Creating a Novel Region Finding Approach to Provide Good Solutions to Mixed-Integer Problems

Jamie Robert Owen

BMaths(Decision Science)

Submitted in fulfilment of the requirements for the degree of

Master of Philosophy (Mathematics)

School of Mathematical Sciences

Faculty of Science

Queensland University of Technology

2023

Copyright in Relation to This Thesis

© Copyright 2023 by Jamie Robert Owen. All rights reserved.

Statement of Original Authorship

"The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made."

Signature: _____

Date: _____

Abstract

Mixed Integer Linear Programming (MILPs) are an important classification of optimisation problem with a wide range of applications. However, some instances are particularly hard to solve. Both of the most well-known methods of solving these problems have large drawbacks. Iterative approaches such as branch and cut can scale poorly with problem size, while metaheuristic approaches require extensive setup and tuning as they cannot be applied effectively to MILPs in standard form. In this work we introduce a new solution method for solving MILPs called the Region Method (RM), that works by generating regions that are guaranteed to contain an integer solution (GIRs).

Through this thesis RM is developed and implemented on small test problems and can be seen to generate valid solutions. However, due both to the relaxation approaches applied and the solving tolerances, invalid solutions can be returned. On inspection of these occurrences, RM was found to return a valid GIR contained within the feasible space.

In its current form RM does not present many advantages over current solution techniques. However, current solution techniques are very mature and required extensive research before they were considered useful. For example, the branching and cutting parts of branch and cut were developed independently and individually were considered to be sub-par. There are many areas which have been identified where RM could be developed, including the possible integration of metaheuristic techniques.

A weakness of RM is that it currently relies on a postulate that is only proven up to two dimensions. Further research is required to verify this postulate with mathematical certainty. Additionally, RM converts a MILP into a non-convex, quadratic problem while removing the integer constraints, however this new problem has many useful properties.

It was hoped that a method of this nature would be able to function as a trade-off between branch and cut, and metaheuristics. In its current form this is not the case, however, there appear to be opportunities to combine RM with metaheuristics. We feel that the concept of regions is well suited to meta-heuristics. If this is the case then instead of maintaining a solution to the decision problem and then perturbing that, it may be better to store a region instead.

Overall, we feel that the concept of using a region as a proxy for an integer solution is a valid proposition for a solution technique and should be further researched.

Keywords

Guaranteed Integer Region; MILP; Mixed Integer Linear Programming; Region Method

Acknowledgements

I'd like to acknowledge my supervisors throughout this project, Belinda Spratt, Robert Burdett, and Glen Tian. Over the period of the project, you have provided support through the many difficulties and setbacks that have arisen.

I'd also like to acknowledge the significant additional academic input provided by Harry Bartlett.

It has been a long and circuitous journey to complete this piece of work. Starting with Belinda who originally gave me the opportunity to pursue Honours, and then encouraging my change to Masters. My Masters was originally approved in February 2020 and consisted of investigating scheduling of outpatient appointments in Queensland hospitals. COVID had other plans.

Through this uncertain period, I spent time pursuing a passion project. This passion project eventually grew to my current research when it became apparent that COVID was not going away. Thank you, Belinda, for giving me the confidence to pursue my passion; your mentorship through undergraduate, VRES, Honours and into Masters was fundamental to me. I was distraught to find you were leaving QUT, but I am so happy that you are enjoying working in industry.

Thank you, Robert, for stepping in when Belinda left. My project was still ill-defined at that time and did not fall nicely within anyone's area of research at QUT. Without your support, my project would have died an early death. The journey has taken far longer than anyone expected, and you have provided consistent support throughout.

When it became clear that vital components of the project required additional expertise in both linear algebra and geometry, Harry joined the team. Thank you for giving so generously of your time, Harry, and for your help formalising and communicating the ideas in my head.

Thanks to the entire mathematics department at QUT. The learning environment throughout both my undergraduate and postgraduate degrees was stimulating, supportive and second to none. Thanks to my fellow HDR students for answering my stupid questions and providing a solid sounding board even when neither of us knew what we were talking about.

Robyn Molan, my knight in shining armour, thank you for bailing me out so many times over the years. Without you, and the support of the Disability Services Department, ADHD would have won.

Last but not least, thank you to my family, thank you for supporting me through this endeavour, your unwavering encouragement, even when my deadline slipped multiple times, allowed me to reach the point of writing these acknowledgements. Special thanks to my mother, Jane Howe, who is still directing me and providing support with writing and motivation even now as I write these final words.

Acronyms

Guaranteed integer region (GIR)

Integer Program (IP)

Linear program (LP)

Mixed integer linear program (MILP)

Non-linear program (NLP)

Region method (RM)

Table of Contents

Abstract iii

Keywords iv

Acknowledgements v

Acronyms vi

Table of Contents vii

List of Figures xii

List of Tables xiv

1 Introduction 1

1.1 Motivation 1

1.2 Problem statement 3

1.2.1 Research Questions 3

1.3 Contribution 4

1.4 Thesis Structure 4

2 Background and Literature Review 5

2.1 Introduction 5

2.2 Pre-Processing 6

2.3 Duality 6

2.4 Relaxations and Decompositions 7

2.4.1 LP Relaxation 7

2.4.2 Penalty method 7

2.4.3 Lagrangian Relaxation 8

2.4.4 Augmented Lagrangian Method 9

2.5 Solution Methods – Exact 9

2.5.1 Branch and Bound 9

2.5.2 Cutting Plane Techniques 10

2.6 Metaheuristics 10

2.6.1 Solution Representation 11

2.6.2 Perturbation Techniques 11
2.6.3 Metaheuristic method 12
2.7 Approximation Algorithms 13
2.7.1 Primal-Dual method for Approximation Algorithms 13
2.7.2 Relaxation Enforced Neighbourhood Search (RENS) 14
2.7.3 Feasible Rounding approach 14
2.8 State of the Art 15
3 Overview of Method 15
3.1 Concept Behind RM 15
3.2 Optimising using a GIR 15
3.2.1 Loosening the Bounds 16
3.2.2 Orientation 16
3.2.3 Test for validity – Ordering constraint 16
3.2.4 Testing if a region is GIR 16
3.2.5 Objective function 16
3.2.6 Integer Solution 17
3.2.7 Summary 17
3.3 Implementation of RM 17
3.3.1 Exploring GIRs 17
3.3.2 Implementation of Constraints with Chosen Transformation Matrix 17
3.3.3 Integer Solution 17
3.3.4 Algorithm 17
3.4 Worked Example 18
4 Notation 18
5 GIRs 18
5.1 GIR Theorem 18
5.2 Pivot postulate 19
5.2.1 Proof for 2 dimensions 20

5.3 Generating GIRs 24
6 Theory of RM 25
6.1 Loosening bounds of problem 26
6.1.1 Example 28
6.2 Orientation 29
6.2.1 2D example 30
6.2.2 Optimising orientation 30
6.3 Test for validity – Ordering constraint 32
6.3.1 2D example 34
6.3.2 N-dimensional example 35
6.3.3 Transformed n-cube example 36
6.4 Testing if a region is GIR 37
6.5 Objective function 38
6.6 Integer Solution 39
6.7 Summary 40
6.7.1 Pre-processing 41
6.7.2 Optimising GIR 42
6.7.3 Extracting Integer Solution 42
6.7.4 Complexity Issues 43
7 Implementation of RM 43
7.1 Exploring GIRs 45
7.1.1 Exploring 2D GIRs 45
7.1.2 GIRs in Higher dimensions 49
7.2 Implementation of Constraints with Chosen transformation Matrix 49
7.2.1 GIR constraints 50
7.2.2 Ordering constraints 52
7.3 Integer Solution 53
7.4 Algorithm 53

7.4.1 Pre-processing 54
7.4.2 Optimising GIR - Quadratic 55
7.4.3 Optimising GIR – Linearising 56
7.4.4 Extracting Integer Solution 58
8 Worked Example 59
8.1 Problem statement 59
8.2 Loosening bounds 60
8.3 Reorienting problem 61
8.4 Finding bounds on d 64
8.5 Applying RM 64
8.6 Finding Integer Solution 67
8.6.1 Method 1 68
8.6.2 Method 2 68
9 Results 70
9.1 Implementation 70
9.2 Objective function 70
9.3 The effect of varying the loosening of the bounds (δ) 71
9.4 Correlation between GIR value and Solution value 72
9.4.1 Surface plots 74
9.5 Overview of testing 79
10 Future work 79
10.1 Proof of pivot postulate 79
10.2 Pre-Processing 80
10.3 Further extensions 81
10.3.1 Convex MIPs 81
10.3.2 Metaheuristics 81
10.3.3 Transformation matrices 83
11 Conclusion 83

12 Appendices 85
12.1 Appendix A. Considered IP Models 85
12.1.1 Integer Problems 85
12.1.2 Binary Problems 86
12.1.3 MILPs 86
12.2 Appendix B. Code 88
13 References 88

List of Figures

Figure 2.1 Demonstration of 2-opt 12
Figure 5.1 Convex hull of positively oriented sub-cubes 20
Figure 5.2 Pivot Postulate Case 1 21
Figure 5.3 Pivot Postulate Case 2 23
Figure 5.4 Minimal GIR unverifiable using pivot postulate 25
Figure 6.1 Example of optimising using a GIR 26
Figure 6.2 Example of RM loosening techniques 27
Figure 6.3 Example of RM loosening techniques on binary problem 29
Figure 6.4 The two pairs of test cases for RM in 2D 30
Figure 6.5 Optimal sub-cubes for all orientations in 2-dimensions 31
Figure 6.6 Example of linking GIR vertices to constraints 33
Figure 6.7 Example of GIRs satisfying and violating the GIR ordering constraints 34
Figure 7.1 Examples of minimal GIRs 44
Figure 7.2 The four variants of the vertical minimal GIR with $\boldsymbol{j = 1}$ 46
Figure 7.3 Demonstration of vertex ordering, vertex I closest to constraint before transformation 47
Figure 7.4 Demonstration of vertex ordering, vertex L closest to constraint after transformation 48
Figure 8.1 Graphical representation of Integer Problem 2 59
Figure 8.2 Integer Problem 2 with loosened bounds 60
Figure 8.3 Integer Problem 2 with convex hull of the oriented sub-cubes 61
Figure 8.4 Integer Problem 2, re-oriented so the sub-cubes are now positively oriented 63
Figure 8.5 Integer Problem 2 with the highest valued GIR found by RM 65
Figure 8.6 Integer Problem 2 with the best solution found within the highest valued GIR, and the original constraints shown 67
Figure 9.1 Effect of varying β on the relationship between the assigned GIR Value and the found solution value, for Integer problem 2 71
Figure 9.2 Plot of Optimality Gap Ratio vs GIR Gap Ratio for 500 randomly generated GIRs for Integer Problems 1-4 73
Figure 9.3 Plot of Optimality Gap Ratio vs GIR Gap Ratio for 500 randomly generated GIRs for MILP Problems 1-2 73
Figure 9.4 Plot of Optimality Gap Ratio vs GIR Gap Ratio for 500 randomly generated GIRs for Binary Problems 1 and 3 74
Figure 9.5 GIR Gap Ratio vs $\boldsymbol{d}$ and Optimality Gap Ratio vs $\boldsymbol{d}$ for Integer Problem 3 76
Figure 9.6 GIR Gap Ratio vs $\boldsymbol{d}$ and Optimality Gap Ratio vs $\boldsymbol{d}$ for Integer Problem 1 77

Figure 9.7 GIR Gap Ratio vs $\boldsymbol{d}$ and Optimality Gap Ratio vs $\boldsymbol{d}$ for Integer Problem 2..... 78
Figure 10.1 Process diagram for how RM can be converted to use metaheuristics..... 82

List of Tables

Table 7.1 Transformation Matrices of 2D minimal GIRs, verifiable by pivot postulate 45

Table 9.1 The percentage of 500 randomly generated GIRs, verified using RM, that have a solution that lies outside the feasible region 72

Table 9.2 Summary of applying RM directly to each problem 79

1 Introduction

1.1 Motivation

Mixed Integer Linear Programs or MILPs are an important tool for modelling optimisation problems.

A MILP is a model composed of variables, both integer and real, which are organized into linear constraints and a linear objective function. The integer variables allow models to capture the discrete nature of some decisions [1], whilst the linear constraints and objective function make the model easier to solve.

For example, some decisions, like building a warehouse or purchasing a new machine, can only be taken or not taken. Additionally, many processes have discrete inputs / outputs, for example you can't manufacture half a car, and if you need 2.5 trucks to deliver goods in reality you need 3 trucks.

Industrial problems often contain many decisions of a discrete nature and attempting to remove these elements will often lead to a poor representation of the problem.

There are two common approaches to solving MILPs, the first of these is an iterative approach whereby restrictions are added one by one. This approach is used in methods such as branch and bound, and branch and cut, whereby MILPs are solved by firstly solving their LP relaxations, and subsequently, the solution to the LP relaxation is used to add an additional constraint. This process is then repeated. After solving the LP relaxation there are many different possible constraints that can be added and many of these are contradictory. This creates a branching space of possible constraints that can be added. Prior to further evaluation it is difficult to determine which choice of mutually exclusive constraint will lead to a better solution, and so to guarantee optimality all branches must be evaluated. Additionally, the nested nature of having to decide between mutually exclusive choices creates a large tree which requires a large amount of computation to explore thoroughly.

The second approach to solving MILPs is to use neighbourhood-based methods like metaheuristics. For these approaches it is necessary to define how a given solution can be perturbed to generate a new solution. Any solution that can be reached in a single perturbation is said to lie in a solution's neighbourhood. The full network of solutions is then iteratively explored using an algorithm which corresponds to the neighbourhood-based approach being used. As there can be many combinations of perturbations that can be used to move from any one solution to another, there is more flexibility in exploring the solution space. However, as perturbations only change the solution a small amount it may take many perturbations to move between local optima.

Both these approaches have benefits and drawbacks. While iterative restrictive methods such as branch and cut are guaranteed to provide optimal solutions, they scale poorly with problem size and complexity, which can result in them ranging from impractical to unusable for certain applications. Neighbourhood-based approaches, such as metaheuristic methods, scale better with problem size. A downside to neighbourhood-based approaches is that neighbourhoods must be defined; these can be difficult to formulate, and the way they are formulated can have a large impact on the efficacy of these techniques. Additionally, neighbourhood-based approaches are not guaranteed to reach an optimal solution whereas iterative methods are.

Iterative methods find the best relaxed solution at each step then, by iteratively restricting the decision space, make this solution coincide with the best integer solution. However, iteratively adding constraints like this is equivalent to searching through a binary tree which scales poorly. Although, iterative methods can be fully automated. Neighbourhood-based approaches deal better with the integer nature of the problem however they require manual setup and tuning. There appears to be an opportunity here to combine the automated nature of iterative methods and the better management of integer constraints from neighbourhood-based approaches. A region, known to contain an integer solution, could be treated as a proxy for the integer solution contained within. This region-based approach then potentially combines benefits from both existing approaches.

The efficacy of a region-based approach is dependent on the ability to create and position a suitable region for the given task. As it would be unproductive to try and consider every region and location only a subset of regions is considered. The formulation of the optimal region and location from this subset may be non-linear depending on the subset of regions being considered, however it does not have any integer restrictions. The absence of integer restrictions may transform the nature of the problem so that a different class of solution techniques can be applied.

The regions generated by a region-based approach are similar in nature to the neighbourhoods generated by neighbourhood-based solution techniques. While perturbations used to create neighbourhoods, in the neighbourhood-based approach, are defined by a human's intuition into how a MILP's solutions can be efficiently mapped from one to another, the subset of regions, in the region-based approach, should be chosen based on the geometry of the MILP's feasible space. This geometrical framework, inherent in the region-based approach, allows MILPs in standard form to be automated, while, for MILPs in standard form, neighbourhood-based approaches require manual setup.

Through this thesis a novel region-based approach was developed but in its current form it is unable to compete with current solving techniques. Additionally, due both to the relaxation approaches

applied and the solving tolerances, invalid solutions can be returned. While this approach does rely on a postulate which has not been proven in all dimensions, it has held for all testing done so far. Potential solutions for all these shortcomings have been suggested for future research.

It is hoped that with further development the performance of a region-based approach will scale similarly to that of metaheuristics, while being as automatable as iterative methods.

1.2 Problem statement

This project aims to create a novel, region-based method that is a trade-off between branch and cut, and metaheuristics. This method is developed to be applicable for a wide variety of problem types that can be modelled using MILPs and should require little tuning and should be easy to apply. Additionally, the method is tested using a small number of simple problems to establish functionality and identify opportunities and shortcomings.

We refer to this region-based method as (RM).

1.2.1 Research Questions

Can a new algorithm for solving MILPs be created using regions that are guaranteed to contain an integer point, removing the need for integer constraints?

We describe the type of region mentioned above as a guaranteed integer region (GIR). These are regions that always contain an integer point under any translation.

To answer the proposed research questions the following additional questions are addressed within this thesis:

- What are the properties of a guaranteed integer region, GIR?
- Does there exist an efficient way to generate many diverse GIRs?
- Can solving a MILP be well represented by finding the best GIR that fits in the feasible space?

As GIRs must lie in a problem's feasible space it is necessary to test this. Naively testing vertices of a GIR against a problem's constraints scales exponentially (shown in Section 6.3) with problem size, therefore an alternate method must be found.

- Can new techniques be formulated that expand the search space without introducing additional integer solutions?

As RM relies on fitting GIRs inside the feasible space it is improved by a problem having loose bounds. This differs fundamentally from branch and cut where tighter bounds produce better results. This means new techniques must be formulated to loosen the bounds of the problem.

1.3 Contribution

This work contributes to the existing work by creating a completely new MILP solution technique. This provides many new opportunities for further research including combining existing pre-solving, cutting plane, and metaheuristic techniques with this new technique. With further work it is hoped that this technique will allow metaheuristics to be applied to problems in standard form with minimal setup.

1.4 Thesis Structure

Chapter 1 introduces some of the current approaches to solving MILPs and discusses the motivation behind this research. It also introduces new terminology relevant to this research and highlights additional mathematical techniques that are used throughout the thesis.

Chapter 2 reviews relevant literature, provides background information, and describes the current state of the art.

Chapter 3 provides an overview of the proposed method, RM.

Chapter 4 summarises the notation that is used throughout the document.

Chapter 5 proves certain properties about GIRs and develops a postulate on a more effective method to verify that a region is a GIR.

Chapter 6 develops the theory and process of RM, as well as solving many of the common issues that arise from using a region method.

Chapter 7 combines the approaches discussed in chapters 5 & 6 into the practical implementation of RM. This chapter explores the application of RM and how to implement it efficiently.

Chapter 8 contains a full worked example using RM

Chapter 9 includes numerical testing showing that RM can be used to solve certain MILPs very efficiently. It also provides visualisations of how choices from chapter 7 allow potential future benefits.

Chapter 10 provides direction for future research.

Chapter 11 contains the conclusion to the thesis.

Chapter 12 contains the Appendices.

Chapter 13 contains References.

2 Background and Literature Review

2.1 Introduction

Mixed Integer Linear Programs or MILPs are a type of optimisation problem with both integer and real valued decision variables. A MILP consists of a linear objective function which defines the fitness, cost or profit of a solution and a set of linear constraints that a solution must satisfy. Additionally, a subset of variables is restricted to be integer. All MILPs can be represented in a standard form which is helpful for derivations and proofs, this standard form has the additional constraint that all variables must be positive. Standard form is represented the system of equations below [2]:

Standard form:

$$\begin{aligned} &\max_{x,y} c^T x + h^T y \\ &s.t. Ax + Gy \leq b \\ &\qquad x \geq 0 \\ &\qquad y \geq 0, \qquad y \in \mathbb{Z} \end{aligned}$$

Currently, MILPs are an "Indispensable tool in business and engineering" [3], and are used in a wide variety of industries [4]. Examples include petroleum, manufacturing, chemical, transport and distribution, mining, advertising, and defence. Although, MILPs are a powerful modelling tool if good solutions can be found, this has not always been the case. In 1989 it was noted that MILPs are "theoretically complicated and computationally cumbersome" [5]. Since this remark was made there has been a remarkable improvement in solution speed [6], which has come from both increased computing power and more effective solution techniques. The ubiquitous nature of MILPs results from their simple structure, their ability to model a "wide variety of non-convex and combinatorial problems" [7] and the effectiveness of commercial solvers such as CPLEX and Gurobi. MILPs are an NP-hard problem, meaning that there is no known polynomial-time algorithm to solve any given MILP or even to determine if any given MILP has any feasible solutions. This means that beyond a certain complexity, with current techniques, it will never be viable to exactly solve MILPs. Additionally, the best-known approximate solution techniques are not exclusive to MILPs.

This literature review covers current solution techniques including the importance of pre-processing (pre-solving), relaxations, exact solution methods, metaheuristics, and approximation algorithms. Exact techniques include branch and bound and cutting plane techniques [2]. Whilst the most widely used approximate techniques are metaheuristics.

2.2 Pre-Processing

Pre-processing (or pre-solving) "is a set of routines that remove redundant information and strengthen a given model formulation with the aim of accelerating the subsequent solution process."[8] Pre-processing is a vital part of current MILP solvers including CPLEX and Gurobi; [8] shows that by disabling pre-processing on Gurobi, performance was adversely affected.[6], [9] both show similar results for CPLEX. As MILPs can be used to model a wide variety of problems, different pre-processing techniques most notably cuts have differing levels of impact according to the type of problem being solved and the way a problem is modelled [10]. Another common application of pre-processing techniques is to reduce the dimension of the problem. To the best of my knowledge, all commercial solvers employ branch and cut as their main solution technique. Hence, pre-processing routines are tuned for best performance with branch and cut [11]. As RM functions on a fundamentally different principal to branch and cut, current pre-processing techniques will instead hinder the solution process.

2.3 Duality

Any optimisation problem may be viewed from either of two perspectives, the Primal and the Dual. Due to this duality is fundamental to a number of solution techniques, including Lagrangian relaxation and the Primal-Dual method both of which are discussed in more detail below.

A dual is an alternative formulation of the original (primal) problem with the characteristics that, in the context of maximisation, the optimal solution to the dual problem is an upper-bound on the optimal value for the original problem. This is called weak duality.

For example, the primal problem:

$$\begin{aligned} &\max_{x} c^T x \\ &s.t. Ax \leq b \\ &\qquad x \geq 0 \end{aligned}$$

has the dual:

$$\begin{aligned} &\min_{y} b^T y \\ &s.t. A^T y \geq c \\ &y \geq 0 \end{aligned}$$

In addition, duality has the property that the dual of a dual is the original (primal) problem. For LPs like the problem above, formulating the dual is straight forward however extending this to MILPs has proven largely impractical [12].

2.4 Relaxations and Decompositions

As MILPs increase in size and complexity, it is unlikely that they may be solved in their original form. To address this issue relaxation methods are used. A relaxation of a problem approximates the problem where the feasible domain of the original problem is a subset of the feasible domain for the relaxation. This means that the optimal solution of the relaxation provides an upper bound on the optimal solution of the original problem.[13]

A relaxation of:

$$z = \max\{c(x) : x \in X \subseteq \mathbb{Z}^n\}$$

is

$$z_{\mathbb{R}} = \max\{c_{\mathbb{R}}(x) : x \in X_{\mathbb{R}} \subseteq \mathbb{R}^n\}$$

and has the properties:

$$X_{\mathbb{R}} \supseteq X$$

$$c_{\mathbb{R}}(x) \geq c(x),\ \forall x \in X$$

Two simple ways to relax a MILP are by either relaxing the integer variable domain or by relaxing the problem's constraints. In practice, integer variable relaxations are almost always used, whilst relaxing the problem's constraints is used in a limited set of circumstances, such as with lazy constraints.

2.4.1 LP Relaxation

The most general relaxation for MILPs is the LP relaxation [14]. The LP relaxation is done by changing the domain of integer variables to a real domain. The relaxation problem is now a linear program and can now be more easily solved. However, linear programs can still take a significant amount of time to solve depending on the size and complexity of the problem [14].

2.4.2 Penalty method

Penalty methods cannot be applied to MILPs without modification, they can however be directly applied to LPs, where they treat some constraints as soft constraints hence allowing them to be violated, but any violations incur a penalty. Non-solutions are now classed as solutions but are penalised, so they are sub-optimal. Thus, allowing the replacement of constraints. There are two different approaches known as exterior and interior. These are similar in concept, but the approach taken is reversed. "Exterior penalty methods start at optimal but infeasible points and iterate to feasibility"[15], conversely "Interior penalty methods start at feasible but sub-optimal points and iterate to optimality"[15].

Of the two approaches Exterior penalty methods are the more commonly used due to the ease of including equality constraints and not needing a feasible solution as a start point.[16]

A core component to any type of penalty method is an indicator function which indicates whether a constraint has been violated. A second component is the penalty function which is used to score the severity of any constraint violation. The final component is the scale factor which is used to scale the strength of the penalty function.

Two of the most used penalty functions for Exterior penalty methods are quadratic penalty functions and absolute value penalty functions. Quadratic penalty functions have the benefits of having continuous derivates which is essential for some optimisation methods, but it is only guaranteed to converge to a feasible solution as the scale factor tends to infinity. Absolute value penalty functions have the benefits of not requiring the scale factor to tend to infinity to converge, the scale factor must only exceed a threshold value. However absolute value penalty functions have discontinuous derivatives.[17]–[19]

2.4.3 Lagrangian Relaxation

Lagrangian Relaxation is utilised when there are a small number of constraints that are significantly "harder" than other constraints. To implement this, the "hard" constraints are separated from the rest of the constraints by decomposing the problem. In the subproblem "harder" constraints are incorporated into the objective function with an added multiplier (λ). The master problem then seeks to minimize the subproblem.

A MILP well suited for Lagrangian Relaxation would have one constraint that is the only link between many disjoint sets of variables. In this case, such a constraint would be considered hard due to its relatively higher level of connectivity. This can be easily seen as, if it was removed, the problem could be decomposed into many individual sub problems.

For any given λ, If the original problem is a maximisation problem, the subproblem has an optimal result larger or equal than that of the original problem.

For example, with the original problem of the form:

$$P: \max_{x} c^T x$$

$$s.t. A_1 x \leq b_1$$
$$A_2 x \leq b_2$$

where $A_2 x \leq b_2$ are the "harder" constraints, applying Lagrangian relaxation results in the dual problem:

$$L: \min_{\lambda} D(\lambda)$$

$$s.t. \lambda \geq 0$$

$$D(\lambda): \max_{x} c^T x + \lambda^T (b_2 - A_2 x)$$

$$s.t \ A_1 x \leq b_1$$

The new Lagrangian dual problem L has several useful properties, of note are that $L \geq P$, it must always contain a solution, and it is convex and continuous [20]. To solve this new problem it is best to use an iterative technique such as a sub gradient or interior point method [21].

2.4.4 Augmented Lagrangian Method

The Augmented Lagrangian Method combines both the Penalty method and Lagrangian Relaxation. It does this by adding a Penalty term to the Lagrangian function. The major advantage of this method is by including the Lagrangian multiplier the Penalty Method's scale factor no longer needs to tend to infinity to solve the original problem. This nullifies the main drawback of quadratic penalty method without adding any additional drawbacks. [22]

2.5 Solution Methods – Exact

2.5.1 Branch and Bound

Branch and bound was first developed in 1960[23], and since then it has formed the basis for many direct solution techniques [1], [24]. Due to this, all commercial solvers (CPLEX[4], Gurobi[1],etc) use variations of this technique. There have been massive improvements in both technique and computation power over the years, with a 2.5 billion-fold estimated improvement in the 30 years prior to 2017[6]. This means that a problem that would take 124 years to solve in 1988 would take less than a second to solve in 2017 [6]. Unfortunately, branch and bound remains a solution technique that has worse than polynomial scaling [25]. This means that the method is still prohibitively slow for many modern applications.

Branch and bound solves MILPs by combining two methods. The first is branching. This involves solving the LP relaxation of a problem and then recursively splitting the problem space and adding additional constraints so that each branch eventually either become infeasible or results in a feasible solution. Bounding involves the selective pruning of these branches based on whether the branch cannot contain a solution better than the best-found feasible solution.[26]

2.5.2 Cutting Plane Techniques

The convex hull of a set of points is the smallest convex set containing these points.[27] When a MILP is reduced to a convex hull of its integer solutions, solving the LP relaxation of the MILP is equivalent to solving the MILP. Cutting plane techniques exploit this property by aiming to reduce a problem to this convex hull in the vicinity of the optimal solution [26]. To construct this restricted convex hull, additional constraints called cuts are iteratively added. These cuts are formulated so that no feasible solutions are removed.

The first cutting planes proposed were Gomory cuts [28], although interestingly they were not seen as particularly useful until the early 1990s when they were used in conjunction with branch and bound[29]. Gomory cuts exploit the property that for a problem in standard form all integer variables must be positive, and therefore for any combination of integer variables with integer coefficients must sum to an integer greater than or equal to zero[30]. This is implemented by taking the solution to a LP relaxation in standard form which then outputs a system of equations. For each equation the fractional parts and integer parts can be equated separately. By doing this the integer parts can be removed and replaced with an inequality forming the Gomory cut.

An additional type of cutting plane in common usage is the Gomory-Chvátal cut, this works by tightening each constraint so that they touch at least one integer point. [31]

2.6 Metaheuristics

Metaheuristics are a higher-level heuristic designed to generate a sufficiently good solution to optimisation problems, particularly NP-Hard problems such as those of a combinatorial nature including MILPs [32], as well as other non-convex and non-linear problems. The effectiveness of metaheuristic techniques relies on three elements: the problem's solution representation, the perturbations used to explore the solution space and the metaheuristic method implemented. Of these, only the metaheuristic method can be widely shared between different problems, with simulated annealing being suited to a wide range of problems. However, both solution representation and perturbation techniques are highly problem specific and require significant levels of understanding about both the problem at hand and modelling techniques.

Two concepts key that are common to a number of widely used metaheuristics are the perturbation operators and the neighbourhood. A perturbation operator is an operator that can be applied to a solution to generate an additional solution. The neighbourhood of a solution is the set of solutions that can be found using perturbation operators.

2.6.1 Solution Representation

An example of the importance of solution representation is with combinatorial problems, which due to their nature contain a large feasible region with interlocking constraints. Under these conditions, changing one variable requires a cascade of other changes to keep the solution feasible. Hence solutions are often isolated from each other unless the solution representation addresses this.

To help to prevent the selection of infeasible solutions and to reduce complexity, different solution representation is often employed. One way this can be done is to implement a priority system instead of explicitly defining which nodes are connected to each other. This technique is easily demonstratable with traveling salesman problems.

MILP formulations for Traveling salesman problems often represent solutions in the form:

$$x_{ij} = \begin{cases} 1, & \text{the path goes from city } i \text{ to city } j \\ 0, & \text{otherwise} \end{cases}$$

An example solution is:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

However, if any x_{ij} is changed this results in an invalid solution. If instead the problem is represented by a vector y where City i is visited as the $y_i{}^{\text{th}}$ city in the sequence of cities visited. Then swapping any two members in y results in a valid solution. When represented in the second form it is much easier to verify if a solution is feasible which can allow for much faster solving. This technique is applicable to a wide range of real-world problems, including job shop scheduling and vehicle routing problems [33], [34].

2.6.2 Perturbation Techniques

A key element to the success of metaheuristic techniques, is their use of perturbation operators [35]. A perturbation operator is an operation that is applied to a solution to generate a different solution. However, a perturbation does not have to directly generate a feasible solution; it may instead lead to a constraint violation. Constraint violations caused by perturbations “may be dealt with in one of three ways. They may be allowed but penalised in the objective function. Secondly they may not be allowed at all, or thirdly they may be resolved.” [35]

The best strategy to deal with constraint violations depends on the problem at hand. The application of perturbations should be fast but allow a diverse neighbourhood. Therefore, to reach a good balance of speed and diversity the correct strategy is different for different problems. For example, job shop

problems have hard constraints to satisfy and so work well with constraint resolution [34]. On the other hand, vehicle routing problems generally have constraints that are easier to satisfy and therefore disallowing constraint violations is a sensible option [33].

While perturbations are used to explore diverse parts of the solution space, local area searches are important to explore the immediate vicinity of a solution. Local area search techniques generally consist of enumerating many small changes to find the best local solution.

A good example of this is 2-opt, this technique works by uncrossing solutions. It does this by reversing all the edges between the two selected vertices and then reconnecting the selected vertices back to their new neighbour. K-opt is an abstraction of this method that when k vertices are selected checks all the permutations of crossing between their edges and selects the best. A demonstration of 2-opt can be seen in Figure 2.1. [36]

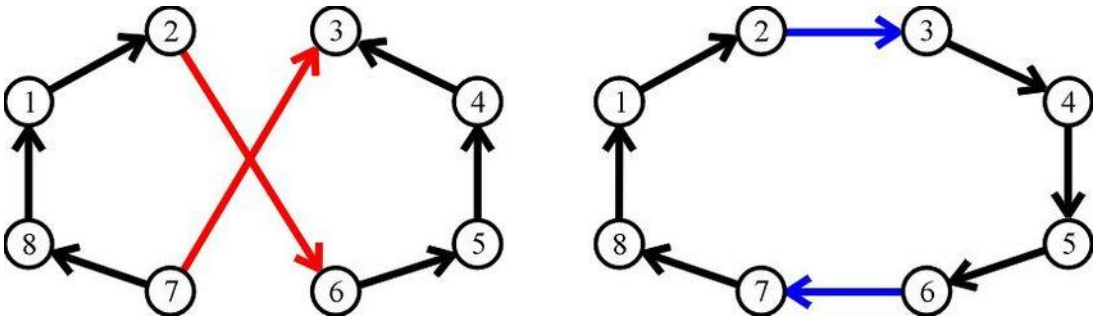

Figure 2.1 Demonstration of 2-opt

In this figure 2-opt has been applied to the vertices 2 and 7 on the left-hand graph, the result of this is shown on the right figure.

2.6.3 Metaheuristic method

The metaheuristic method is the framework that directs the perturbation operators. “Two major components of any metaheuristic algorithm are intensification and diversification” [37], [38]. Diversification is the ability to generate diverse solutions so that the algorithm does not get stuck in local optimal solutions, and intensification is where local solutions are explored if the current solution is good.

There are two main classes of metaheuristic methods, agent based and single solution.

One of the most widely used metaheuristics is Simulated Annealing, which is a single solution metaheuristic, based on the real-world annealing process which is used to strengthen forged metal. A

key aspect of simulated annealing is a heat variable, which decays exponentially mimicking reality. This variable indicates whether the algorithm is trying to intensify or diversify. If the heat is high, this indicates more diversification and vice versa. This process is achieved by perturbing a current solution and if the new solution found is better, choosing it. If the new solution is worse, it still can be chosen if a randomly generated number is smaller than the current heat.

Agent based solutions use interactions between solutions (agents) to help direct the perturbations. Particle swarm is an example of this type of metaheuristic. These methods are often inefficient for higher complexity problems due to the computational burden introduced through agent interaction. Burdens include implementing offspring tournaments for genetic algorithms, and the quadratic overhead of having each agent interact with all others.

2.7 Approximation Algorithms

Approximation algorithms are algorithms that find approximate solutions to optimisation problems with provable guarantees about the relationship between the found solution and the optimal solution [39]. Approximation algorithms for MILPs are mainly based around transforming the MILP into a LP and generally fall into two different approaches, primal-dual and rounding [40]. Additionally, approximation algorithms can be categorised as either, a Constructive Algorithm or an improvement heuristic. Constructive Algorithms do not need a known solution, whereas improvement heuristics take a known solution and aim to improve it.

Most improvement heuristics can be formulated as local search methods, “The Local Search approach generalizes the idea of k-OPT: one defines a neighbourhood of some reference point, determines a point of this neighbourhood which is optimal for some function (e.g. the objective function of the MILP or some feasibility measure), which is then used as a new reference point in the next iteration step.”[41]

2.7.1 Primal-Dual method for Approximation Algorithms

The classical primal-dual method is an algorithm for solving LPs. However, more recently it has been used in combinatorial optimisation, as it “leads to a very general methodology for the design of approximation algorithms for NP-hard problems” [42]. The primal-dual method uses duals to generate approximate feasible solutions. In the context of minimisation, “The key insight about the Primal-Dual approach is that, if you find a feasible solution for the dual problem, then by weak duality, it is the lower bound for the optimal solution of the primal” [43]. The method works by optimising the dual part of the problem until a constraint becomes tight, then setting the corresponding primal part of the problem to be integer and continuing this process every time a constraint becomes tight. When the dual cannot be further improved a full approximate solution has been generated. The advantage of

the primal-dual method is that it "leads to a very general methodology for the design of approximation algorithms for NP-hard problems"[42].

2.7.2 Relaxation Enforced Neighbourhood Search (RENS)

RENS is a Large Neighbourhood Search or (LNS) algorithm. LNS is a variant of local search where instead of iteratively updating the reference point and searching locally, a relatively large neighbourhood is searched only once. RENS constructs a neighbourhood by bounding each integer variable. The bounds are generated by solving the LP relaxation and rounding the solution.[41]

2.7.3 Feasible Rounding approach

The Feasible rounding approach is an approximation algorithm that is applicable to Mixed Integer Convex Problems (MICPs), of which MILPs are a subset. The method requires problems to be granular, a property shared by many MILPs [44]. This approach works by enlarging the inner parallel set of a MILP, which is a region for which any rounded solution is feasible for the original problem [44]. The authors state that "For optimization problems which are granular, solving certain linear problems and rounding their optimal points always leads to feasible points of the original mixed-integer problem. Thus, the resulting feasible rounding approach is deterministic and even efficient."

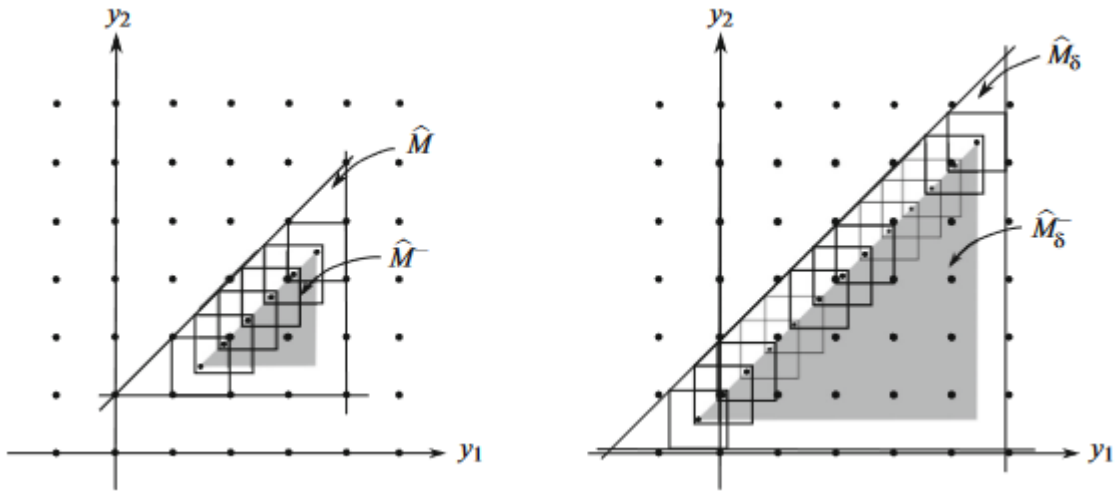

Figure 2.2 Construction of the (enlarged) inner parallel set, using the Feasible Rounding Approach

In Figure 2.2 the figure on the left shows the inner parallel set, $\widehat{M}^-$ for the MILP $\widehat{M}$. The figure on the right shows the same MILP $\widehat{M}_\delta$ and its new inner parallel set, $\widehat{M}_\delta^-$ after Feasible Rounding has been applied.

As seen in Figure 2.2 [44], and will later be seen in Figure 6.3 the feasible rounding approach and RM are similar in trying to utilise the full feasible space including the space known to not contain an integer solution to the MILP.

2.8 State of the Art

The only approach found in the literature that is similar to the Region Method proposed in this Masters thesis, is the Feasible Rounding Approach. The methods are similar in that they both modify the feasible region so that integer constraints are no longer required. However, they differ in their approach to this. Feasible Rounding adds constraints so that non-integer solutions, i.e. real-valued solutions, can always be rounded to give a valid integer solution. By contrast, the Region Method adds additional constraints so that non-integer solutions always have a valid integer solution in their immediate vicinity. It is relevant to note that the authors found that their approach was different from every other approximation algorithm [44]. In October 2021 the authors published the most recent update to this method extending it to allow equality constraints [45].

3 Overview of Method

In this section the concept behind the proposed region method and general details of the method are discussed.

3.1 Concept Behind RM

We propose a new method for the solution of mixed integer programming problems (MILPs). This new method is called the Region Method (RM). RM uses Guaranteed Integer Regions (GIRs) which are regions that are guaranteed to contain at least one integer point.

The general idea for RM is to indirectly solve a MILP by optimising a GIR that lies inside the MILP's feasible space. This is made possible due to the properties of GIRs. These are: firstly, a GIR can be evaluated. Secondly, if a feasible space contains a GIR it also contains an integer solution. Combining these properties means that a GIR will always contain a solution that is bounded by the range of the GIR. Additionally, as a GIR always contains an integer solution, the Integer constraints can be relaxed.

3.2 Optimising using a GIR

For a GIR to be used in RM, firstly we must consider the GIR, its definitions, properties, and how to generate one. Section 5 addresses these points.

For the purposes of this overview, it should be noted that in RM the regions are generated by applying a linear transformation to a unit n-cube and are tested to ensure they are GIRs. The pivot postulate has been developed to test if a region is a GIR, this is addressed in Section 5.2.

Using a GIR as a proxy for an integer point introduces several issues, relating to optimising a region rather than a point.

3.2.1 Loosening the Bounds

A consequence of optimising a region is that you are no longer optimising an object with no volume, therefore it is no longer beneficial to bound the problem as tightly as possible, as this limits the regions that can fit inside the feasible space. For RM it is good to loosen the bounds – this is not the same as relaxing the bounds. We are not allowing additional solutions just taking advantage of the full feasible space. Section 6.1 addresses these points.

3.2.2 Orientation

The GIRs generated in RM can be validated by testing if they conform to the pivot postulate, defined in Section 5. This happens if the region passes at least one in 2^{n-1} criteria where n is the number of integer variables in the MILP. Clearly it quickly becomes computationally infeasible to test all these criteria. To address this, we re-orientate the problem so that only one set of criteria must be tested. This is discussed in Section 6.2.

3.2.3 Test for validity – Ordering constraint

Another issue that arises from optimising a region is ensuring that the region remains inside the feasible space. For convex regions such as those used in RM this means that every vertex of the region must be tested against every constraint. This adds too much complexity so, to reduce complexity, vertices are assigned to a constraint. The vertex must then remain closest to its assigned constraint. Thus, each vertex must only be tested against one constraint. The implementation of this is discussed in Section 6.3.

3.2.4 Testing if a region is GIR

For RM to generate GIRs we must formulate the criterion established in Section 6.2 into a constraint. This is done in Section 6.4.

3.2.5 Objective function

As we are no longer optimising a single point, if we apply the objective function without amendment, we will return a set of values rather than a single value with which to evaluate the solution. While multi-objective problems can be solved, it was found that it was straightforward to calculate metrics to assess the bounds for an integer solution inside the GIR. This is shown in Section 6.5.

3.2.6 Integer Solution

After RM has found a suitable GIR, we must extract the best integer solution contained within. To do this we must construct a MILP that is constrained by the GIR. Section 6.6 shows how these constraints are generated.

3.2.7 Summary

While the steps outlined above have been formed with the intention of allowing RM to generate as many different GIRs as possible. Section 6.7 shows that using RM in this form is still computationally infeasible as the constraints require both a transformation matrix and its adjugate.

3.3 Implementation of RM

As it is not tractable to use RM in the above form it is necessary to investigate ways to reduce the complexity.

3.3.1 Exploring GIRs

Firstly, we investigate GIRs with volume 1, which are referred to as minimal GIRs as they have the smallest possible volume. The investigation starts by looking at all minimal GIRs in two dimensions and is then extended to a subset of minimal GIRs in all higher dimensions. This is addressed in Section 7.1. Through this investigation we found a subset of linear transformations that can both generate the desired minimal GIRs and have an algebraic inverse.

3.3.2 Implementation of Constraints with Chosen Transformation Matrix

Through Section 7.2 we reformulate the constraints generated in Section 6, by substituting the chosen form of transformation matrix found in Section 7.1.

3.3.3 Integer Solution

As we have chosen the form of transformation matrix we can now explicitly calculate the inverse of the transformation matrix allowing the extraction of the integer solution without needing to use slack variables. Giving a second way method for extracting the integer solution.

3.3.4 Algorithm

Using the reformulated constraints from Section 7.2 we can now construct an algorithm for RM. Due to the structure of the chosen transformation matrix we can formulate the Optimising GIR stage two separate ways. The first as a quadratic non-convex problem, which is difficult to solve. The second as a master and sub problem, which are both form LPs. This second form may provide opportunities for future research.

3.4 Worked Example

In Section 8 we walk through a simple problem with only two variables so that we can visually show the method.

4 Notation

A	Matrix
$\vec{b}$	Vector
$A_{i,j}$	Minors of matrix A
$a_{i,j}$	Element of A in the ith row and jth column
$\vec{a}_{i,*}$	Column vector composed of the ith row of A
$\vec{a}_{*,j}$	Column vector composed of the jth column of A
$\sigma(x)$	for vector and matrix arguments, this function is applied element-wise $\sigma(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0 \\ -1, & x < 0 \end{cases}$
$(A)_i$	Row vector composed of the ith row of A
$\vec{b}_i$	ith component of the vector $\vec{b}$
b_i	ith element of the vector $\vec{b}$

5 GIRs

A Guaranteed Integer Region (GIR) is a region which under any translation contains an integer point, where an Integer point is a point for which all coordinates are integers. More formally a GIR is a subset of $\mathbb{R}^n$ that is closed, simply connected and under any translation contains at least one member of $\mathbb{Z}^n$.

5.1 GIR Theorem

A GIR is defined by the following proposition:

Proposition 1: Shape A under an arbitrary shift s always contains an integer point.

We now show that Proposition 1 is equivalent to proposition 2:

Proposition 2: The image of shape A under the functional mapping $F: \mathbb{R}^n \to [0,1)^n$ defined by

$$(x_1, x_2, \dots, x_n) \mapsto (x_1 - \lfloor x_1 \rfloor, x_2 - \lfloor x_2 \rfloor, \dots, x_n - \lfloor x_n \rfloor)$$

covers the entire half-open unit n-cube $[0,1)^n$

Theorem: Propositions 1 and 2 are equivalent.

To prove this, we firstly establish that Proposition 1 implies Proposition 2. Then we establish that Proposition 2 implies Proposition 1.

Assume Proposition 1 is true;

Let $s \in \mathbb{R}^n$ be an arbitrary shift and I_s be the associated integer point in $A + s$

Then $I_s - s \subseteq A, \forall s \in \mathbb{R}^n$

So $F(I_s - s) \subseteq F(A), \forall s \in \mathbb{R}^n$

Then from the definition of F

$$F(I_s - s) = F(-s), \forall s \in \mathbb{R}^n$$

so

$$F(-s) \subseteq F(A), \forall s \in \mathbb{R}^n$$

But, by definition, $\{F(-s) | \ s \in \mathbb{R}^n\}$ is the range of F, that is, the half-open unit n-cube. Thus, we have shown that any shape A that satisfies Proposition 1 must also satisfy Proposition 2.

Now suppose that Proposition 2 Is true, namely that $F(A)$ is the half-open unit n-cube. For any point s in the half-open unit n-cube.

Now, $F^{-1}(s) \in A$ and $F^{-1}(s) - s$ is an integer point in $A - s$. Since this applies for any s in the half-open unit n-cube, any shifted version of A must contain an integer point.

As we have proven that Proposition 1 implies Proposition 2 and the reverse, we have shown that they are equivalent.

5.2 Pivot postulate

Suppose a linear transformation T is applied to a unit n-cube centred on the origin. If the image of this cube under T encloses an n-cube with side length $\frac{1}{2}$ centred on the point α, where $\alpha \in \left\{-\frac{1}{4}, \frac{1}{4}\right\}^n$, then the transformed n-cube is a GIR.

It should be noted that if T encloses the sub-cube centred on α then T is said to have an orientation of $(\sigma(\alpha_1), \sigma(\alpha_2), \sigma(\alpha_3), \dots, \sigma(\alpha_n))$. If this is the vector $\vec{1}$ it is said to have positive orientation.

5.2.1 Proof for 2 dimensions

Without loss of generality assume that the transformation matrix T has a positive orientation. As transforming under T is a linear operation then additionally the sub cube centred on $-\frac{\vec{1}}{4}$ is contained within the transformation.

The convex hull of this shape is shown in Figure 5.1.

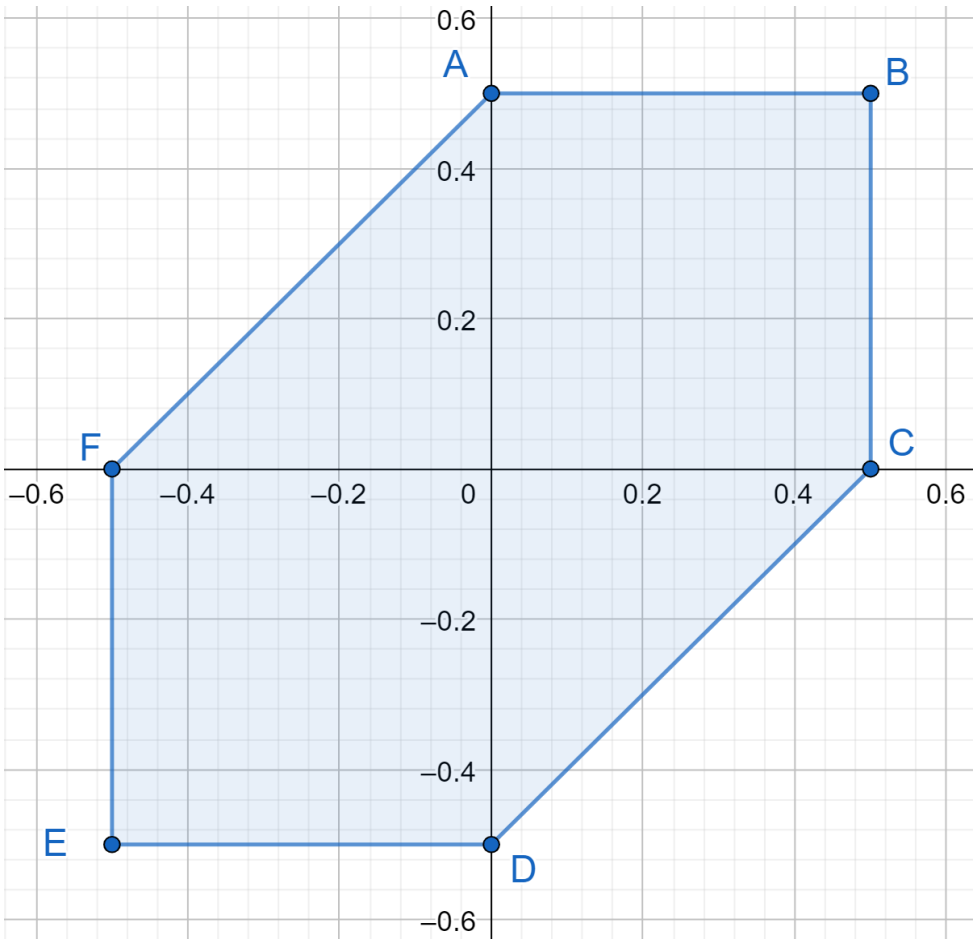

Figure 5.1 Convex hull of positively oriented sub-cubes

We only need to consider transformations that have each edge of the image touching the convex hull of the sub-cubes as the truth of the postulate in this case guarantees its truth in any other case.

Due to the linearity of T, the image of the unit cube under T must be a quadrilateral, which means that the postulate only needs to be proven for the cases when at least two of the vertices A, B, C lie on the edges of the transformed region. Also, due to symmetry only one of A, B and B, C need to be checked.

From the definition of a GIR, Section 5.1, a shape is a GIR when it covers the unit n-cube under the functional mapping (F)

$$F: \mathbb{R}^n \to [0,1)^n$$

$$(x_1, x_2, \ldots, x_n) \mapsto (x_1 - \lfloor x_1 \rfloor, x_2 - \lfloor x_2 \rfloor, \ldots, x_n - \lfloor x_n \rfloor)$$

It should be noted that under this functional mapping $F(I + x) = F(x), \forall I \in \mathbb{Z}$, therefore translation by an Integer amount does not affect whether a shape is a GIR.

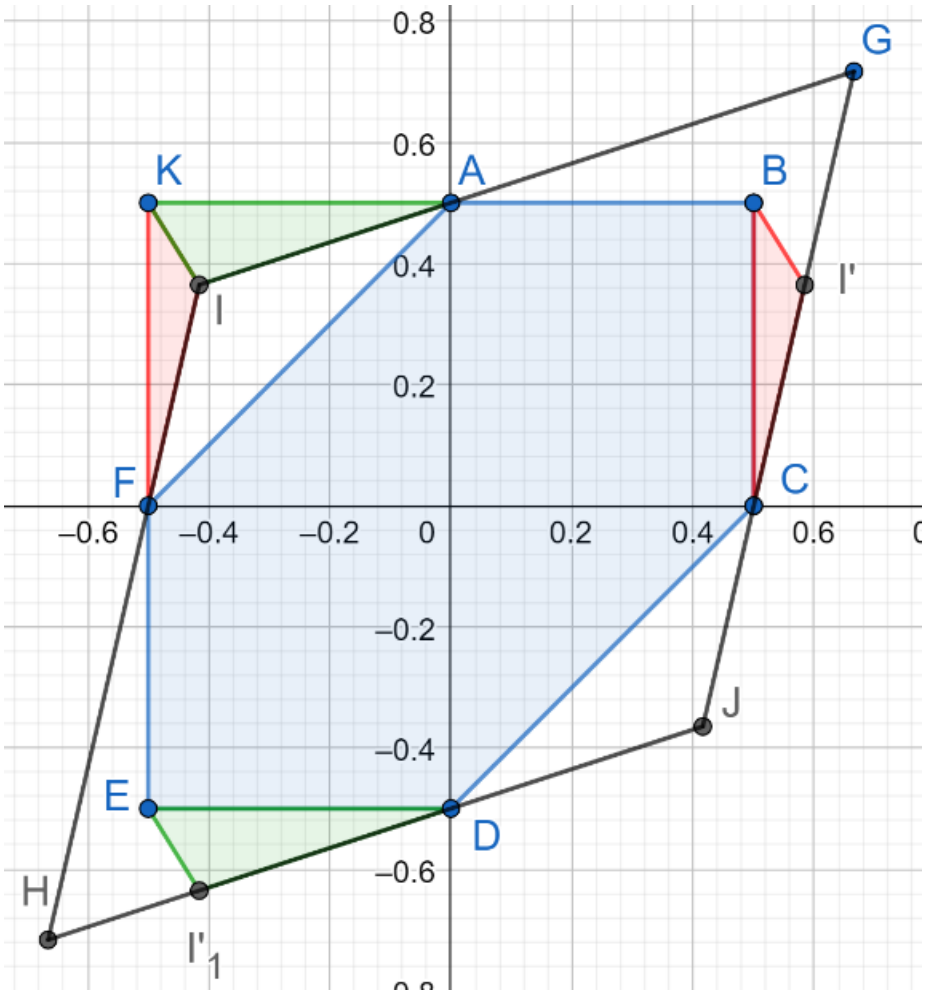

Figure 5.2 Pivot Postulate Case 1

Figure 5.2 shows the case where the vertices A and C are touching the transformed n-cube

Theorem: $GIHJ$ forms a GIR

We prove this by showing that $GIHJ$ satisfies Proposition 2 of Section 5.1.

The vertex K is defined such that it is the vertex in the second quadrant of the unit n-cube centred on the origin.

The vertex I' is defined such that $FII'C$ forms a parallelogram. As F lies on HI and C lies on GJ then by the definition of I', I' must lie on GJ. Therefore, the triangle BCI' lies within the transformed n-cube $GIHJ$.

The vertex I_1' is defined such that $ADI_1'I$ forms a parallelogram. As D lies on HJ and A lies on IG then by the definition I_1', I_1' must lie on HJ. Therefore, the triangle DEI_1' lies within the transformed n-cube $GIHJ$.

From the definition of a parallelogram $\overrightarrow{FI} = \overrightarrow{CI'}$.

From the definition of K and the definition of the convex hull of the sub-cubes $\overrightarrow{BC} = \overrightarrow{FK}$.

Hence the triangle $FIK \equiv CI'B$.

Applying the argument again the triangle $AKI \equiv DEI_1'$.

Then as translation by an integer amount is equivalent to an identity operation under the function mapping F it is clear that $F(CI'B) = FIK$ and $F(DEI_1') = AKI$.

A similar argument clearly applies in the fourth quadrant. Thus, we have shown that $F(GIHJ)$ is the half-open unit cube, and hence that $GIHJ$ forms a GIR.

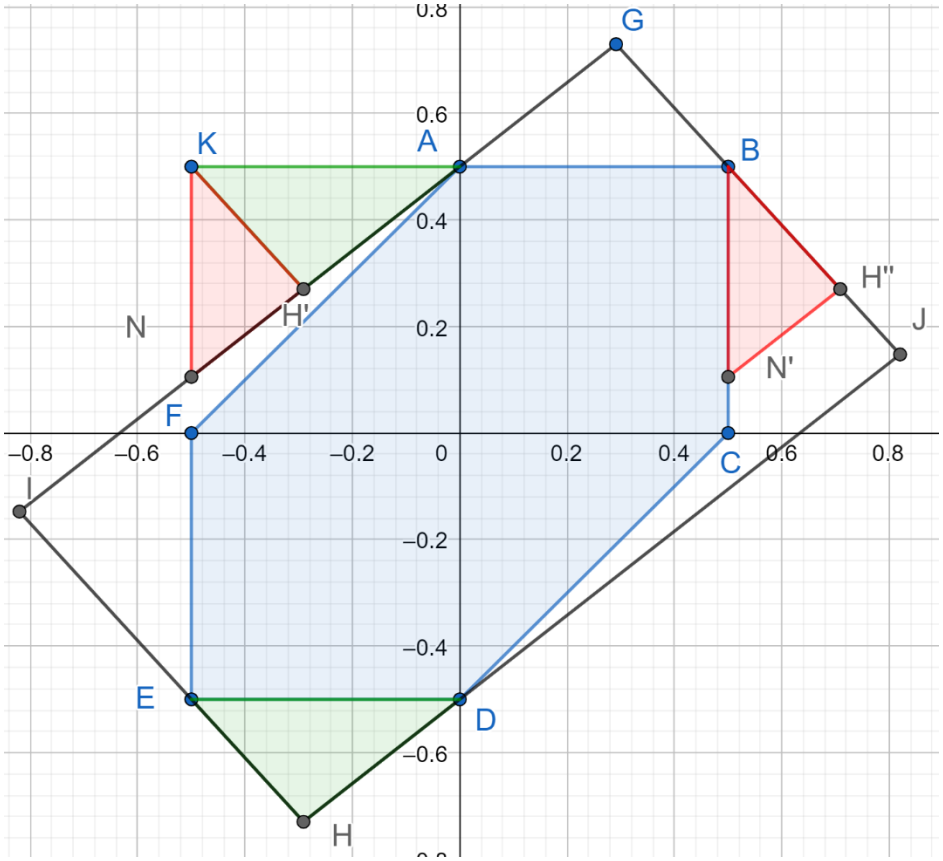

Figure 5.3 Pivot Postulate Case 2

Figure 5.3 shows the case where the vertices A and B are touching the transformed n-cube.

Theorem: $GIHJ$ forms a GIR

We prove this by showing that $GIHJ$ satisfies Proposition 2 of Section 5.1.

The vertex K is defined such that it is the vertex in the second quadrant of the unit n-cube centred on the origin.

Vertex H' is defined such that $AH'K \equiv DHE$

The vertex N is defined as the intersection of GI and KF, the vertex N' is defined by the rectangle $NKBN'$.

The vertex H'' is defined such that $KH'N \equiv BH''N'$

We now need to prove that the triangle $BH''N'$ lies within the transformed n-cube.

As N lies above F, N' lies above C. Therefore, since NH', $N'H''$, and HJ are parallel, H'' lies above or equal to J and $BH''N'$ is indeed within $GIHJ$.

Then as translation by an integer amount is equivalent to an identity operation under the function mapping F it is clear that $F(BH''N') = KH'N$ and $F(DHE) = AH'K$.

A similar argument clearly applies in the fourth quadrant. Thus, we have shown that $F(GIHJ)$ is the half-open unit cube, and hence that $GIHJ$ forms a GIR.

5.3 Generating GIRs

RM generates a simplified representation of a MILP by abstracting the original problem of finding the best integer solution to one of finding the best GIR that lies within the feasible space. For this abstraction to be effective it is important to find a method that can quickly generate GIRs or verify if regions are GIRs.

A notable property of GIRs is that they must have volume of at least one. We describe any GIRs that have volume one as minimal GIRs.

The simplest GIR is the unit n-cube, a minimal GIR. A simple way to generate additional GIRs is to apply a shear transform to this n-cube. It is easily shown that this polytope is also a minimal GIR. However, if an additional shear transform, with a different orientation, is applied to this polytope this new polytope may no longer be a GIR. This observation prompted the development of the pivot postulate, Section 5.2, which appears to hold for higher dimensions based on testing.

An advantage of the pivot postulate is that by representing a GIR as a linear transformation of a n-cube, it is easy to construct a test to verify if this transformation satisfies the postulate, and thus, to determine whether it generates a GIR when applied to an n-cube.

However, pivot postulate has two major flaws that are yet to be fully explored. The first and most obvious of these is that it is only a postulate and has not been proven for higher dimensions. The second is that it is unknown how the subset of GIRs, verifiable using pivot postulate, compare to alternative subsets of GIRs. For example, Figure 5.4 shows a minimal GIR that has been found to work despite not satisfying this postulate. As neither of the sets of sub-cubes fit within the region.

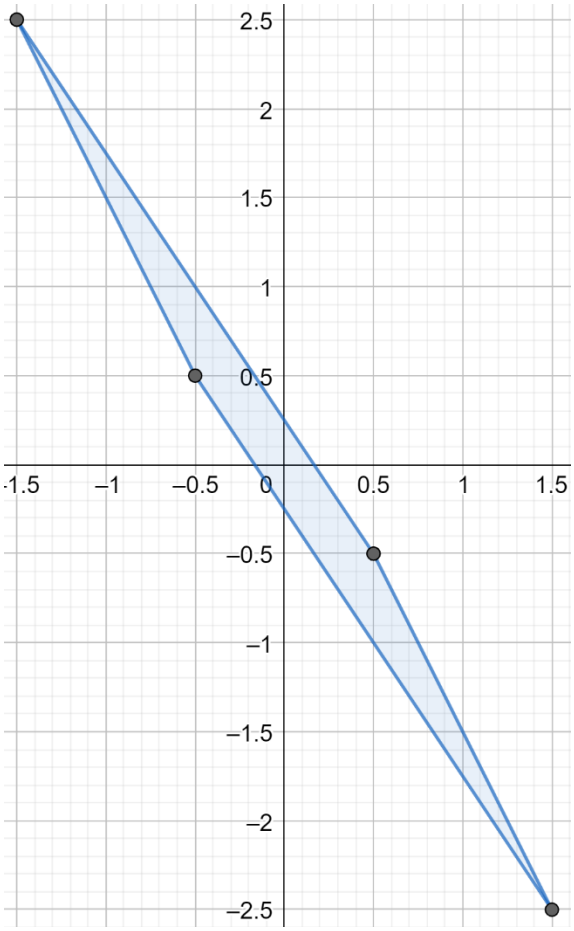

Figure 5.4 Minimal GIR unverifiable using pivot postulate

It can easily be seen from the 2D proof of pivot postulate that many symmetries are involved in the verification of a GIR. Therefore, an initial effort to increase efficiency should aim to address some of these symmetries. This is done in Section 6.2.

6 Theory of RM

This section deals with turning the idea behind RM into a set of constraints and an objective function. To do this we must overcome issues that arise when changing from optimising a single point to optimising a GIR.

We will address this in the subheadings introduced in Section 3.

Unless stated otherwise assume all MILPs have the form

$$Max\ \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

Subject to

$$A\vec{x} + G\vec{y} \leq \vec{b},$$

$$\vec{x} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Throughout Section 6 we will use the following simple problem as a visual tool to aid explanation.

$$\max x + y$$

$$s.t. -2x + 6y \leq 13$$

$$8x - 4y \leq 15$$

$$\vec{x}, \vec{y} \in \mathbb{Z}^n$$

It should be noted that in the figures used we are not solving the problem using RM. The figures show an example of optimising the problem by fitting examples of GIRs into the feasible space. We will use a unit square as the initial GIR.

Figure 6.1 below shows the result of fitting the optimal GIR into the above problem.

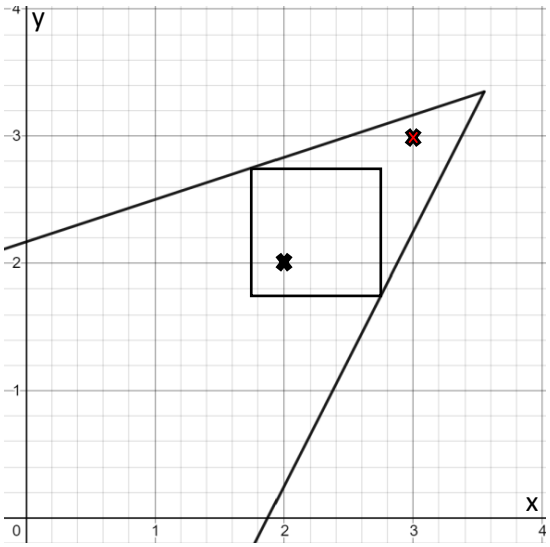

Figure 6.1 Example of optimising using a GIR

In this figure the thick black lines denote the constraints, the thin black lines the region being optimised, the red cross the optimal integer solution, and the black cross the found integer solution.

6.1 Loosening bounds of problem

As we are now optimising a region which has volume this greatly affects the effectiveness of different pre-processing techniques. Like all other Operations Research techniques RM benefits from refining the search space of the problem. Unlike most of these techniques, however, RM does not benefit from

methods such as bound strengthening [8], other than in a reduction in complexity it brings. This is due to RM benefiting from search spaces with a higher volume.

As RM benefits from search spaces with higher volumes it is beneficial if constraints can be loosened without introducing new solutions. Employing these loosening techniques can benefit RM by allowing additional GIRs to be valid. Allowing additional GIRs can allow additional problems to be solved and allow better solutions to already solvable problems. Figure 6.2 below shows how the constraints change after a loosening technique has been applied to a region optimisation problem.

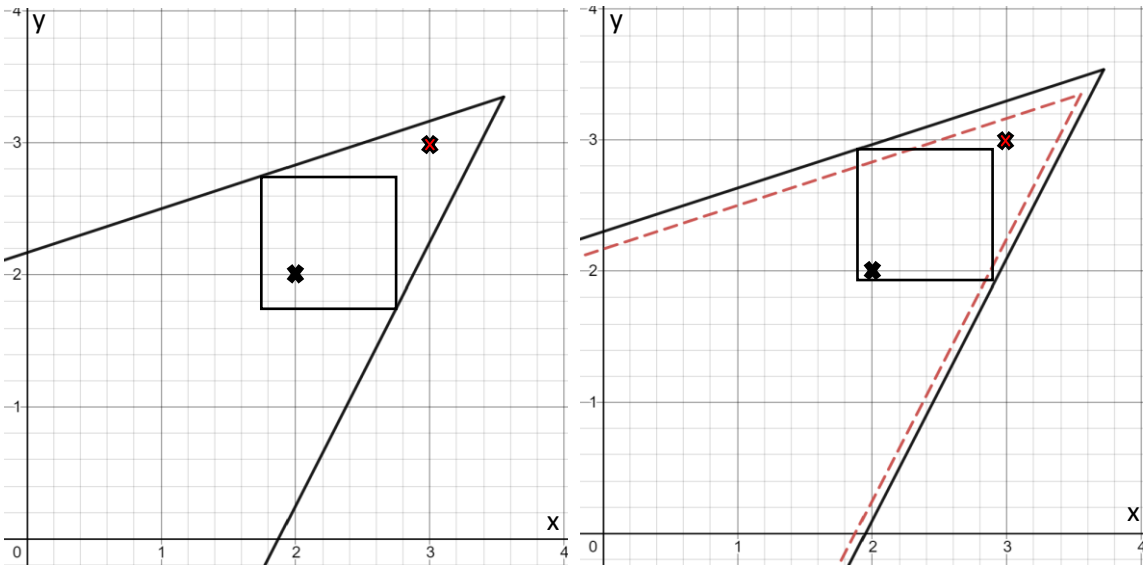

Figure 6.2 Example of RM loosening techniques

In this figure the thick black lines denote the constraints, the thin black lines the region being optimised, the red dotted line the original constraints, the red cross the optimal integer solution, and the black cross the found integer solution.

To loosen the bounds, we can adapt Gomory-Chvátal cuts. A Gomory-Chvátal cut is a cut that can be applied to constraints of the form

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq a_0$$

$$x \in \mathbb{Z}^n$$

If $a_1, \ldots, a_n$ are all integer, then

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq \lfloor a_0 \rfloor$$

is a Gomory-Chvátal cut. If instead the constraint is replaced by

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq \lfloor a_0 \rfloor + 1 - \delta$$

$$\delta > 0$$

this then loosens the constraint without adding any additional solutions. First modifying the constraint to be

$$\frac{(a_1x_1 + a_2x_2 + \cdots + a_nx_n)}{\gcd(a_1, \ldots, a_n)} \leq \frac{a_0}{\gcd(a_1, \ldots, a_n)}$$

ensures that the relevant coefficients are integer; additionally, it provides a loosening that is strong.

6.1.1 Example

An example of how powerful this tool can be is the case of an at most one constraint i.e.

$$x_1 + x_2 + \cdots + x_n \leq 1$$

$$0 \leq x_i \leq 1, \qquad \forall i \in \{1, \ldots, n\}$$

$$x \in \mathbb{Z}^n$$

In its original form the feasible space has volume $\frac{1}{n!}$. As explained in Chapter 5, for RM to work the feasible space must have volume of at least one.

When the previously discussed loosening is applied to this problem its new feasible space becomes

$$x_1 + x_2 + \cdots + x_n \leq 2 - \delta$$

$$\delta - 1 \leq x_i \leq 2 - \delta, \qquad \forall i \in \{1, \ldots, n\}$$

$$\delta > 0$$

This feasible space now has volume

$$\frac{1}{n!}\big(1 + (n+1)(1-\delta)\big)^n$$

It should be noted that for this case, if $\delta \leq 1 - \frac{1}{e} \approx 0.632$ then the relaxed feasible space should always have area greater than one, no matter how many dimensions are used. However, this does not guarantee RM can find a GIR that fits in this region, just that the magnitude of the volume is greater than is needed for a GIR. Figure 6.3, shows the effect of this loosening in two dimensions.

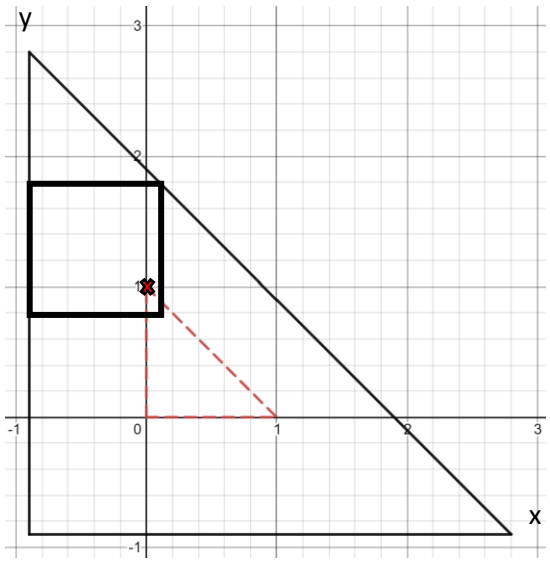

Figure 6.3 Example of RM loosening techniques on binary problem

The red dashed lines in Figure 6.3 represent the original constraints, the solid black lines the new loosened constraints, the thick black lines show a GIR that now fits withing the feasible region, and the red cross the integer solution that lies within the GIR.

6.2 Orientation

As RM requires a region to be a GIR to work as a solution technique, we need to be able to generate GIRs when optimising. As shown in Section 5.2, if a region satisfies the Pivot postulate, then it is a GIR. Pivot postulate works by testing if the region passes at least one in 2^{n-1} criteria where n is the number of integer variables in the MILP. Clearly it quickly becomes computationally infeasible to test all these criteria. To address this, we apply a pre-processing step which re-orientates the problem so that only one set of criteria must be tested.

In this section we use the term "sub-cube" to denote an n-cube of side length 0.5, which has a centre $\vec{x}$ such that $|x_i| = 0.25, \forall i \in \{1, \dots, n\}$

From pivot postulate, Section 5.2, a linear transformation of an n-cube forms a GIR when a suitable set of sub-cubes lie within the transformation.

6.2.1 2D example

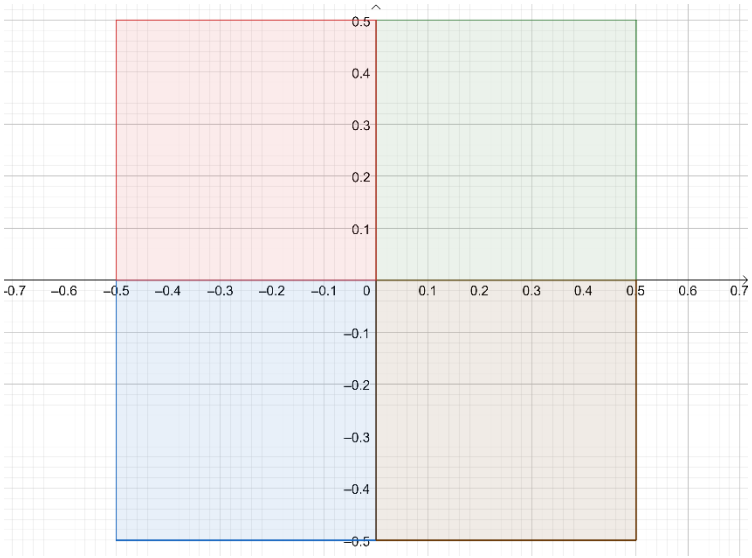

Figure 6.4 The two pairs of test cases for RM in 2D

The different shaded regions show the different sub cubes that lie within the square.

Given that for RM, GIRs are generated by applying linear transformation to n-cubes, all GIRs generated by this method are point symmetric. As can be seen in Figure 6.4, in 2D there are two distinct pairs of sub-cubes that can be tested, this corresponds to the blue-green pair and the red-brown pair. A GIR that covers one or more of these sub-cubes can be said to have an orientation given by the sign of the corresponding sub-cubes' centre. It should be noted that a GIR with orientation of $\vec{1}$ is said to have positive orientation.

It can easily be seen that there are 2^{n-1} pairs of n-cubes that can be chosen for a GIR of degree n. As can be seen in Section 6.3 the problem of optimising a GIR even with a given orientation is a non-convex problem. So, to simultaneously optimise the orientation of the GIR while checking it satisfies said orientation only results in further complexity. To address this, first the orientation is optimised then a region is optimised with the given orientation.

6.2.2 Optimising orientation

As the orientation is optimised separately from the region that uses it, this pre-processing step is an important stage of RM. After this step only GIRs with a positive orientation are considered. This means that RM is highly dependent on the chosen orientation.

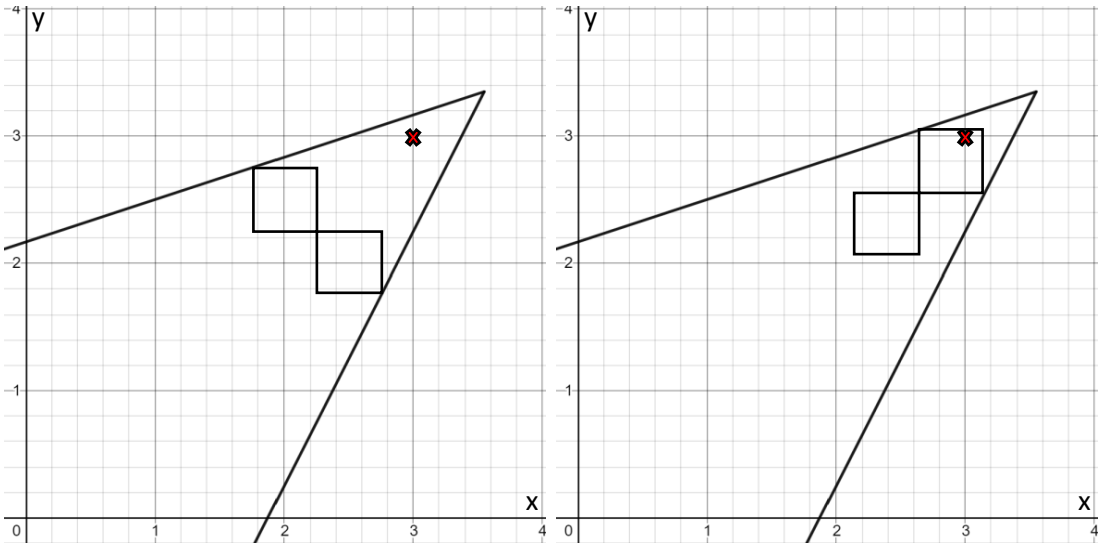

Figure 6.5 Optimal sub-cubes for all orientations in 2-dimensions

In this figure the thick black lines denote the constraints, the thin black lines the pair of sub-cubes that must fit into a region with a given orientation, the red cross the optimal integer solution. The figure on the left has an orientation of $(1,-1)$ while the figure on the right has orientation $(1,1)$, which is referred to as positive orientation.

To summarise, if a GIR has an orientation of $\vec{\lambda}$, it means that the both n-cubes centred at $\frac{\vec{\lambda}}{4}$, and at $-\frac{\vec{\lambda}}{4}$, both with side length $\frac{1}{2}$, fit inside the GIR.

To find the orientation for a MILP we use a heuristic. This heuristic works by optimising the region defined by the required sub-cubes for a given orientation and is evaluated on the midpoint of the region.

So, the heuristic to find an appropriate orientation for a MILP of the form

$$Max\ \vec{c}\cdot\vec{x}+\vec{h}\cdot\vec{y}$$

subject to

$$A\vec{x}+G\vec{y}\leq\vec{b},$$

$$\vec{x}\in\mathbb{Z}^n$$

$$\vec{y}\in\mathbb{R}^n$$

Is found by optimising the LP relaxation of the MILP with the added constraints that the sub-cubes at $\vec{x}+\frac{\vec{\lambda}}{4}$, and at $\vec{x}-\frac{\vec{\lambda}}{4}$, fit inside the feasible space.

This modified problem can be represented as

$$Max\ \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

subject to

$$A\left(\vec{x} + \frac{\vec{\lambda}}{4}\right) + G\vec{y} \leq \vec{b} - \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

$$A\left(\vec{x} - \frac{\vec{\lambda}}{4}\right) + G\vec{y} \leq \vec{b} - \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

$$\vec{x} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^n$$

$$\vec{\lambda} \in \{-1,1\}^n$$

where $|A_{*,j}|$ is the vector of absolute values of the elements in the jth column of the matrix A.

This formulation is found by considering that the location of the vertices of the sub-cubes that lie at

$$\vec{x} + \frac{\vec{\lambda}}{4} \pm \frac{\vec{1}}{4}$$

and

$$\vec{x} - \frac{\vec{\lambda}}{4} \pm \frac{\vec{1}}{4}$$

respectively, by applying the constraint to each of the vertices it is apparent that the constraint is tightest at

$$A\left(\vec{x} + \frac{\vec{\lambda}}{4}\right) + \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

or

$$A\left(\vec{x} - \frac{\vec{\lambda}}{4}\right) + \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

6.3 Test for validity – Ordering constraint

An issue that arises from optimising a region instead of a point, concerns adapting constraints to ensure that the region lies fully within the feasible space. When optimising for a single point the solution must be tested against every constraint, this scales linearly with the number of constraints.

For a solution that is represented by a region, this naively means that every point inside this region must be tested against every constraint.

For any non-empty region this involves testing infinitely many points against each constraint. Clearly a better approach is needed.

For any problem with convex constraints and a convex region, by the maximum principle for convex functions, the maximum of any convex function on this region is attained on the boundary.

This means that only the vertices of the region must be tested for each constraint. This works well for n-spheres, and for simplexes where vertices don't scale exponentially with the number of dimensions. However, this still presents an issue for n-cubes which are used in RM.

To address this issue in RM the vertex closest to each constraint is calculated then constraints are added so that when the n-cube is transformed each constraint remains tightest on its corresponding vertex.

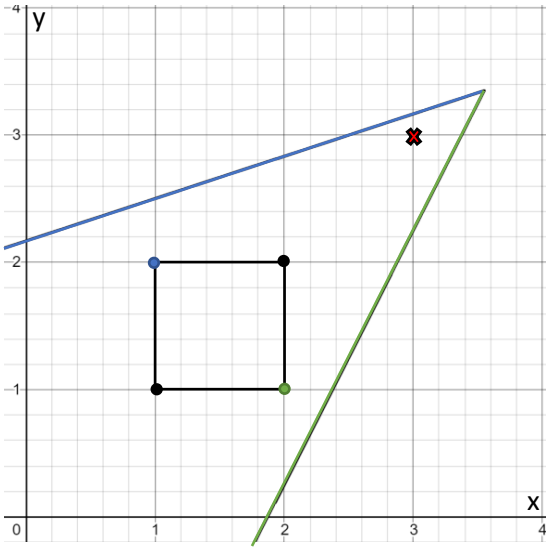

Figure 6.6 Example of linking GIR vertices to constraints

In this figure the blue and the green lines denote the constraints, the blue and green dots represent the vertices closest to their respective constraint, the thin black lines the GIR, the red cross the optimal integer solution.

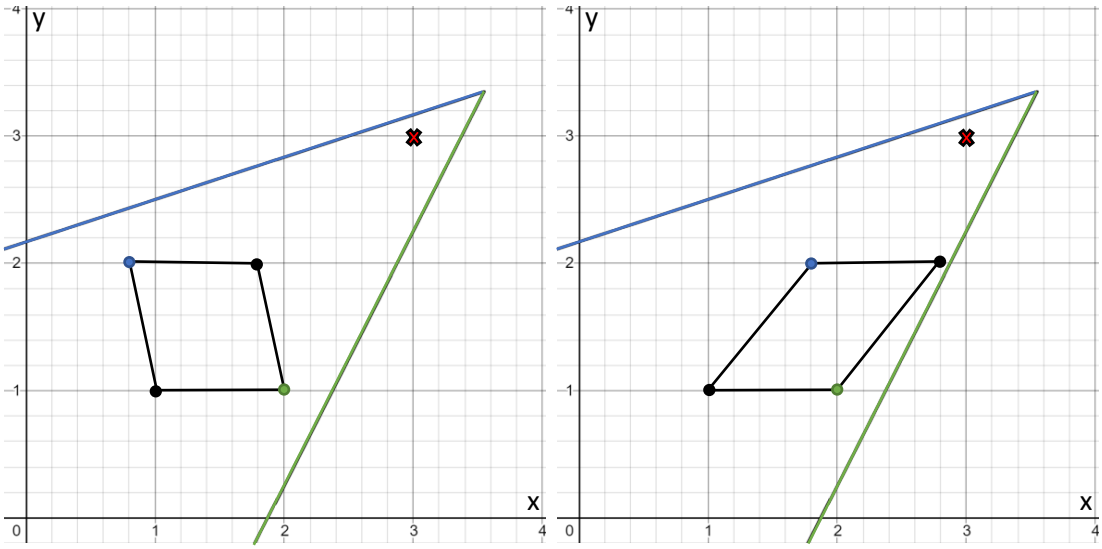

Figure 6.7 Example of GIRs satisfying and violating the GIR ordering constraints

In this figure the blue and the green lines denote the constraints, the blue and green dots represent the vertices linked to their respective constraint, the thin black lines the GIR, the red cross the optimal integer solution. The left figure satisfies the ordering constraints as both coloured vertices are closest to their respective constraints, while the right figure does not.

6.3.1 2D example

Consider a unit square centred on the origin. Its vertices have co-ordinates

$$(x_1, x_2), \qquad x_i \in \left\{-\frac{1}{2}, \frac{1}{2}\right\}, \forall i$$

Suppose this region is bounded by the constraint

$$a_1 x_1 + a_2 x_2 \leq b$$

and let (x_1^*, x_2^*) denote the vertex closest to the constraint. Clearly

$$(x_1^*, x_2^*), \qquad x_i^* \in \left\{-\frac{1}{2}, \frac{1}{2}\right\}, \forall i$$

By definition (x_1^*, x_2^*) satisfy the equation

$$\max_x a_1 x_1 + a_2 x_2 = a_1 x_1^* + a_2 x_2^*$$

By inspection

$$\max_x a_1 x_1 + a_2 x_2 = \left|\frac{a_1}{2}\right| + \left|\frac{a_2}{2}\right|$$

Therefore

$$\left|\frac{a_1}{2}\right| + \left|\frac{a_2}{2}\right| = a_1 x_1^* + a_2 x_2^*$$

This means that

$$x_i^* = \frac{\sigma(a_i)}{2}$$

where $\sigma(x)$ is the sign of x. It should be noted that this holds true when $a_i = 0$ as when this is the case x_i^* can take any value as $a_i x_i = 0$.

In practice, this means that instead of testing every vertex against every constraint, only one vertex must be tested per constraint.

6.3.2 N-dimensional example

This method can also be applied to transformations of unit n-cubes if the closest vertex to each constraint does not change under the transformation.

To use this approach for transformed regions it is useful to first consider the trivial case of no transformation.

Consider a linear, region-based optimisation problem of optimising the position of a unit n-cube, in this case, the individual constraints can be formulated as

$$\vec{a} \cdot (\vec{y} + \vec{\mathrm{x}}) \leq \mathrm{b}, \qquad \{\forall \vec{\mathrm{x}} | \vec{\mathrm{x}} \in \Omega\} \tag{6.1}$$

where $\vec{y} + \vec{x}$ represents any vertex of the unit n-cube centred at $\vec{y}$, $\vec{a}$ the coefficients of the constraint, b the value of the constraint, and Ω the set of all vectors representing the vertices of a unit n-cube centred on the origin:

$$\Omega = \left\{ [\omega_1, \dots, \omega_n] \mid \omega_j \in \left\{-\frac{1}{2}, \frac{1}{2}\right\}, \forall j \in \{1, \dots, n\} \right\}$$

Rearranging (6.1) gives

$$\vec{a} \cdot \vec{\mathrm{x}} \leq \mathrm{b} - \vec{a} \cdot \vec{y}, \qquad \{\forall \vec{\mathrm{x}} | \vec{\mathrm{x}} \in \Omega\}$$

Let $\vec{\mathrm{x}}^* \in \Omega$ represent the tightest vertex of the n-cube against the constraint.

By the definition of $\vec{\mathrm{x}}^*$

$$\vec{a} \cdot \vec{\mathrm{x}} \leq \vec{a} \cdot \vec{\mathrm{x}}^*, \qquad \{\forall \vec{\mathrm{x}} | \vec{\mathrm{x}} \in \Omega\} \tag{6.2}$$

Using the definition of Ω and the reasoning of Section 6.3.1 it can be seen that

$$\vec{\mathrm{x}}^* = \frac{1}{2}\sigma(\vec{a})$$

where σ is now applied element-wise to its argument.

6.3.3 Transformed n-cube example

The approach used in Section 6.3.2 can be extended to include transformations of the unit n-cube. This is done by modifying the constraint in (6.1) to incorporate transformations of the n-cube. The modified constraint is expressed as

$$\vec{a} \cdot (\vec{y} + T\vec{\mathrm{x}}) \leq \mathrm{b}, \qquad \{\forall x | x \in \Omega\} \tag{6.3}$$

where T is the applied transformation matrix.

Rearranging (6.3) gives

$$\vec{a} \cdot T\vec{\mathrm{x}} \leq \mathrm{b} - \vec{a} \cdot \vec{y}, \qquad \{\forall \vec{\mathrm{x}} | \vec{\mathrm{x}} \in \Omega\}$$

Let $\vec{\mathrm{x}}^* \in \Omega$ represent the tightest vertex of the n-cube against the constraint.

By the definition of $\vec{\mathrm{x}}^*$

$$\vec{a} \cdot T\vec{\mathrm{x}} \leq \vec{a} \cdot T\vec{\mathrm{x}}^*, \qquad \{\forall \vec{\mathrm{x}} | \vec{\mathrm{x}} \in \Omega\} \tag{6.4}$$

When T Is the Identity matrix then (6.4) is equivalent to equation (6.2) and hence when this is the case then we again have

$$\vec{\mathrm{x}}^* = \frac{1}{2}\sigma(\vec{a}) \tag{6.5}$$

where σ is again applied element-wise to its argument.

We wish to identify the transformations for which this applies. Substituting (6.5) into (6.4) gives us a condition on these transformations.

$$\vec{a} \cdot T\vec{\mathrm{x}} \leq \vec{a} \cdot \frac{1}{2} T\sigma(\vec{a}), \qquad \{\forall \vec{x} | \vec{x} \in \Omega\} \tag{6.6}$$

Rearranging (6.6) gives:

$$0 \leq \vec{a} \cdot T\left(\frac{1}{2}\sigma(\vec{a}) - \vec{\mathrm{x}}\right), \qquad \{\forall \vec{x} | \vec{x} \in \Omega\}$$

As every element of $\frac{1}{2}\sigma(\vec{a}) - \vec{\mathrm{x}}$ either equals 0 or $\sigma(\vec{a})$ this can be instead simplified to

$$0 \leq \vec{a} \cdot T\vec{g}, \qquad \{\forall \vec{g} | \vec{g} \in \Gamma_{\vec{\mathrm{a}}}\} \tag{6.7}$$

where

$$\Gamma_{\vec{\mathrm{a}}} = \left\{[\gamma_1, \dots, \gamma_n] \mid \gamma_j \in \{0, \sigma(a_j)\}, \forall j \in \{1, \dots, n\}\right\}$$

However due to the linearity of T, condition (6.7) is equivalent to

$$0 \leq \vec{a} \cdot T\sigma(\vec{a}_j), \qquad \forall j \in \{1, \dots, n\} \tag{6.8}$$

where $\vec{a}_j$ is the j-th component vector of $\vec{a}$.

Explicitly evaluating equation (6.8) gives

$$0 \leq \sigma(a_j) \sum_{k \in \{1,\dots,n\}} t_{k,j} a_k \,, \qquad j \in \{1, \dots, n\}$$

It should be noted that this constraint is always satisfied when $a_j = 0$ so this can be rewritten as

$$0 \leq \sigma(a_j) \sum_{k \in \{1,\dots,n\}} t_{k,j} a_k \,, \qquad j \in \{1, \dots, n\}, a_j \neq 0$$

As we no longer need to consider the case where $a_j = 0$ we can divide through by $|a_j|$, giving

$$0 \leq \frac{1}{a_j} \sum_{k \in \{1,\dots,n\}} t_{k,j} a_k \,, \qquad j \in \{1, \dots, n\}, a_j \neq 0 \tag{6.9}$$

Equation (6.9) can be extended to work with a matrix of constraints, A, giving

$$0 \leq \frac{1}{a_{i,j}} \sum_{k \in \{1,\dots,n\}} t_{k,j} a_{i,k} \,, \qquad i \in \{1, \dots, m\}, j \in \{1, \dots, n\}, a_{i,j} \neq 0 \tag{6.10}$$

This implementation however scales poorly, given that a single point-based constraint is transformed into n region-based constraints. Solutions for this are explored in Section 7.2.2.

The constraint to test if this worst vertex lies within the feasible space can then be represented as

$$A_{i,*}\left(\vec{x} + \frac{1}{2} T\sigma\left(A_{i,*}\right)^T\right) + G_{i,*}\vec{y} \leq \vec{b}_i, \qquad i \in \{1, \dots, m\} \tag{6.11}$$

6.4 Testing if a region is GIR

A fundamental part of RM is that the region generated by the transformation matrix T, must be a GIR. To ensure this, constraints must be added to restrict T.

It is important to note that from Section 5.3 that a GIR must have volume of at least one, and the unit n-cube that is being transformed has volume one, hence:

$$|\det(T)| \geq 1 \tag{6.12}$$

and from Section 6.3 the identity matrix, I, always satisfies the validity constraints. The constraint (6.12) does not have a continuous domain, which adds significant complexity. To resolve this the constraint

$$\det(T) \geq 1 \tag{6.13}$$

is introduced. The positive domain is chosen due to I lying within it.

Given that the orientation of a GIR represents the position of an n-cube that lies inside the GIR, it is possible to transform both a GIR and its orientation using reflections. As this is also the case for MILPs it can be assumed, without loss of generality, that the orientation of a problem is positive; a positive orientation means that the n-cube with side length $\frac{1}{2}$ centred on $\frac{\vec{1}}{4}$ fits within the GIR.

If the region generated by the transformation matrix T satisfies the pivot postulate, Section 5.2, and has positive orientation, then the vertices of the n-cube with side length $\frac{1}{2}$ centred on $\frac{\vec{1}}{4}$ fit within the GIR. A way to check this is by applying T^{-1} to the sub-cube and checking if it lies within the original n-cube. The way chosen to do this is to compare every vertex of the inversely transformed sub-cube against the vertices of the original n-cube.

The vertices $\vec{v}$ of the sub-cube are the points satisfying $2\vec{v} \in \Lambda$, where

$$\Lambda = \left\{ [\lambda_1, \dots, \lambda_n] \mid \lambda_j \in \{0,1\}, \forall j \in \{1, \dots, n\} \right\}$$

For these points we therefore require

$$\left| \frac{(T^{-1})_i \vec{v}}{2} \right| \leq \frac{1}{2}, \qquad \forall \mathrm{i} \in \{1, \dots, n\}, \forall \vec{v} \in \Lambda$$

By taking note of constraint (6.13), this can be re-written as

$$\left| \left(\mathrm{adj}(T)\right)_i \vec{v} \right| \leq \det(T) \cdot \vec{1}, \qquad \forall \mathrm{i} \in \{1, \dots, n\}, \forall \vec{v} \in \Lambda \tag{6.14}$$

where $(.)_i$ denotes the i-th row of the matrix $(.)$

It is apparent that the performance of these constraints relies on the structure of T. As both the adjugate and determinant of T are used.

6.5 Objective function

The effectiveness of a GIR generated through RM is determined by the best integer solution that lies inside it. However, as RM works by optimising a GIR as a proxy for the integer solution, directly finding the best integer solution defeats the purpose of using this method. Instead, a simpler evaluation method must be used to judge the effectiveness of a GIR. There are three measures of the GIR that can be easily calculated. These are the maximum and minimum values and the centroid. These values are easily calculable as they are represented by the worst and best vertices according to the objective

function, and the midpoint of the GIR. This is due to the GIRs verified by pivot postulate being linear and therefore symmetric.

Optimising by the centroid of the GIR can be formulated as

$$\vec{c} \cdot \vec{x}$$

Optimising by the minimum or maximum values of the GIR can be formulated as

$$\vec{c} \cdot \vec{x} - \frac{1}{2}\vec{c} \cdot T\sigma(\vec{c})$$

$$\vec{c} \cdot \vec{x} + \frac{1}{2}\vec{c} \cdot T\sigma(\vec{c})$$

respectively. However, when optimising by the minimum or maximum value the constraint

$$\vec{c} \cdot T\sigma(\vec{c}) \geq 0$$

must be added to the problem. This is a vertex ordering constraint based on the formulation from Section 6.3.

6.6 Integer Solution

Once RM has been performed the integer solution must be extracted. This is done by solving the original MILP with added cuts corresponding to the planes of the GIR found through RM.

The equations of the added cuts can be represented by

$$T^{-1}(\vec{y} - \vec{x}) \leq \frac{\vec{1}}{2}$$

$$T^{-1}(\vec{y} - \vec{x}) \geq -\frac{\vec{1}}{2}$$

Where $\vec{y}$ is the integer solution to the MILP.

These can be rearranged to give:

$$\vec{y} - \vec{x} = T\vec{s}$$

$$-\frac{\vec{1}}{2} \leq \vec{s} \leq \frac{\vec{1}}{2}, \qquad \vec{s} \in \mathbb{R}^n$$

Where the slack variable $\vec{s}$ is added to remove the need to calculate T^{-1}.

Additionally, the original constraints must be included

$$A\vec{x} + G\vec{y} \leq \vec{b},$$

$$\vec{x} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^n$$

The objective function for this problem is

$$Max\ \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

For an IP, theoretically, only the cuts are needed to extract the integer solution. However, in practise, due to limits on accuracy, solutions may be found that lie outside of the feasible space. This can be mitigated by including the original constraints as lazy constraints.

Additionally, although rearranging the constraints in this way prevents the need to calculate the inverse transformation matrix, however, for certain transformation matrices, it may be quicker to calculate the inverse and use the constraint in its original form.

The integer extraction MILP has a few important properties. The first is that its feasible space is a subset of the original feasible space, the second is that, due to how the new constraints have been constructed, Integer variables are now very often constrained to a very small set of values. The final property is that the extraction MILP is guaranteed to contain a solution, due to the definition of a GIR.

6.7 Summary

This section summarises the techniques developed throughout Chapter 6 and show RM in its most general form. By doing this, we will show that in this form RM is too complex to implement. However, using the understanding gained in Chapter 6 we will be able to develop RM further in Chapter 7.

RM can be broken down into 3 main stages, these are:

1) Pre-processing:
    - Loosening bounds
    - Orientation
2) Optimise GIR
3) Extract Integer solution

For a MILP of the form

$$\text{Max}\ \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

subject to

$$A\vec{x} + G\vec{y} \leq \vec{b},$$

$$\vec{x} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Where A is of size $m \times n$ and G is of size $m \times n_2$

6.7.1 Pre-processing

We first loosen the bounds of the problem by employing a method like that used in Gomory-Chvátal cuts.

The first step in doing this is to rescale the constraints by dividing each constraint, i, by $\gcd(A_{i,*})$. Let these rescaled constraints be represented by

$$A\vec{x} + G\vec{y} \leq \vec{b}^*$$

This is important as $\vec{b}^*$ is used to extract the integer solution, as while loosening does not allow additional integer solutions, it can allow non-integer variables to take invalid values.

Subsequently, the rescaled set of bounds $\vec{b}^*$ is replaced by the loosened bounds

$$\vec{b} = \lfloor \vec{b}^* \rfloor + \vec{1} - \vec{\delta}$$

where

$$\delta > 0$$

Where $1 - \delta$ is the amount by which the constraint is loosened, and δ can be thought of as the chosen level of tolerance.

We then find the orientation $\vec{\lambda}$ by solving the following MILP

$$\text{Max } \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

Subject to

$$A\left(\vec{x} + \frac{\vec{\lambda}}{4}\right) + G\vec{y} \leq \vec{b} - \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

$$A\left(\vec{x} - \frac{\vec{\lambda}}{4}\right) + G\vec{y} \leq \vec{b} - \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

$$\vec{\lambda} \in \{-1,1\}^n$$

$$\vec{x} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Then we re-orient the problem to the positive orientation by redefining A and $\vec{c}$ to be

$$A_{*,j} = A_{*,j} \cdot \vec{\lambda}_j, \qquad \forall j \in \{1, \dots, n\}$$

$$c_j = c_j \cdot \vec{\lambda}_j, \qquad \forall j \in \{1, \dots, n\}$$

6.7.2 Optimising GIR

If optimisation by the centroid is chosen RM is

$$\text{Max } \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

subject to

$$\det(T) \geq 1$$

$$\left|\left(\text{adj}(T)\right)_i \vec{v}\right| \leq \det(T) \cdot \vec{1}, \qquad \forall \text{i} \in \{1, \dots, n\}, \forall \vec{v} \in \Lambda$$

$$\Lambda = \left\{[\lambda_1, \dots, \lambda_n] \mid \lambda_j \in \{0,1\}, \forall j \in \{1, \dots, n\}\right\}$$

$$0 \leq \frac{1}{a_{i,j}} \sum_{k \in \{1,\dots,n\}} t_{k,j} a_{i,k}\,, \qquad i \in \{1, \dots, m\}, j \in \{1, \dots, n\}, a_{i,j} \neq 0$$

$$A_{i,*}\left(\vec{x} + \frac{1}{2} T\sigma\left(A_{i,*}\right)^T\right) + G_{i,*}\vec{y} \leq \vec{b}_i, \quad \forall i \in \{1, \dots, m\}$$

$$\vec{x} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

where $\vec{x}$ is the centre of the GIR that is generated by the transformation matrix T.

These constraints come from Equations (6.13), (6.14), (6.10),(6.11).

6.7.3 Extracting Integer Solution

To generate the integer solution for the orientationally transformed version of the original problem detonated as $\vec{\gamma}$, the following MILP must be solved

$$\text{Max } \vec{c} \cdot \vec{\gamma} + \vec{h} \cdot \vec{y}$$

$$\vec{\gamma} - \vec{\text{x}} = Ts$$

$$-\frac{\vec{1}}{2} \leq \vec{s} \leq \frac{\vec{1}}{2}, \qquad \vec{s} \in \mathbb{R}^n$$

$$A\vec{\gamma} + G\vec{y} \leq \vec{b}^*$$

$$\vec{\gamma} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Thus, the solution to the original problem is

$$x_j = \gamma_j \cdot \lambda_j, \qquad \forall j \in \{1, \dots, n\}$$

6.7.4 Complexity Issues

Optimising the GIR using the constraints shown above is too complex to be solved in this form. However, in Section 7 it is shown that the constraints can be reduced to a more manageable form, through considered choice of transformation matrix T.

7 Implementation of RM

As can be seen from Section 6.7, the performance of RM is greatly shaped by the properties of the chosen transformation matrix T. Thus, for RM to be an effective solution technique a trade-off must be made between the GIRs that the transformation matrix can form and the speed at which the solution space can be explored.

As there is no simple algebraic form of $adj(T)$ it is prohibitively computationally expensive to implement constraints involving $adj(T)$ when T can be any transformation matrix. To address this, T must be limited to a smaller set of transformation matrices. This smaller set of transformation matrices would ideally remove as few useful transformations as possible whilst removing as many symmetries, and useless cases as possible. To be able to do this it is important to identify useful cases, such as volume minimising cases (minimal GIRs), so that these are preserved. Some examples of these minimal cases are shown in Figure 7.1.

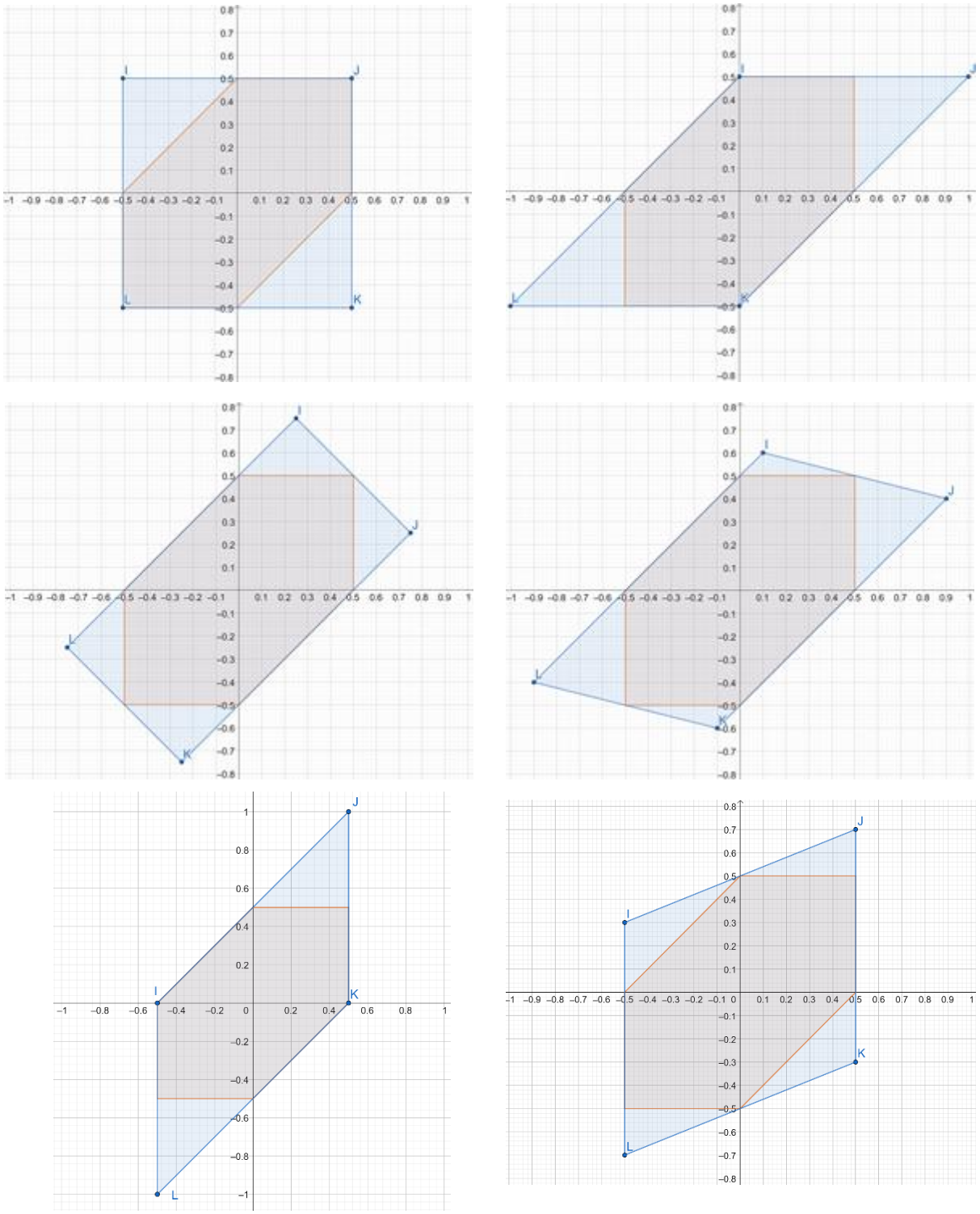

Figure 7.1 Examples of minimal GIRs

The examples shown in Figure 7.1 can be categorised from left to right, top to bottom as: vertical/horizontal $j = 0$, horizontal $j = 1$, diagonal $j = 0.5$, diagonal $j = 0.8$, vertical $j = 1$, vertical $j = 0.4$ with reference to Table 7.1.

7.1 Exploring GIRs

7.1.1 Exploring 2D GIRs

As an initial step to identifying important cases of T it is useful to restrict the problem to its simplest non-trivial form which is in two dimensions. We only consider transformation matrices that have a positive determinant and that generate positively oriented minimal GIRs, verifiable using pivot postulate. The positive determinant constraint was introduced in Section 6.4, the positive orientation restriction was discussed in Section 6.2, pivot postulate is the method of verifying GIRs introduced in Section 5.2.

To find this set of GIRs the following simultaneous equations must be solved.

Pivot postulate with a positive orientation:

$$\left|\left(\mathrm{adj}(T)\right)_i \vec{v}\right| \leq \det(T) \cdot \vec{1}, \qquad \forall \mathrm{i} \in \{1, \dots, n\}, \forall \vec{v} \in \Lambda$$

$$\Lambda = \left\{[\lambda_1, \dots, \lambda_n] \mid \lambda_j \in \{0,1\}, \forall j \in \{1, \dots, n\}\right\}$$

Minimal GIR and positive determinant:

$$\det(T) = 1$$

These can be simplified in two dimensions to

$$\left|t_{i,j}\right| \leq 1$$

$$\left|t_{2,2} - t_{1,2}\right| \leq 1$$

$$\left|t_{1,1} - t_{2,1}\right| \leq 1$$

$$t_{1,1} \cdot t_{2,2} - t_{2,1} \cdot t_{1,2} = 1$$

Solving this set of equations results in 12 different classes of solution, which are shown in Table 7.1, these have been grouped by the shape of GIR formed and the ordering of their vertices, for all these solutions $j \in [0,1]$.

	IJKL	JKLI	KLIJ	LIJK
Vertical	$\begin{pmatrix} 1 & 0 \\ j & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 \\ 1 & -j \end{pmatrix}$	$\begin{pmatrix} -1 & 0 \\ -j & -1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ -1 & j \end{pmatrix}$
Horizontal	$\begin{pmatrix} 1 & j \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} j & -1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} -1 & -j \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} -j & 1 \\ -1 & 0 \end{pmatrix}$
Diagonal	$\begin{pmatrix} j & 1 \\ j-1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & -j \\ 1 & 1-j \end{pmatrix}$	$\begin{pmatrix} -j & -1 \\ 1-j & -1 \end{pmatrix}$	$\begin{pmatrix} -1 & j \\ -1 & j-1 \end{pmatrix}$

Table 7.1 Transformation Matrices of 2D minimal GIRs, verifiable by pivot postulate

Note that in Table 7.1 the columns of the table refer to the relative positions of the vertices. While the rows denote the shapes they make.

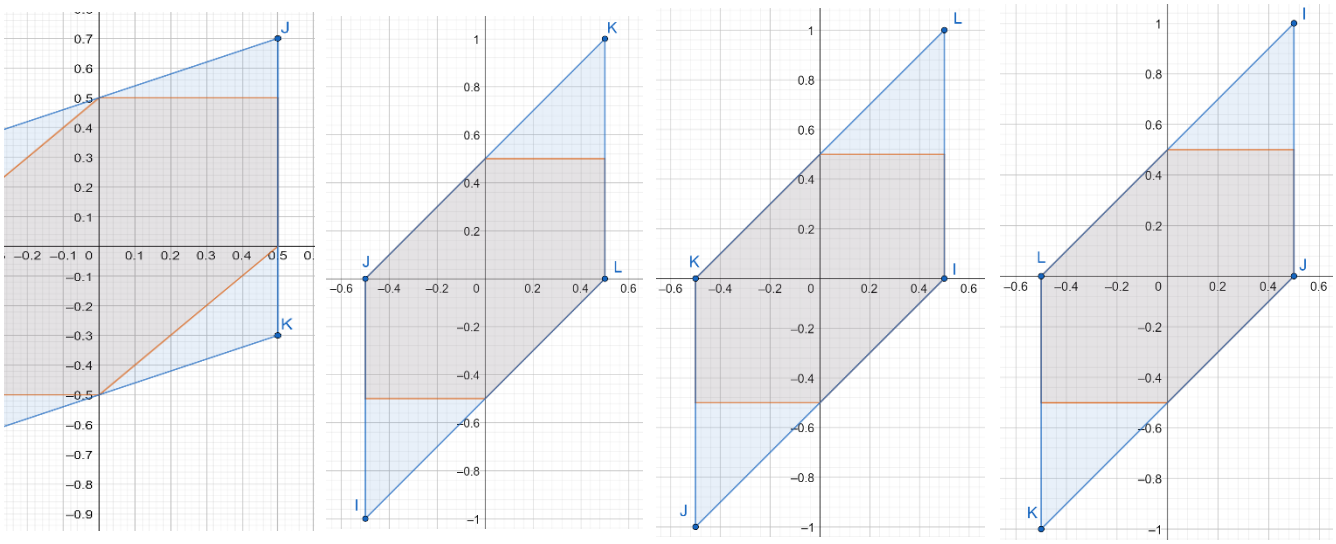

Figure 7.2 The four variants of the vertical minimal GIR with $j = 1$

A parameterisation of T that allows for all the above minimal GIRs to be represented is

$$T = \begin{pmatrix} q_1 + u_1 & u_2 \\ u_1 & q_2 + u_2 \end{pmatrix}$$

where

$$\vec{u}, \vec{q} \in \mathbb{R}^2$$

However, by inspecting Figure 7.2, it can be easily seen that there is rotational symmetry between the vertical case. By the inspection of the matrices, it can also be easily seen that this pattern is repeated for all differing GIR shapes. Using this symmetry T can be re-parameterised as

$$T = \begin{pmatrix} q_1 + u_1 & u_2 \\ u_1 & q_2 + u_2 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}^n, \qquad \forall n \in \{0,1,2,3\}$$

$$\vec{u} \in \mathbb{R}$$

$$\vec{q} \in \mathbb{R}^+$$

From this parameterisation it is easy to remove this rotational symmetry leaving

$$T = \begin{pmatrix} q_1 + u_1 & u_2 \\ u_1 & q_2 + u_2 \end{pmatrix}$$

$$\vec{u} \in \mathbb{R}$$

$$\vec{q} \in \mathbb{R}^+$$

The removal of this rotational symmetry does have wider implications. From the vertex ordering constraints in Section 6.3, the application of T cannot change the relative ordering of the vertices in relation to the constraints. In practice this means that most of the time only one vertex ordering of each type of GIR is valid for any given value of j. Thus, by removing this symmetry some valid GIRs may no longer be generatable, an example of this is shown in the Figures below. Figure 7.3 shows the untransformed n-cube with the vertex I closest to the constraint, shown in green. Figure 7.4 shows that a valid GIR generated by the restricted T violates the vertex ordering constraints as vertex L is now closest to the constraint. While a T that generates an identically shaped GIR exists, which does not violate the vertex ordering constraints, it has been removed by removing the symmetries.

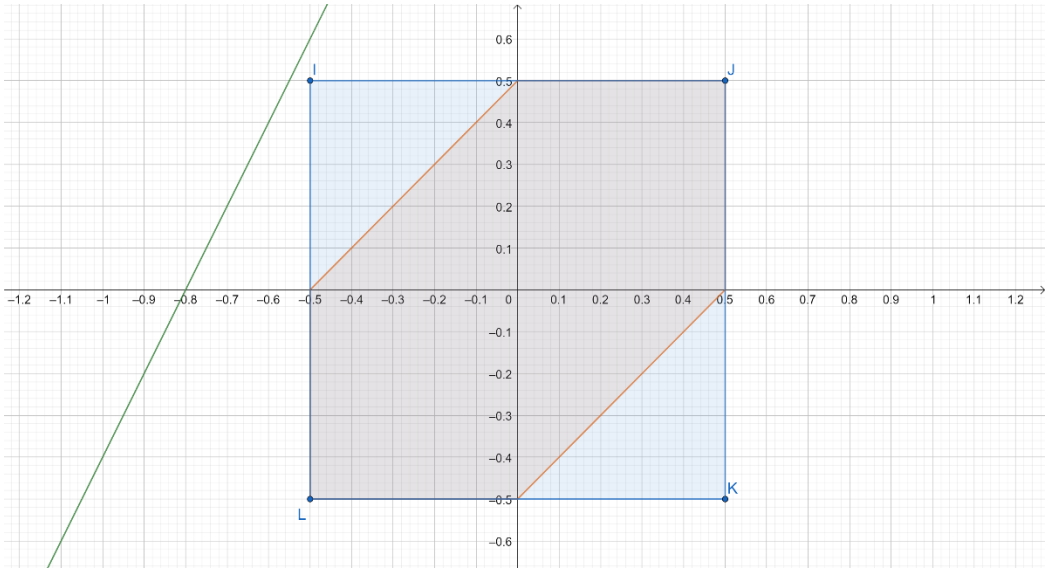

Figure 7.3 Demonstration of vertex ordering, vertex I closest to constraint before transformation

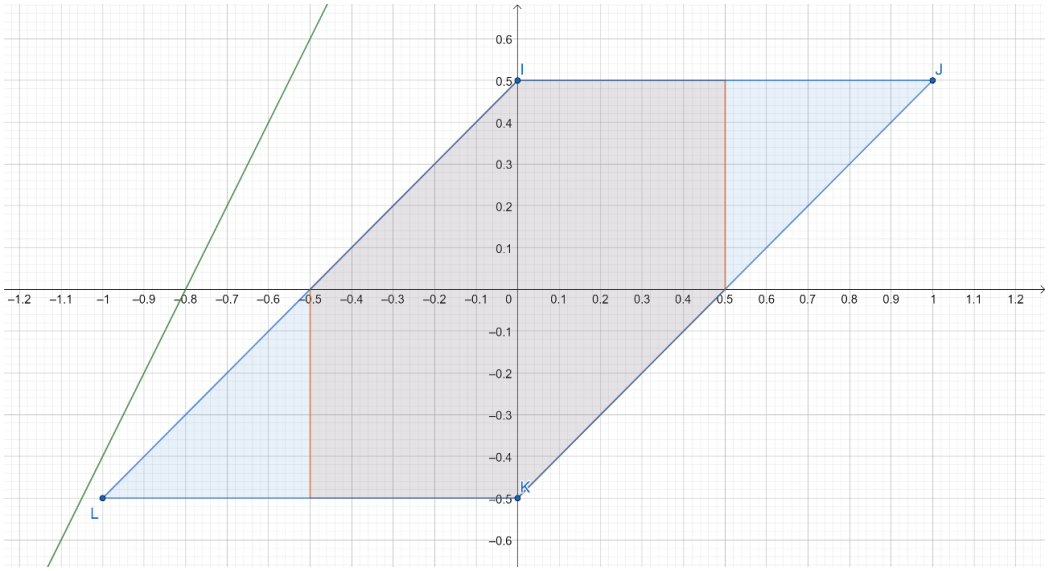

Figure 7.4 Demonstration of vertex ordering, vertex L closest to constraint after transformation

One further modification to the parameterisation of T must occur as in its current form there is an undesirable effect as the ratio $\frac{u_i}{q_i}$ occurs frequently when constraints are generated. To address this issue the substitution $d_i = \frac{u_i}{q_i}$ was used, giving

$$\begin{pmatrix} q_1(1+d_1) & q_2 d_2 \\ q_1 d_1 & q_2(1+d_2) \end{pmatrix}$$

$$\vec{d} \in \mathbb{R}$$

$$\vec{q} \in \mathbb{R}^+$$

A major benefit of this form of the matrix is that it makes $\vec{q}$ only affect the scaling the transformation, this can easily be seen by expressing the matrix as

$$\begin{pmatrix} (1+d_1) & d_2 \\ d_1 & (1+d_2) \end{pmatrix} \begin{pmatrix} q_1 & 0 \\ 0 & q_2 \end{pmatrix}$$

$$\vec{d} \in \mathbb{R}$$

$$\vec{q} \in \mathbb{R}^+$$

Whereas $\vec{d}$ affects both the scaling and the shape. However, there are now two immediate consequences of this. The first is that any constraint that involves a transformation by T is now quadratic. Secondly, the diagonal class of minimal GIR identified above, can only be formed at the limit $\lim_{q_i \to 0} q_i d_i = 1$.

7.1.2 GIRs in Higher dimensions

Now we now have a transformation matrix that works well for generating GIRs in two dimensions, we need to extend it to higher dimensions. To do this only the minimal GIRs that touched a vertex that has only one component are considered. This is the easiest case to consider, there may be many better ways to extend generating GIRs in higher dimensions.

For a minimal GIR to be generated with these conditions, individually, each column of the transformation matrix must be able to generate a 1 on the main diagonal and a 0 elsewhere. Extending the matrix used in the two-dimensional case gives:

$$T = diag(\vec{q}) + \vec{1}\left(\vec{q} \circ \vec{d}\right)^T$$

$$\vec{d} \in \mathbb{R}$$

$$\vec{q} \in \mathbb{R}^+$$

where ∘ is the element-wise product.

In three dimensions this results in

$$\begin{pmatrix} q_1(1+d_1) & q_2 d_2 & q_3 d_3 \\ q_1 d_1 & q_2(1+d_2) & q_3 d_3 \\ q_1 d_1 & q_2 d_2 & q_3(1+d_3) \end{pmatrix}$$

7.2 Implementation of Constraints with Chosen transformation Matrix

As noted in the previous Section the transformation matrix that has been chosen is

$$T = diag(\vec{q}) + \vec{1}\left(\vec{q} \circ \vec{d}\right)^T$$

$$\vec{d} \in \mathbb{R}$$

$$\vec{q} \in \mathbb{R}^+$$

Where ∘ is the element-wise product.

An example of a matrix of this form for three dimensions is

$$\begin{pmatrix} q_1(1+d_1) & q_2 d_2 & q_3 d_3 \\ q_1 d_1 & q_2(1+d_2) & q_3 d_3 \\ q_1 d_1 & q_2 d_2 & q_3(1+d_3) \end{pmatrix}$$

As noted in Section 7.1.1, for this type of matrix $\vec{q}$ acts as a scaling parameter while $\vec{d}$ acts as a shape parameter. An advantage to using a matrix of this form is that the inverse can be explicitly calculated using the Sherman-Morrison formula [46].

$$(A + \vec{u}\vec{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\vec{u}\vec{v}^T A^{-1}}{1 + v^T A^{-1} u}$$

and the determinant can be calculated using the matrix determinant lemma [47]

$$\det(A + \vec{u}\vec{v}^T) = (1 + v^T A^{-1} u)\det(A)$$

7.2.1 GIR constraints

Recall that the following constraints were established in Section 6.4.

$$\det(T) \geq 1 \tag{7.1}$$

$$\left|\left(\text{adj}(T)\right)_i \vec{v}\right| \leq \det(T) \cdot \vec{1}, \qquad \forall \text{i} \in \{1, \dots, n\}, \forall \vec{v} \in \Lambda \tag{7.2}$$

$$\Lambda = \left\{[\lambda_1, \dots, \lambda_n] \mid \lambda_j \in \{1,0\}, \forall j \in \{1, \dots, n\}\right\}$$

It is important to note that any T that satisfies constraint (7.2) forms a GIR, while $|\det(T)| \geq 1$ is a property implicit to all GIRs. Thus constraint (7.1) can be relaxed to

$$\det(T) \geq 0 \tag{7.3}$$

without changing the feasible space.

Explicitly evaluating Equation (7.3) we get

$$\prod_{s=1}^{n} q_s \left(1 + \sum_{j=1}^{n} d_j\right) \geq 0$$

As $\vec{q} \in \mathbb{R}^+$ this constraint can be simplified to

$$\sum_{j=1}^{n} d_j \geq -1 \tag{7.4}$$

by manipulating constraint (7.1) in this way the constraint has been linearised.

Explicitly evaluating Equation (7.2) gives

$$\left|\prod_{s=1}^{n} \frac{q_s}{q_i}\left(v_i\left(1 + \sum_{j=1}^{n} d_j\right) - \sum_{k=1}^{n} d_k v_k\right)\right| \leq \prod_{s=1}^{n} q_s \left(1 + \sum_{j=1}^{n} d_j\right), \qquad \forall \text{i} \in \{1, \dots, n\}, \forall \vec{v} \in \Lambda$$

Using $\vec{q} \in \mathbb{R}^+$ and noting the term $j = i$ disappears in the summation on the LHS

$$\left| v_i + \sum_{j \in \{1,\dots,n\} \setminus i} d_j (v_i - v_j) \right| \leq q_i \left(1 + \sum_{j=1}^{n} d_j \right), \quad \forall i \in \{1, \dots, n\}, \vec{v} \in \Lambda \tag{7.5}$$

It is important to note here that

$$(v_i - v_j) \in \{-1,0,1\}^n$$

So, the maximum value that the lhs can achieve is either

$$1 + \sum_{j \in \{1,\dots,n\} \setminus i} r_j, \quad \forall i \in \{1, \dots, n\}$$

or

$$\sum_{j \in \{1,\dots,n\} \setminus i} s_j, \quad \forall i \in \{1, \dots, n\}$$

where r and s represent the positive and negative components of d respectively

$$r_j = \begin{cases} d_j, & d_j > 0 \\ 0, & d_j \leq 0 \end{cases}$$

$$s_j = \begin{cases} -d_j, & d_j < 0 \\ 0, & d_j \geq 0 \end{cases}$$

By substituting $d_j = r_j - s_j$ into all previous constraints and additionally adding the constraint

$$r_i \geq 0, \quad \forall i \in \{1, \dots, n\}$$

$$s_i \geq 0, \quad \forall i \in \{1, \dots, n\}$$

constraint (7.5) becomes

$$1 + \sum_{j \in \{1,\dots,n\} \setminus i} r_j \leq q_i \left(1 + \sum_{j=1}^{n} r_j - s_j \right), \quad \forall i \in \{1, \dots, n\}$$

$$\sum_{j \in \{1,\dots,n\} \setminus i} s_j \leq q_i \left(1 + \sum_{j=1}^{n} r_j - s_j \right), \quad \forall i \in \{1, \dots, n\}$$

Additionally due to this substitution constraint (7.4) becomes

$$\sum_{j=1}^{n} r_j - s_j \geq -1$$

7.2.2 Ordering constraints

To form the ordering constraints, we start with the following equation from Section 6.3.3

$$0 \leq \frac{1}{a_{i,j}} \sum_{k \in \{1,\dots,n\}} a_{i,k} t_{k,j}\,, \qquad i \in \{1,\dots,m\}, j \in \{1,\dots,n\}, a_{i,j} \neq 0 \tag{7.6}$$

This equation transforms a single points-based constraint into n region-based constraints, where n is the number of integer variables, so by naively implementing this for every constraint it scales poorly. Hence a better method must be created to reduce this overhead.

This equation can be simplified if T shares many common elements in each column. By substituting the transformation matrix $T = diag(\vec{q}) + \vec{1}\left(\vec{q} \circ \vec{d}\right)^T$, defined in Section 7.1.2, into constraint (7.6) the following constraint is found.

$$0 \leq \frac{1}{a_{i,j}} \left(q_j \cdot a_{i,j} + d_j q_j \sum_{k \in \{1,\dots,n\}} a_{i,k} \right), \qquad i \in \{1,\dots,m\}, j \in \{1,\dots,n\}, a_{i,j} \neq 0$$

This can be simplified further by noting that $\vec{q} \in \mathbb{R}^+$, hence this can be simplified to

$$0 \leq q_j \left(1 + \frac{d_j}{a_{i,j}} \sum_{k \in \{1,\dots,n\}} a_{i,k} \right), \qquad i \in \{1,\dots,m\}, j \in \{1,\dots,n\}, a_{i,j} \neq 0$$

This can be further simplified to

$$-1 \leq \frac{d_j}{a_{i,j}} \sum_{k \in \{1,\dots,n\}} a_{i,k}\,, \qquad i \in \{1,\dots,m\}, j \in \{1,\dots,n\}, a_{i,j} \neq 0$$

by noting that only the extrema of the coefficients of d_j need to be checked this can be simplified to the pair of constraints below.

$$-1 \leq d_j \min_{i \in \{1,\dots,m\}} \left(\frac{1}{a_{i,j}} \sum_{k \in \{1,\dots,n\}} a_{i,k} \right), \qquad j \in \{1,\dots,n\}, a_{i,j} \neq 0$$

$$-1 \leq d_j \max_{i \in \{1,\dots,m\}} \left(\frac{1}{a_{i,j}} \sum_{k \in \{1,\dots,n\}} a_{i,k} \right), \qquad j \in \{1,\dots,n\}, a_{i,j} \neq 0$$

It is important to note that the ordering constraints only affect the shape parameter $\vec{d}$.

7.3 Integer Solution

As noted in Section 6.6 for certain cases of T it may be faster to implement the constraints for extracting the integer from the GIR in their original form:

$$T^{-1}(\vec{\gamma} - \vec{\mathrm{x}}) \leq \frac{\vec{1}}{2}$$

$$T^{-1}(\vec{\gamma} - \vec{\mathrm{x}}) \geq -\frac{\vec{1}}{2}$$

where $\vec{\gamma}$ is the integer solution to the problem and $\vec{x}$ is the centre of the GIR.

Due to now having the constraint

$$\det(T) \geq 1$$

We can now rewrite these constraints as

$$\left|\left(\mathrm{adj}(T)\right)_i(\vec{\gamma} - \vec{\mathrm{x}})\right| \leq \det(T) \cdot \frac{\vec{1}}{2}, \qquad \forall \mathrm{i} \in \{1, \dots, n\}, \forall \vec{v} \in \Lambda$$

This takes a form that is very similar to Equation (7.2) and then can therefore be simplified to

$$\left|\gamma_i - x_i + \sum_{j \in \{1,\dots,n\} \setminus i} d_j(\gamma_i - x_i + x_j - \gamma_j)\right| \leq \frac{q_i}{2}\left(1 + \sum_{j=1}^{n} d_j\right), \quad \forall i \in \{1, \dots, n\}, \vec{\mathrm{v}} \in \Lambda$$

which is similar in form to Equation (7.5).

7.4 Algorithm

RM can be broken down into 3 main stages, these are:

1) Pre-processing:
   - Loosening bounds
   - Orientation
2) Optimise GIR
3) Extract Integer solution

After restricting the transformation matrix introduced in Section 7.1, the method listed in Section 6.7 can now be adapted.

For a MILP of the form

$$\mathrm{Max}\ \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

subject to

$$A\vec{x} + G\vec{y} \leq \vec{b},$$

$$\vec{x} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Where A is of size $m \times n$ and G is of size $m \times n_2$

With a restricted transformation matrix

$$T = diag(\vec{q}) + \vec{1}\left(\vec{q} \circ \vec{d}\right)^T$$

$$\vec{d} \in \mathbb{R}$$

$$\vec{q} \in \mathbb{R}^+$$

7.4.1 Pre-processing

Bound loosening:

The first step in doing this is to rescale the constraints by dividing each constraint, i, by $\gcd\left(A_{i,*}\right)$. Let these rescaled constraints be represented by

$$A\vec{x} + G\vec{y} \leq \vec{b}^*$$

This is important as $\vec{b}^*$ is used to extract the integer solution. While loosening does not allow additional integer solutions, it can allow any non-integer variables to take invalid values.

Subsequently, the rescaled set of bounds $\vec{b}^*$ is replaced by the loosened bounds

$$\vec{b} = \lfloor \vec{b}^* \rfloor + \vec{1} - \vec{\delta}$$

where

$$\delta > 0$$

Where $1 - \delta$ is the amount by which the constraint is loosened, and δ can be thought of as the chosen level of tolerance.

Orientation:

We then find the orientation $\vec{\lambda}$ by solving the following MILP

$$\max \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

Subject to

$$A\left(\vec{x}+\frac{\vec{\lambda}}{4}\right)+\mathrm{G}\vec{\mathrm{y}}\leq\vec{b}-\frac{1}{4}\sum_{j=1}^{n}|A_{*,j}|$$

$$A\left(\vec{x}-\frac{\vec{\lambda}}{4}\right)+\mathrm{G}\vec{\mathrm{y}}\leq\vec{b}-\frac{1}{4}\sum_{j=1}^{n}|A_{*,j}|$$

$$\vec{\lambda}\in\{-1,1\}^{n}$$

$$\vec{x}\in\mathbb{R}^{n}$$

$$\vec{y}\in\mathbb{R}^{n_2}$$

Then we re-orient the problem to the positive orientation by redefining A and $\vec{c}$ to be

$$A_{*,j}=A_{*,j}\cdot\vec{\lambda}_j,\qquad\forall j\in\{1,\dots,n\}$$

$$c_j=c_j\cdot\vec{\lambda}_j,\qquad\forall j\in\{1,\dots,n\}$$

7.4.2 Optimising GIR - Quadratic

The equations in Section 6.7.2 have now been simplified to

$$\max\vec{c}\cdot\vec{x}+\vec{h}\cdot\vec{y}$$

subject to

$$\vec{x},\vec{r},\vec{s},\vec{\mathrm{q}},\vec{\mathrm{d}}\in\mathbb{R}^{n}$$

$$\vec{y}\in\mathbb{R}^{n_2}$$

$$r_j,s_j,q_j\geq0,\qquad j\in\{1,\dots,n\}$$

$$d_j=r_j-s_j,\qquad j\in\{1,\dots,n\}$$

$$T=diag(\vec{q})+\vec{1}\left(\vec{q}\circ\vec{d}\right)^{T}$$

GIR constraints:

$$\sum_{j=1}^{n}d_j\geq-1$$

$$1+\sum_{j\in\{1,\dots,n\}\setminus i}r_j\leq q_i\left(1+\sum_{j=1}^{n}d_j\right),\quad\forall i\in\{1,\dots,n\}$$

$$\sum_{j\in\{1,\dots,n\}\setminus i} s_j \le q_i\left(1+\sum_{j=1}^{n} d_j\right),\quad \forall i\in\{1,\dots,n\}$$

Ordering constraints:

$$-1 \le d_j \max_{i\in\{1,\dots,m\}}\left(\frac{1}{a_{i,j}}\sum_{k\in\{1,\dots,n\}} a_{i,k}\right),\qquad j\in\{1,\dots,n\}, a_{i,j}\neq 0$$

$$-1 \le d_j \min_{i\in\{1,\dots,m\}}\left(\frac{1}{a_{i,j}}\sum_{k\in\{1,\dots,n\}} a_{i,k}\right),\qquad j\in\{1,\dots,n\}, a_{i,j}\neq 0$$

$$A_{i,*}\left(\vec{x}+\frac{1}{2}T\sigma\left(A_{i,*}\right)^T\right)+G_{i,*}\vec{y}\le\vec{b}_i,\quad \forall i\in\{1,\dots,m\}$$

While this significantly reduces the complexity compared to the equations in Section 6.7.2, these constraints do form a non-convex quadratic optimisation problem.

7.4.3 Optimising GIR – Linearising

An important property to note from the equations in Section 7.4.2 is that the only variables that interact non-linearly are q and d. This is a significant improvement given the complexity of interactions in the equations in Section 6.7.2.

The fact that q and d have a non-linear interaction is not surprising given that T is almost entirely composed of the product of $q_i d_i,\ \forall i\in\{1,\dots,n\}$. However, as noted in Section 7.1.1, while d affects both the size and shape of the GIR, q is only a scaling parameter.

Whilst investigating, it was observed that with a small change in d, the value of the best GIR that could be found did not change much. Whilst a small change in q had a far greater impact on the quality of the solution.

This prompted the development of an alternative solution technique which involves decomposing the optimisation of the GIR into a master and sub problem as shown below:

Master problem

$$\max L(d)$$

Subject to

$$\vec{\mathrm{d}}\in\mathbb{R}^n$$

$$\sum_{j=1}^{n} d_j \geq -1$$

$$-1 \leq d_j \max_{i \in \{1,..,m\}} \left(\frac{1}{a_{i,j}} \sum_{k \in \{1,...,n\}} a_{i,k} \right), \qquad j \in \{1, \dots, n\}, a_{i,j} \neq 0$$

$$-1 \leq d_j \min_{i \in \{1,..,m\}} \left(\frac{1}{a_{i,j}} \sum_{k \in \{1,...,n\}} a_{i,k} \right), \qquad j \in \{1, \dots, n\}, a_{i,j} \neq 0$$

Sub problem

$$L(d)\colon \max \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

subject to

$$\vec{x}, \vec{\mathrm{q}} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

$$r_j = \begin{cases} d_j, & d_j > 0 \\ 0, & d_j \leq 0 \end{cases}$$

$$s_j = \begin{cases} -d_j, & d_j < 0 \\ 0, & d_j \geq 0 \end{cases}$$

$$q_j \geq 0, \qquad j \in \{1, \dots, n\}$$

$$T = diag(\vec{q}) + \vec{1}\left(\vec{q} \circ \vec{d}\right)^T$$

$$1 + \sum_{j \in \{1,...,n\} \setminus i} r_j \leq q_i \left(1 + \sum_{j=1}^{n} d_j \right), \quad \forall i \in \{1, \dots, n\}$$

$$\sum_{j \in \{1,...,n\} \setminus i} s_j \leq q_i \left(1 + \sum_{j=1}^{n} d_j \right), \quad \forall i \in \{1, \dots, n\}$$

$$A_{i,*} \left(\vec{x} + \frac{1}{2} T \sigma \left(A_{i,*} \right)^T \right) + G_{i,*} \vec{y} \leq \vec{b}_i, \quad \forall i \in \{1, \dots, m\}$$

From the decomposition it can be seen that all the constraints of the master problem are just bounds on d and all but one constraint of the subproblem are bounds on q. In addition, the subproblem now forms an LP similar in complexity to the LP relaxation of the original MILP. This is significant as a major component of branch and cut is solving LP relaxations of MILPs, so if the overheads involved in using

RM are small enough, and RM reduces the LP relaxations needing to be solved, then it has the potential to compete with existing commercial techniques.

7.4.4 Extracting Integer Solution

We can now use the GIR found by either previous method to generate a MILP that, when solved, gives the solution to the original MILP. To do this we need x, the location of the GIR, and q, d which define the shape. There are two formulations that can be used, both giving identical results, but the choice of formulation should be based on computation efficiency. Both formulations need their output reorientated to reverse the previous reorientation.

Method 1:

The integer extraction step has been simplified to

$$\text{Max } \vec{c} \cdot \vec{\gamma} + \vec{h} \cdot \vec{y}$$

subject to

$$\left| \gamma_i - x_i + \sum_{j \in \{1,\dots,n\} \setminus i} d_j (\gamma_i - x_i + x_j - \gamma_j) \right| \leq \frac{q_i}{2} \left(1 + \sum_{j=1}^{n} d_j \right), \quad \forall i \in \{1, \dots, n\}, \vec{\mathrm{v}} \in \Lambda$$

$$A\vec{\gamma} + G\vec{y} \leq \vec{b}^*$$

$$\vec{\gamma} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Method 2:

$$\text{Max } \vec{c} \cdot \vec{\gamma} + \vec{h} \cdot \vec{y}$$

$$\vec{\gamma} - \vec{\mathrm{x}} = Ts$$

$$-\frac{\vec{1}}{2} \leq \vec{s} \leq \frac{\vec{1}}{2}, \qquad \vec{s} \in \mathbb{R}^n$$

$$A\vec{\gamma} + G\vec{y} \leq \vec{b}^*$$

$$\vec{\gamma} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Thus, the solution to the original problem is

$$x_j = \gamma_j \cdot \lambda_j, \qquad \forall j \in \{1, \dots, n\}$$

8 Worked Example

In this section we work through applying RM to a simple problem, Integer Problem 2. We will follow through the algorithm listed in Section 7.4.

8.1 Problem statement

Problem

Max $z = 2x + 5y$

s.t.

$12x + 5y \leq 60$

$2x + 10y \leq 35$

$x, y\ int$

This can be put into standard MILP form as follows.

$$A = \begin{pmatrix} 12 & 5 \\ 2 & 10 \end{pmatrix}$$

$$G = (0)$$

$$b = \begin{pmatrix} 60 \\ 35 \end{pmatrix}$$

$$c = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

$$h = (0)$$

The problem is represented graphically in Figure 8.1.

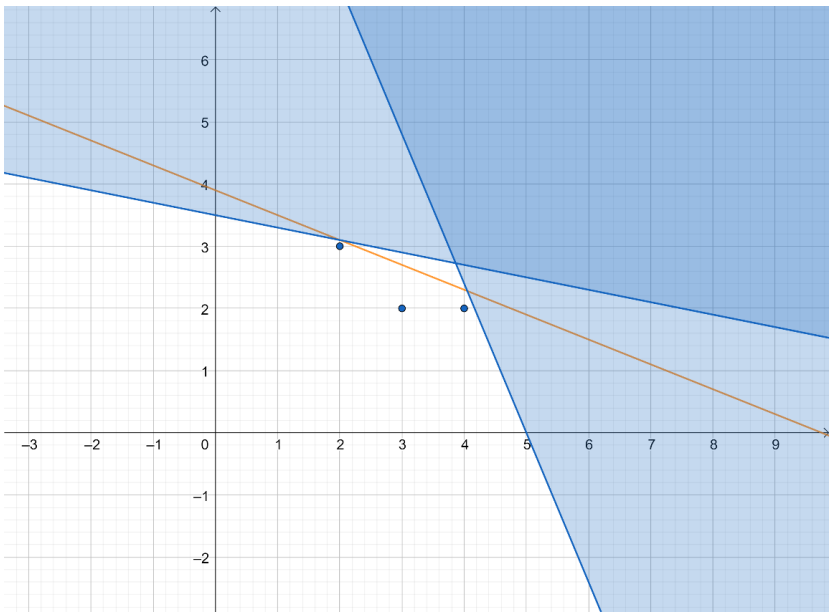

Figure 8.1 Graphical representation of Integer Problem 2

In this figure the dots represent possible integer solutions, the objective function is represented by the orange line, the constraints are represented by the blue lines and shading, and the feasible space is the unshaded region. This figure shows the 3 best integer solutions.

8.2 Loosening bounds

Before RM can be applied some pre-processing can be done, the first stage of this is loosening the bounds. Figure 8.2 shows Integer problem 2 with the bounds loosened.

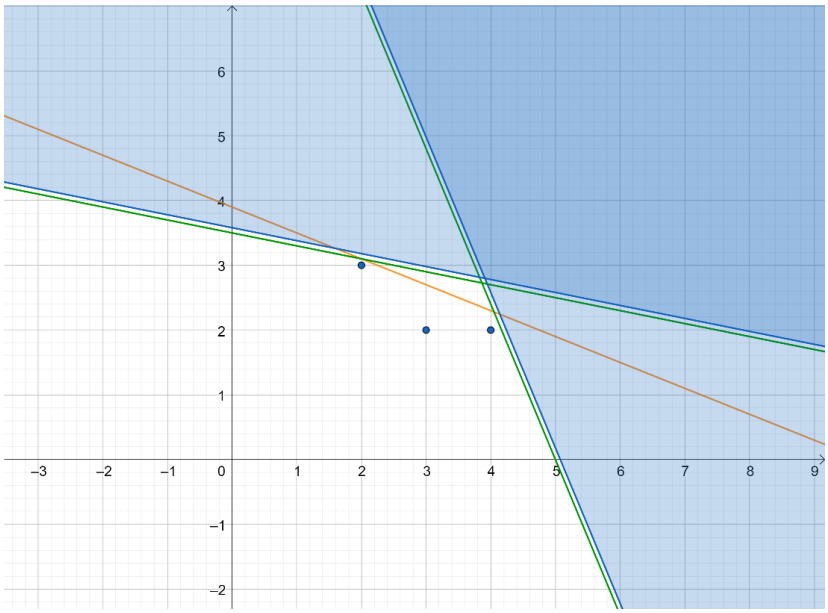

Figure 8.2 Integer Problem 2 with loosened bounds

In this figure the dots represent possible integer solutions, the objective function is represented by the orange line, the updated constraints are represented by the blue lines and shading, while the green lines represent the original constraints, and the feasible space is the unshaded region.

The first stage in doing this is to apply Gomory-Chvátal cuts for every constraint. This is done by dividing each row of the A matrix by the gcd of the row, also do this to b and G, then take the floor of b.

$$A = \begin{pmatrix} \frac{12}{1} & \frac{5}{1} \\ \frac{2}{2} & \frac{10}{2} \end{pmatrix} = \begin{pmatrix} 12 & 5 \\ 1 & 5 \end{pmatrix}$$

$$b = \begin{pmatrix} \left\lfloor \frac{60}{1} \right\rfloor \\ \left\lfloor \frac{35}{2} \right\rfloor \end{pmatrix} = \begin{pmatrix} 60 \\ 17 \end{pmatrix}$$

Let this rescaled b be represented by b^*.

Then the problem is relaxed by adding $1 - \delta$ to each element of b.

where

$$\delta > 0$$

For this problem $\delta = 0.1$ was chosen.

$$b = \begin{pmatrix} 60.9 \\ 17.9 \end{pmatrix}$$

As can be seen from Figure 8.2 when we loosen the bounds there are no new solutions added to the feasible space.

8.3 Reorienting problem

The next step is to find which orientation the GIR should take. Figure 8.3 shows the problem in its original orientation.

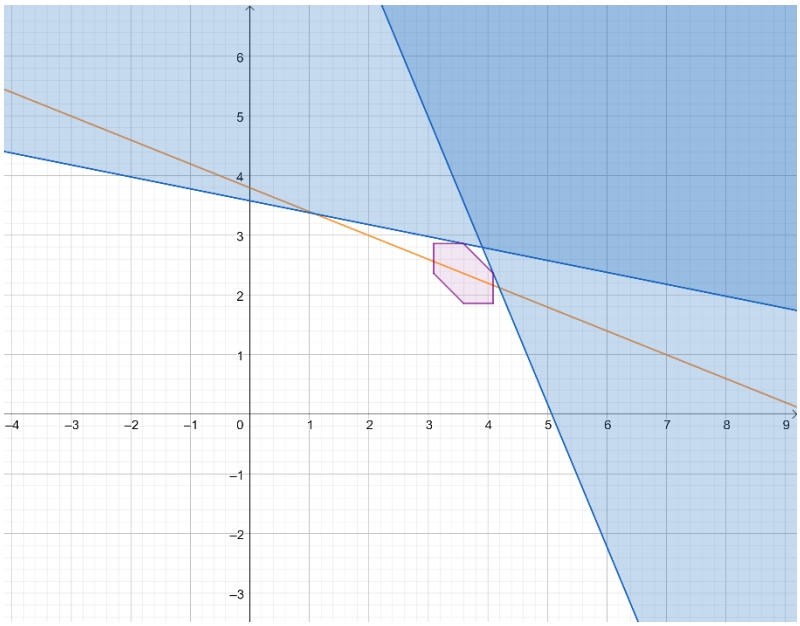

Figure 8.3 Integer Problem 2 with convex hull of the oriented sub-cubes

In this figure the objective function is represented by the orange line, the constraints are represented by the blue lines and shading, the feasible space is the unshaded region, and the convex hull of the sub-cubes that are required for orientating the problem are represented by the purple shaded region. The set of sub-cubes that best fit this problem show that the problem doesn't have a positive orientation and therefore must be re-oriented.

The problem is reoriented by solving

$$\text{Max } \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

Subject to

$$A\left(\vec{x} + \frac{\vec{\lambda}}{4}\right) + \text{G}\vec{\text{y}} \leq \vec{b} - \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

$$A\left(\vec{x} - \frac{\vec{\lambda}}{4}\right) + \text{G}\vec{\text{y}} \leq \vec{b} - \frac{1}{4}\sum_{j=1}^{n} |A_{*,j}|$$

$$\vec{\lambda} \in \{-1,1\}^n$$

$$\vec{x} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

where $|A_{*,j}|$ is the vector of absolute values of the elements in the jth column of the matrix A.

Substituting in the coefficients from this problem gives

$$\text{Max } 2x_1 + 5x_2$$

Subject to

$$12x_1 + 3\lambda_1 \leq 60.9 - \frac{17}{4}$$

$$5x_2 + \frac{5}{4}\lambda_2 \leq 17.9 - \frac{6}{4}$$

$$12x_1 - 3\lambda_1 \leq 60.9 - \frac{17}{4}$$

$$5x_2 - \frac{5}{4}\lambda_2 \leq 17.9 - \frac{6}{4}$$

$$\vec{\lambda} \in \{-1,1\}^n$$

$$\vec{x} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

For a more effective implementation it is evident that the term $2\lambda_i - 1$ is replaced with a binary variable.

Solving this gives

$$\vec{\lambda} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\vec{x} = \begin{pmatrix} 3.591 \\ 2.362 \end{pmatrix}$$

This solution is depicted in Figure 8.3.

Then using the orientation $\vec{\lambda}$ we transform the problem to ensure that the orientation is positive as seen in Figure 8.4.

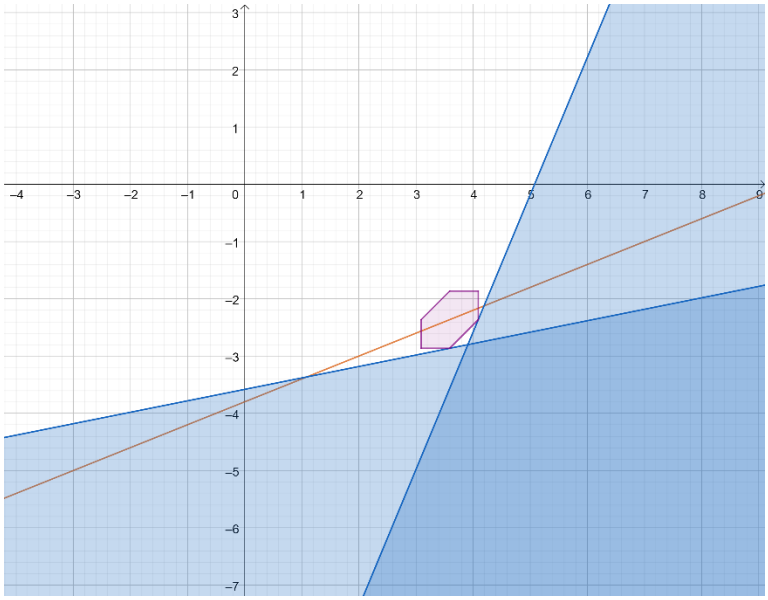

Figure 8.4 Integer Problem 2, re-oriented so the sub-cubes are now positively oriented

In this figure the objective function is represented by the orange line, the constraints are represented by the blue lines and shading, the feasible space is the unshaded region, and the convex hull of the sub-cubes that are required for orientating the problem are represented by the purple shaded region. As the problem has been re-oriented the optimal sub-cubes are now positively oriented.

This is done by redefining A and $\vec{c}$ to be

$$A_{*,j} = A_{*,j} \cdot \vec{\lambda}_j, \qquad \forall j \in \{1, \dots, n\}$$

$$c_j = c_j \cdot \vec{\lambda}_j, \qquad \forall j \in \{1, \dots, n\}$$

which gives

$$A = \begin{pmatrix} 12 & -5 \\ 1 & -5 \end{pmatrix}$$

$$c = \begin{pmatrix} 2 \\ -5 \end{pmatrix}$$

8.4 Finding bounds on $\vec{d}$

Now that an orientation has been selected and imposed, we can pre-calculate the bounds on the shape parameter $\vec{d}$ so that the orientation constraints are not violated. As we are optimising by the centroid of the GIR, we do not need to consider $\vec{c}$ when finding the bounds. These bounds can be found by evaluating the following inequalities:

$$-1 \leq d_j \min_{i\in\{1,..,m\}} \left(\frac{1}{a_{i,j}} \sum_{k\in\{1,...,n\}} a_{i,k} \right), \qquad j \in \{1, \dots, n\}, a_{i,j} \neq 0$$

$$-1 \leq d_j \max_{i\in\{1,..,m\}} \left(\frac{1}{a_{i,j}} \sum_{k\in\{1,...,n\}} a_{i,k} \right), \qquad j \in \{1, \dots, n\}, a_{i,j} \neq 0$$

The following bounds result:

$$-1 \leq \frac{7}{12} d_1$$

$$-1 \leq \frac{4}{5} d_2$$

$$-1 \leq -4 d_1$$

$$-1 \leq -\frac{7}{5} d_2$$

It is possible to re-write these in the form of

$$d_j^{\min} \leq d_j \leq d_j^{\max}, \qquad j \in \{1, \dots, n\}$$

$$-\frac{12}{7} \leq d_1 \leq \frac{1}{4}$$

$$-\frac{5}{4} \leq d_2 \leq \frac{5}{7}$$

8.5 Applying RM

Now that the pre-calculation steps are complete, we can optimise the GIR. In this worked example the quadratic method of optimising has been chosen and the highest value GIR is shown in Figure 8.5.

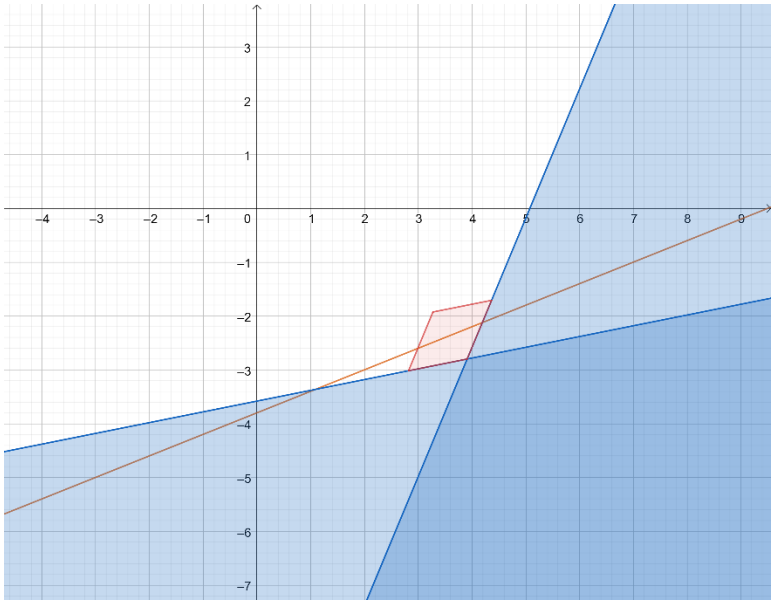

Figure 8.5 Integer Problem 2 with the highest valued GIR found by RM

In this figure the objective function is represented by the orange line, the constraints are represented by the blue lines and shading, the feasible space is the unshaded region, and the optimal GIR found is the red region.

The highest value GIR is found by solving the following problem:

$$\text{Max } \vec{c} \cdot \vec{x} + \vec{h} \cdot \vec{y}$$

Subject to

$$T = diag(\vec{q}) + \vec{1}\left(\vec{q} \circ \vec{d}\right)^T$$

$$d_j = r_j - s_j, \qquad j \in \{1, \dots, n\}$$

$$1 + \sum_{j \in \{1,\dots,n\} \setminus i} r_j \leq q_i \left(1 + \sum_{j=1}^{n} d_j\right), \quad \forall i \in \{1, \dots, n\}$$

$$\sum_{j \in \{1,\dots,n\} \setminus i} s_j \leq q_i \left(1 + \sum_{j=1}^{n} d_j\right), \quad \forall i \in \{1, \dots, n\}$$

$$A_{i,*}\left(\vec{x} + \frac{1}{2} T \sigma\left(A_{i,*}\right)^T\right) + G\vec{y} \leq \vec{b}$$

$$\sum_{j=1}^{n} d_j \geq -1$$

$$d_j^{\min} \leq d_j \leq d_j^{\max}, \qquad j \in \{1, \dots, n\}$$

$$r_j, s_j \geq 0, \qquad j \in \{1, \dots, n\}$$

$$\vec{x}, \vec{r}, \vec{s} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

For the best performance the substitution $d_j = r_j - s_j$ should be applied before solving.

Substituting the coefficients into this gives the following:

$$\text{Max } 2x_1 - 5x_2$$

Subject to

$$d_1 = r_1 - s_1$$

$$d_2 = r_2 - s_2$$

$$1 + r_2 \leq q_1(1 + d_1 + d_2)$$

$$1 + r_1 \leq q_2(1 + d_1 + d_2)$$

$$s_2 \leq q_1(1 + d_1 + d_2)$$

$$s_1 \leq q_2(1 + d_1 + d_2)$$

$$12x_1 - 5x_2 + \frac{1}{2}\Big(12(q_1(1 + d_1) - q_2 d_2) - 5\big(q_1 d_1 - q_2(1 + d_2)\big)\Big) \leq 60.9$$

$$x_1 - 5x_2 + \frac{1}{2}\Big((q_1(1 + d_1) - q_2 d_2) - 5\big(q_1 d_1 - q_2(1 + d_2)\big)\Big) \leq 17.9$$

$$-d_1 - d_2 \leq 1$$

$$-\frac{12}{7} \leq d_1 \leq \frac{1}{4}$$

$$-\frac{5}{4} \leq d_2 \leq \frac{5}{7}$$

$$r_j, s_j \geq 0, \qquad j \in \{1, \dots, n\}$$

$$\vec{x}, \vec{r}, \vec{s} \in \mathbb{R}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

An approximate solution to this is

$$x = \begin{pmatrix} 3.591 \\ -2.362 \end{pmatrix}$$

$$q = \begin{pmatrix} 0.873 \\ 0.636 \end{pmatrix}$$

$$d = \begin{pmatrix} 0.250 \\ 0.714 \end{pmatrix}$$

This solution creates a GIR centred on the point x and a transformation matrix of

$$T = \begin{pmatrix} q_1(1+d_1) & q_2 d_2 \\ q_1 d_1 & q_2(1+d_2) \end{pmatrix}$$

Which in this case is

$$T = \begin{pmatrix} 1.091 & 0.455 \\ 0.218 & 1.091 \end{pmatrix}$$

This solution is graphically represented in Figure 8.5.

It is worth noting that when either $\vec{q}$ or $\vec{d}$ is fixed the problem becomes linear, therefore once a solution is found it can be refined using a linear solver.

8.6 Finding Integer Solution

We can now find the integer solution contained within the GIR as seen in Figure 8.6.

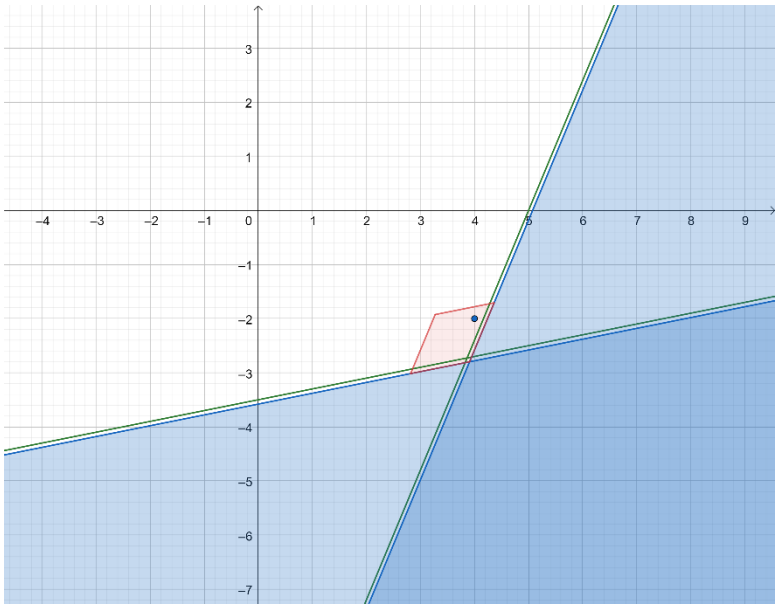

Figure 8.6 Integer Problem 2 with the best solution found within the highest valued GIR, and the original constraints shown

In this figure the constraints are represented by the blue lines and shading, the green lines the original constraints, the feasible space is the unshaded region, the optimal GIR found is the red region, and the optimal integer solution found within the GIR is represented by the dot.

There are two different ways to extract the integer as discussed in Sections 6.6 and 7.3.

8.6.1 Method 1

The first method uses slack variables and is done by solving:

$$\text{Max } \vec{c} \cdot \vec{\gamma} + \vec{h} \cdot \vec{y}$$

$$\vec{\gamma} - \vec{\mathrm{x}} = Ts$$

$$-\frac{\vec{1}}{2} \leq \vec{s} \leq \frac{\vec{1}}{2}, \qquad \vec{s} \in \mathbb{R}^n$$

$$A\vec{\gamma} + G\vec{y} \leq \vec{b}^*$$

$$\vec{\gamma} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

where $\vec{x}$ is the centre of the GIR found in the previous Section, and $\vec{b}^*$ is found in Section 8.2.

The solution to original problem is

$$\gamma_j \cdot \lambda_j, \qquad \forall j \in \{1, \dots, n\}$$

With $\vec{y}$ being unchanged from solving the previous MILP.

Substituting the solution found in Section 8.5 this results in

$$\text{Max } 2\gamma_1 - 5\gamma_2$$

$$\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} - \begin{pmatrix}3.591\\ -2.362\end{pmatrix} = \begin{pmatrix}1.091 & 0.455\\ 0.218 & 1.091\end{pmatrix}\begin{pmatrix}s_1\\ s_2\end{pmatrix}$$

$$-\frac{\vec{1}}{2} \leq \vec{s} \leq \frac{\vec{1}}{2}, \qquad \vec{s} \in \mathbb{R}^n$$

$$\begin{pmatrix}12 & -5\\ 1 & -5\end{pmatrix}\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \leq \begin{pmatrix}60\\ 17\end{pmatrix}$$

$$\vec{\gamma} \in \mathbb{Z}^n$$

8.6.2 Method 2

Alternatively, the solution can be found using

$$\text{Max } \vec{c} \cdot \vec{\gamma} + \vec{h} \cdot \vec{y}$$

subject to

$$\left| \gamma_i - x_i + \sum_{j \in \{1,\dots,n\} \setminus i} d_j \left(\gamma_i - x_i + x_j - \gamma_j \right) \right| \leq \frac{q_i}{2} \left(1 + \sum_{j=1}^{n} d_j \right), \quad \forall i \in \{1, \dots, n\}, \vec{v} \in \Lambda$$

$$A\vec{\gamma} + G\vec{y} \leq \vec{b}^*$$

$$\vec{\gamma} \in \mathbb{Z}^n$$

$$\vec{y} \in \mathbb{R}^{n_2}$$

Substituting the solution found in Section 8.5 this results in

$$|1.714\gamma_1 - 0.714\gamma_2 - 7.841| \leq 0.857$$

$$|-0.250\gamma_1 + 1.250\gamma_2 + 3.850| \leq 0.625$$

$$\begin{pmatrix} 12 & -5 \\ 1 & -5 \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \leq \begin{pmatrix} 60 \\ 17 \end{pmatrix}$$

$$\vec{\gamma} \in \mathbb{Z}^n$$

Both methods result in the solution

$$\vec{\gamma} = \begin{pmatrix} 4 \\ -2 \end{pmatrix}$$

As shown in Figure 8.6.

To obtain the solution to the original problem $\vec{x}$ we must undo the reorientation.

$$x_j = \gamma_j \cdot \lambda_j, \qquad \forall j \in \{1, \dots, n\}$$

$$x_j = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

Whilst this solution is close to the optimal solution it is not the optimal solution which lies at

$$x_j = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

This is discussed in more detail in Section 9.4.

9 Results

The aim of this research was to investigate whether a new algorithm for solving MILPs can be created, using regions that are guaranteed to contain an integer point, removing the need for integer constraints. This method has been successfully implemented and tested. However, extensive testing and further optimisation of this technique could be explored in more detail as future research.

Throughout this section, GIR Value denotes the value of the objective function of the Optimising GIR stage of RM. Whereas Solution Value denotes the value of the integer solution found within the GIR, which is obtained through the Extracting Integer Solution stage. The GIR gap ratio is the ratio between a GIR Value and the highest GIR Value obtained when optimising the problem. The optimality gap ratio is the ratio between a Solution Value and the optimal solution to the original MILP.

9.1 Implementation

We implemented Region Method (RM) with the programming language Julia [48], using the package JuMP [49], to allow for flexibility in the choice of solvers used. It was decided to use CPLEX to solve any linear problems and to use NLopt [50] to solve any non-linear problems. The algorithm used within NLopt was the COBYLA [51] algorithm. A link to codes developed can be found in the appendix.

9.2 Objective function

As noted in Section 6.5, there are different objective functions that can be chosen when assessing the quality of a GIR. These being the centroid of the GIR, and the two extrema of the GIR in respect to the original problem's objective function. These are represented by the following equation:

$$\vec{c} \cdot \vec{x} + \frac{\beta}{2} \vec{c} \cdot T\sigma(\vec{c})$$

$$-1 \leq \beta \leq 1$$

when $\beta \in \{-1,0,1\}$ this corresponds to the minima, centroid, and maxima respectively. Using this representation, we can smoothly interpolate between the three options. From numerical tests, examples of which are shown in 9.1, it was evident that the choice of β did not change the relationship between the GIR value and the solution value, apart from when $\beta = 1$. When $\beta = 1$ the relationship between the GIR value and the solution value appears to break down and so $\beta = 1$ should not be chosen. This observation was consistent across all problems tested. Given that the centroid does not need a vertex ordering constraint for the objective function, see Section 6.5, and appears to perform as well as other choices, all further work uses $\beta = 0$, which corresponds to evaluating the GIR based on its centroid.

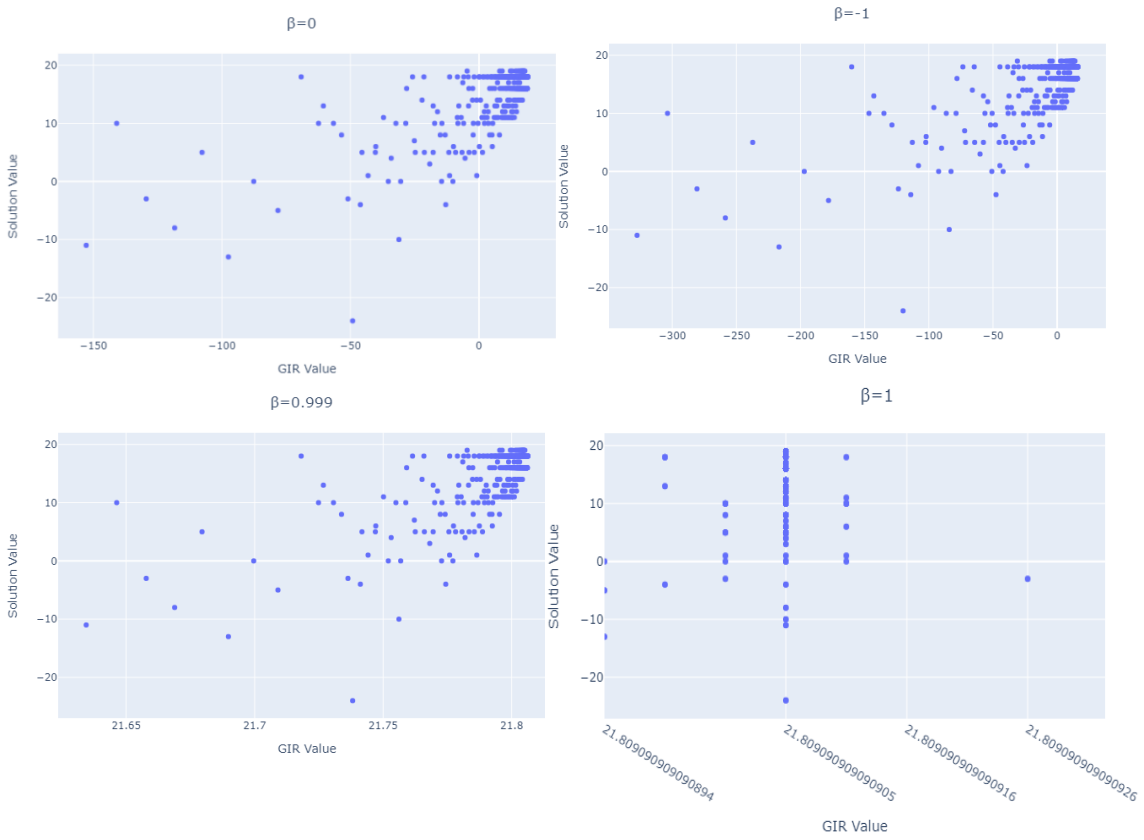

Figure 9.1 Effect of varying β on the relationship between the assigned GIR Value and the found solution value, for Integer problem 2

This figure shows that the correlation between GIR value and solution value does not change unless β equals 1.

9.3 The effect of varying the loosening of the bounds (δ)

As introduced in Section 6.1 the parameter δ controls the level of slackness applied to the problem. In loosening the problem like this it creates constraints that are arbitrarily close to Integer values which are not solutions. This is important, as making δ smaller loosens the constraints thus permitting more GIRs to be found. This can cause issues when solving RM as all solvers have a built-in tolerance for constraint violations and the interaction between the tolerance and δ can cause issues if δ is too small. When this happens, RM generates a region that is a GIR but that violates the now loosened constraints, this then causes the integer extraction to fail. When $\delta \geq 0.1$ this was not an issue for problems of two variables, but as the number of variables increased this issue became more prevalent. However, by increasing δ, the proportion of cases that returned invalid solutions decreased. This was tested by randomly generating a valid $\vec{d}$ and then finding the corresponding optimal $\vec{q}$ to generate a GIR. This GIR was then tested against the original problem and it was noted if the solution provided was outside the feasible region. As we can see from Table 9.1, the occurrence of invalid solutions generated decreases as δ is increased.

	Binary Problem 1	Binary Problem 3	MILP Problem 1	MILP Problem 2
Number of variables	4	10	9	15
Number of binary variables	4	10	3	4
$\delta = 0.1$	19.2%	27.2%	42.8%	21.2%
$\delta = 0.2$	9%	11.8%	33.4%	16.2%
$\delta = 0.4$	1.6%	2.6%	26.4%	2.4%

Table 9.1 The percentage of 500 randomly generated GIRs, verified using RM, that have a solution that lies outside the feasible region

This issue can be avoided by tightening bounds whenever a solution is found that violates them.

9.4 Correlation between GIR value and Solution value

One of the research questions was, "Can solving a MILP be well represented by finding the best GIR that fits in the feasible space?" To answer this, we randomly generated valid $\vec{d}$s and found the corresponding optimal $\vec{q}$s hence generating GIRs. We then noted the value assigned by RM to this GIR. We then computed the optimal solution to the original problem that lay inside the GIR and removed any results that did not find a valid integer solution. We also calculated the optimality gap ratio between the found integer solution and the true optimal solution. Additionally, we calculated the GIR gap ratio by finding the optimality gap ratio between the value assigned to each GIR and the maximum value assigned to a GIR found. In the Figures below, optimal solutions are shown in red and solutions with a GIR gap ratio of over 1.5 were removed.

The ideal result is that a significant proportion of all valid GIRs result in a small optimality gap or none at all, as demonstrated in MILP problem 1, and Integer Problem 3 as shown in Figure 9.2, and Figure 9.3. When this happens this means that any arbitrary $\vec{d}$ when paired with its optimal $\vec{q}$ can be chosen to generate the GIR, which means that instead of having to perform full RM a linear simpler version can be solved.

The second-best scenario is that low GIR Gap ratios result in a small optimality gap or none at all. An example that shows this behaviour is Integer Problem 1 in Figure 9.2. When this is the case finding the highest valued GIR results in an optimal or near optimal solution.

Another positive scenario is when solutions with a small optimality gap or none at all can be found at low GIR Gap ratios, but low GIR Gap solutions also give poorer solutions. This can be seen in Integer Problems 2 and 4 and MILP Problem 2 in Figure 9.2, and Figure 9.3. Finding these solutions has not been explored in this thesis and would be an area of future research.

As none of these cases apply to Binary Problems 1 and 3, shown in Figure 9.4 we would expect RM to struggle to produce a good solution to these problems. This may be improved with future improvements.

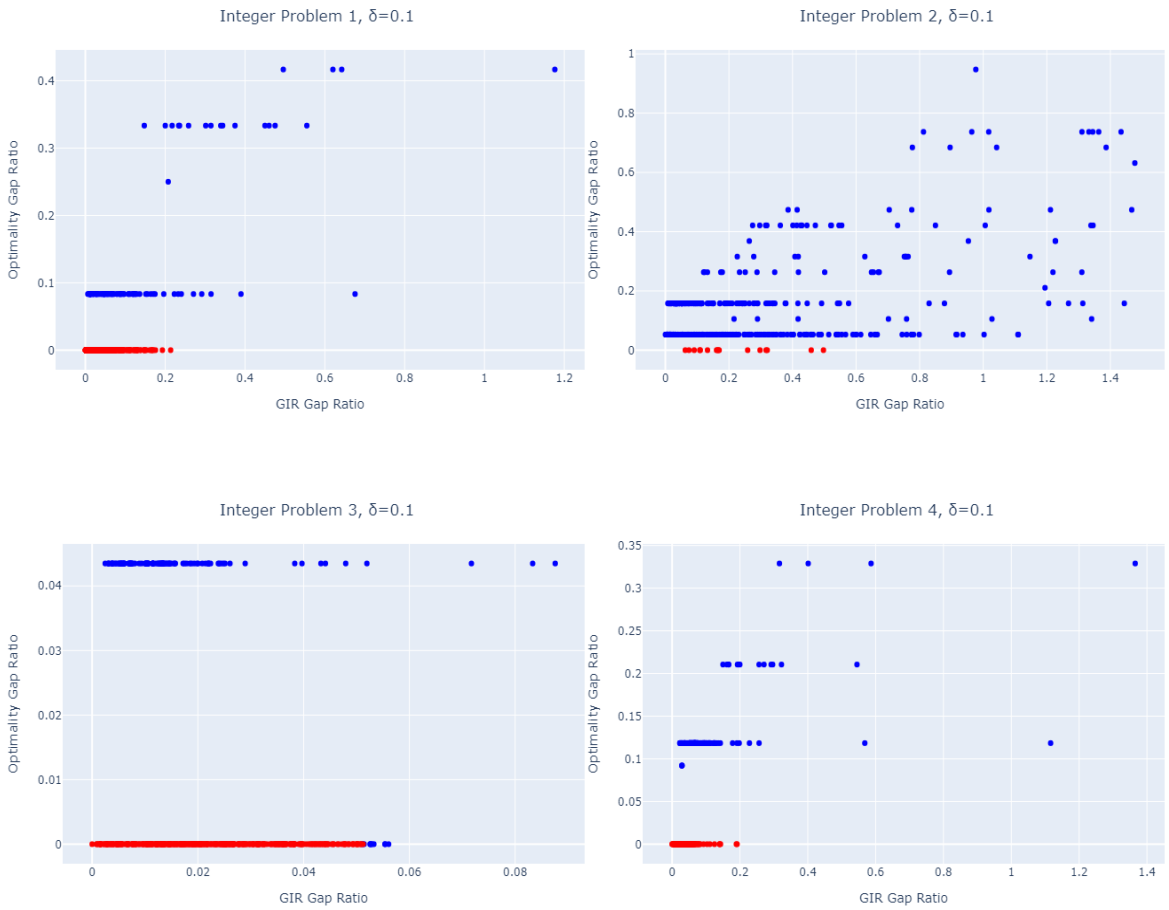

Figure 9.2 Plot of Optimality Gap Ratio vs GIR Gap Ratio for 500 randomly generated GIRs for Integer Problems 1-4

In these figure red points denote an optimality gap of zero.

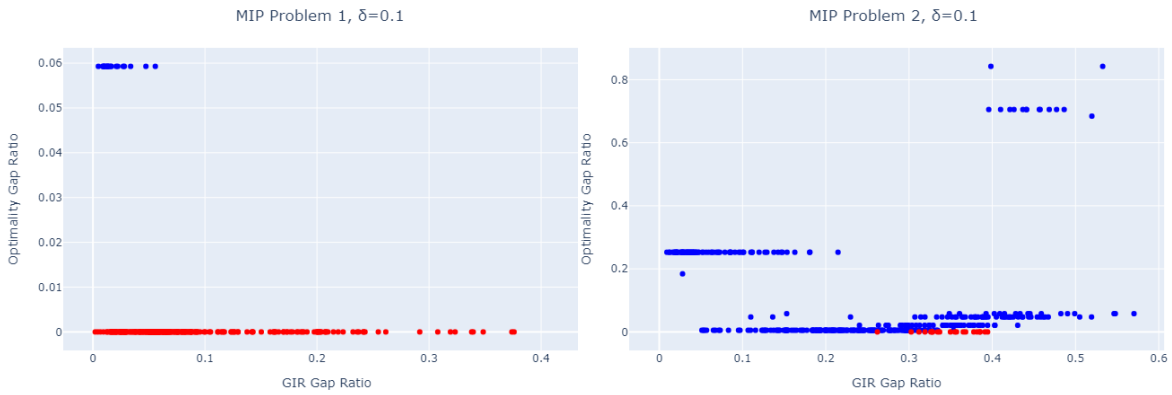

Figure 9.3 Plot of Optimality Gap Ratio vs GIR Gap Ratio for 500 randomly generated GIRs for MILP Problems 1-2

In these figure red points denote an optimality gap of zero.

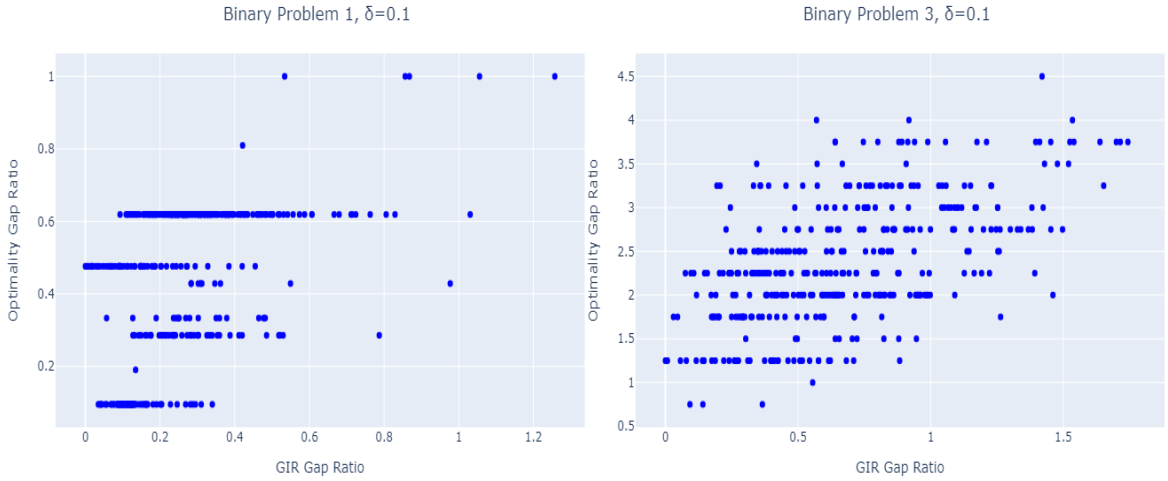

Figure 9.4 Plot of Optimality Gap Ratio vs GIR Gap Ratio for 500 randomly generated GIRs for Binary Problems 1 and 3

In these figure red points denote an optimality gap of zero.

9.4.1 Surface plots

The three cases identified in Section 9.4 were further investigated. Previously $\vec{d}$ was selected at random, so to further understand the relationship between the GIR gap ratio and the optimality gap ratio, $\vec{d}$ was systematically explored. As some problems do not have restrictions on the value of $\vec{d}$ it was decided to limit the set to $-5 \leq d_i \leq 5,\ \forall i \in \{1,2\}$. To visualise this relationship, problems with only two variables were selected as this allowed a surface plot to be drawn. The results of this are shown in Figure 9.5, Figure 9.6, and Figure 9.7. By graphing the problems in this manner, and acknowledging the small sample set, various observations can be made.

- Both the GIR Gap Ratio and the Optimality Gap Ratio are smooth over $\vec{d}$ when using an optimal $\vec{q}$, this property did not hold when $\vec{q}$ was selected using an optimal $\vec{d}$.
- Whenever the Optimality Gap Ratio graph was convex, the optimal point of both graphs overlapped.
- When the Optimality Gap Ratio graph was non-convex, the optimal point on the GIR Gap Ratio graph was a local optimum of the Optimality Gap Ratio graph.

From these observations it is evident that solving a MILP can be well represented by finding the best GIR that fits in the feasible space.

One of the motivations of this research was to be able to employ metaheuristic approaches on MILPs without having to do manual set-up. For this to be effective the relative value assigned to a GIR must

correlate well to relative optimality of the best integer solution contained within the GIR. Surface plots allow this relationship to be easily visualised for small problems. How this relationship allows for metaheuristics to be applied is explore further in Section 10.3.2.

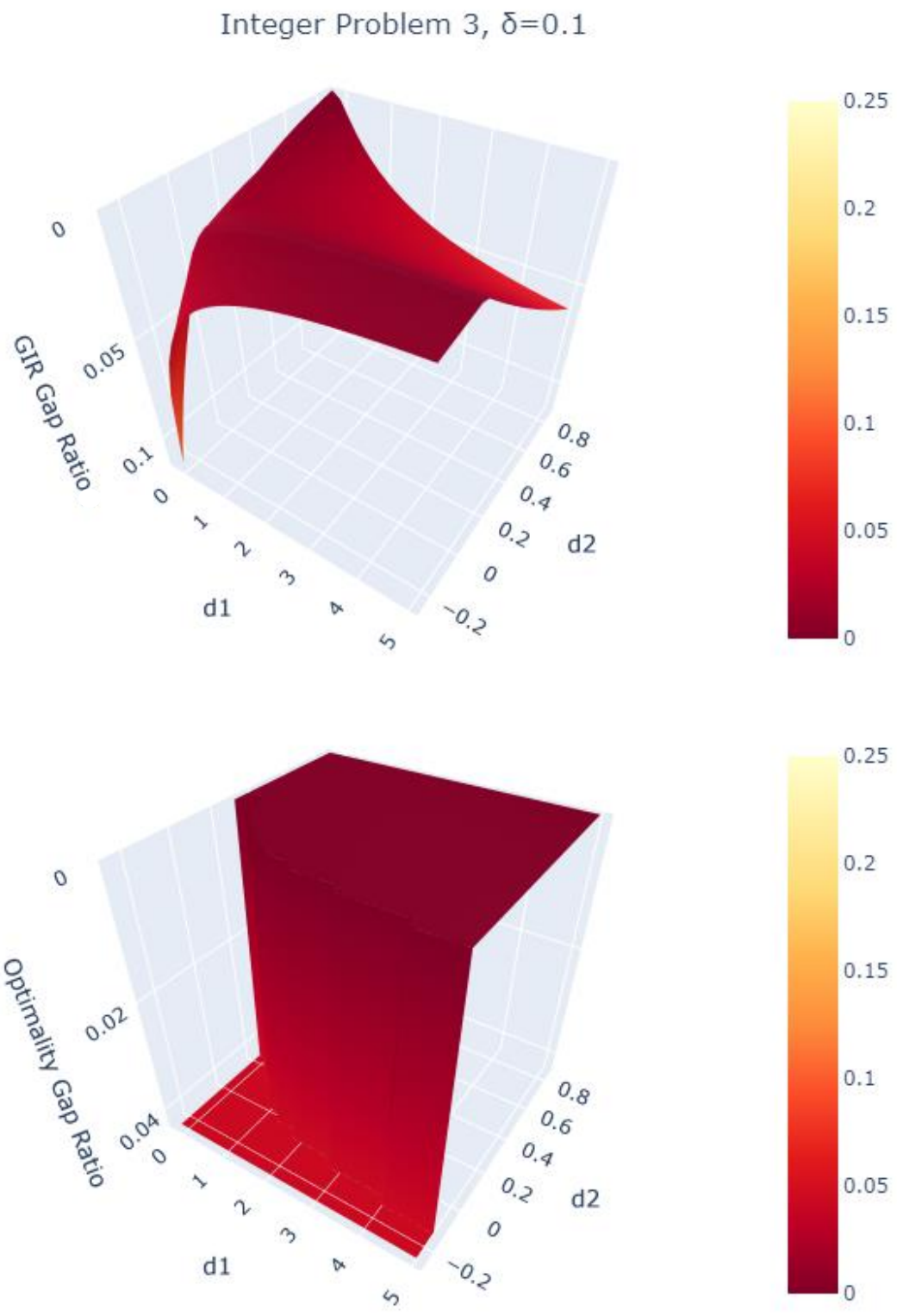

Figure 9.5 GIR Gap Ratio vs d and Optimality Gap Ratio vs d for Integer Problem 3

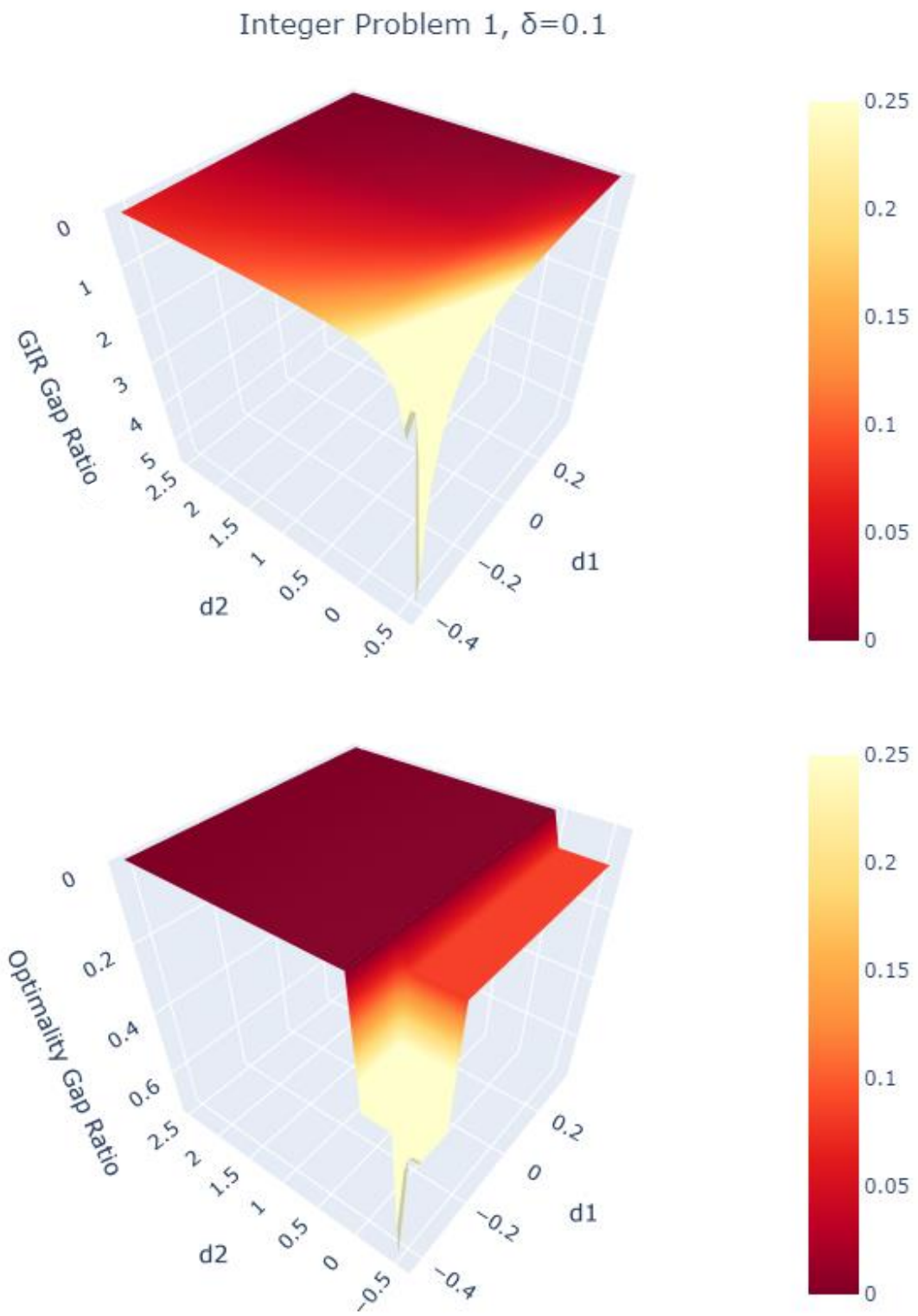

Figure 9.6 GIR Gap Ratio vs d and Optimality Gap Ratio vs d for Integer Problem 1

