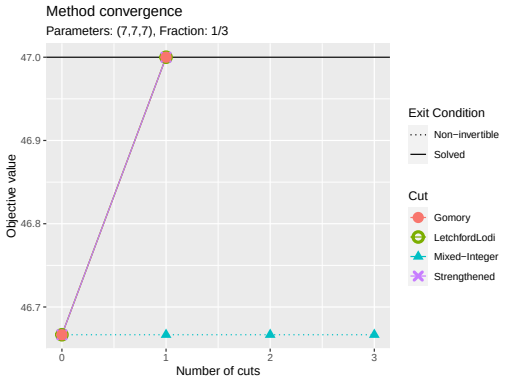


(f) Configuration with fraction $\frac{1}{7}$

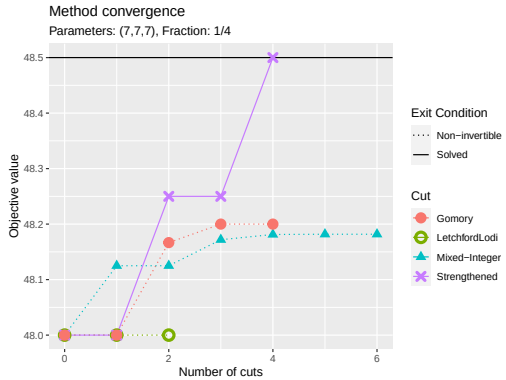
Figure C.1: Plots showing cutting method performance for configurations with parameters $(9, 9, 9)$. The solid black line indicate the optimal value for the binary solution, and the linetype of each cutting plane method denote the reason for termination.



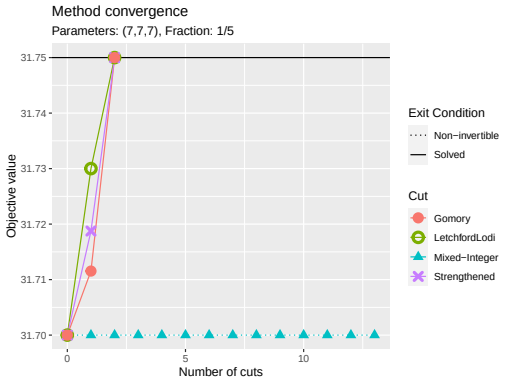
(a) Configuration with fraction $\frac{1}{2}$



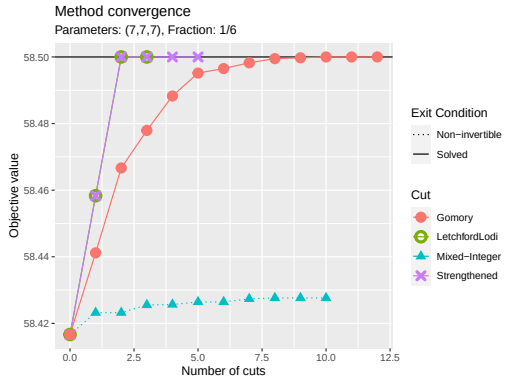
(b) Configuration with fraction $\frac{1}{3}$



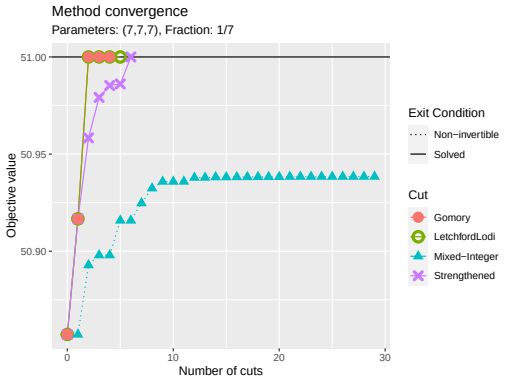
(c) Configuration with fraction $\frac{1}{4}$



(d) Configuration with fraction $\frac{1}{5}$

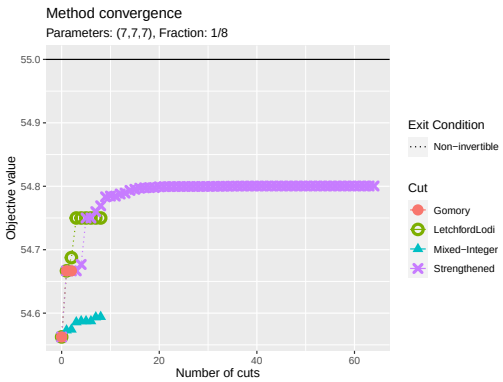


(e) Configuration with fraction $\frac{1}{6}$

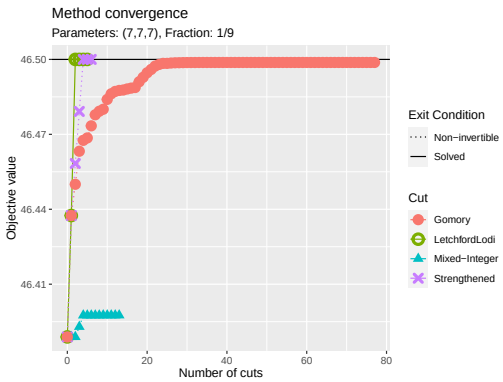


(f) Configuration with fraction $\frac{1}{7}$

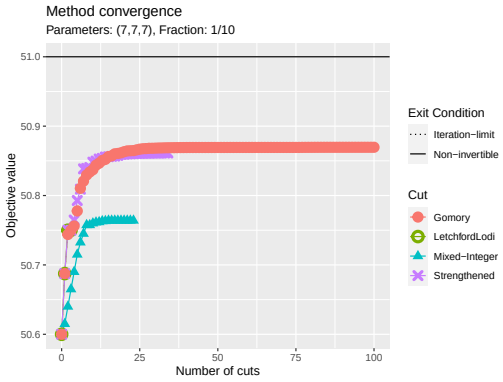
Figure C.2: Plots showing cutting method performance for configurations with parameters $(7, 7, 7)$. The solid black line indicates the optimal value for the binary solution, and the linetype of each cutting plane method denotes the reason for termination.



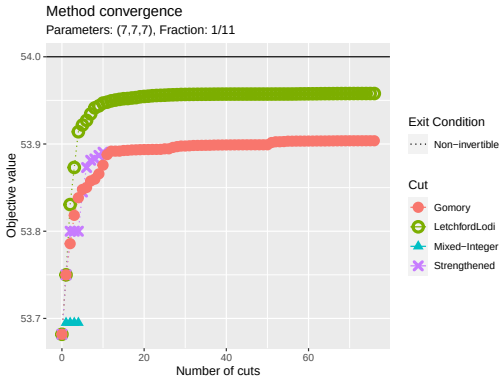
(g) Configuration with fraction $\frac{1}{8}$



(h) Configuration with fraction $\frac{1}{9}$



(i) Configuration with fraction $\frac{1}{10}$



(j) Configuration with fraction $\frac{1}{11}$

Figure C.2: Plots showing cutting method performance for configurations with parameters $(7, 7, 7)$. The solid black line indicates the optimal value for the binary solution, and the linetype of each cutting plane method denotes the reason for termination.

Fraction:	**1/2**	**1/3**	**1/4**	**1/5**	**1/6**	**1/7**	**1/8**	**1/9**	**1/10**
Binary objective	68.50	47.50	64.00	55.00	63.50	65.00	63.50	61.50	54.50
Gomory	68.26 *51**	47.25 *3**	64.00 *6**	55.00 7	63.38 *6**	65.00 9	63.49 *101†*	61.44 *2**	54.50 3
Letchford–Lodi	68.36 *66**	47.49 *12**	64.00 *8**	55.00 2	63.24 *10**	64.86 *2**	63.57 *68**	61.50 1	54.50 1
Mixed-Integer	68.29 *13**	47.42 *25**	63.75 *7**	54.93 *5**	63.47 *16**	64.86 *17**	63.40 *8**	61.44 *26**	54.41 *9**
Strengthened Gomory	68.32 *64**	47.42 *21**	63.88 *4**	55.00 1	63.41 *62**	65.00 2	63.50 4	61.50 2	54.50 1

Table C.1: For the $(8, 8, 8)$ parameters, a number of configurations, characterized by the smallest fraction in an optimal solution to the relaxed problem, the table lists the binary optimal objective value and—for each of the cutting-plane methods—the closest to optimal value found and the corresponding number of cuts added. The markings * and † indicate that—for cases for which the binary optimal value was not reached—the termination criterion used is numerical issues and iteration limit, respectively.

Fraction:	**1/2**	**1/3**	**1/4**	**1/5**	**1/6**	**1/7**	**1/8**	**1/9**	**1/10**
Binary objective	83.00	88.50	80.50	64.00	84.50	98.00	86.00	69.00	81.50
Gomory	83.00 *14**	88.50 5	80.43 *11**	63.78 *70**	84.04 *7**	97.80 *101†*	85.92 *70**	68.98 *54**	81.50 4
Letchford–Lodi	83.00 2	88.50 1	80.38 *63**	63.86 *58**	84.15 *26**	97.79 *101†*	85.83 *5**	69.00 4	81.50 *3**
Mixed-Integer	82.75 *18**	88.33 *17**	80.00 *19**	63.75 *25**	83.92 *10**	97.49 *48**	85.81 *16**	68.72 *2**	81.45 *6**
Strengthened Gomory	83.00 2	88.50 1	80.38 *11**	63.91 *48**	84.50 *19**	97.79 *101†*	85.90 *9**	68.98 *15**	81.62 *49**

Table C.2: For the $(9, 9, 9)$ parameters, a number of configurations, characterized by the smallest fraction in an optimal solution to the relaxed problem, the table lists the binary optimal objective value and—for each of the cutting-plane methods—the closest to optimal value found and the corresponding number of cuts added. The markings * and † indicate that—for cases for which the binary optimal value was not reached—the termination criterion used is numerical issues and iteration limit, respectively.

Fraction:	**1/1**	**1/2**	**1/3**	**1/4**	**1/5**	**1/6**	**1/7**	**1/8**
Binary objective	99.50	100.00	122.00	106.50	93.00	97.50	87.00	91.50
Gomory	101.00 3	100.00 5	122.00 1	106.00 *3**	92.90 *46**	96.90 *44**	86.80 *9**	91.50 2
Letchford–Lodi	99.50 3	100.00 2	122.00 1	106.00 *3**	93.00 4	96.96 *65**	86.80 *59**	91.50 1
Mixed-Integer	99.17 *5**	99.75 *6**	122.00 1	106.00 *5**	92.82 *11**	96.87 *9**	86.93 *28**	91.44 *10**
Strengthened Gomory	99.50 *5**	100.00 1	122.00 1	106.00 *7**	92.88 *42**	96.92 *52**	86.78 *9**	91.50 1

Table C.3: For the $(10, 10, 10)$ parameters, a number of configurations, characterized by the smallest fraction in an optimal solution to the relaxed problem, the table lists the binary optimal objective value and—for each of the cutting-plane methods—the closest to optimal value found and the corresponding number of cuts added. The markings * and † indicate that—for cases for which the binary optimal value was not reached—the termination criterion used is numerical issues and iteration limit, respectively.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY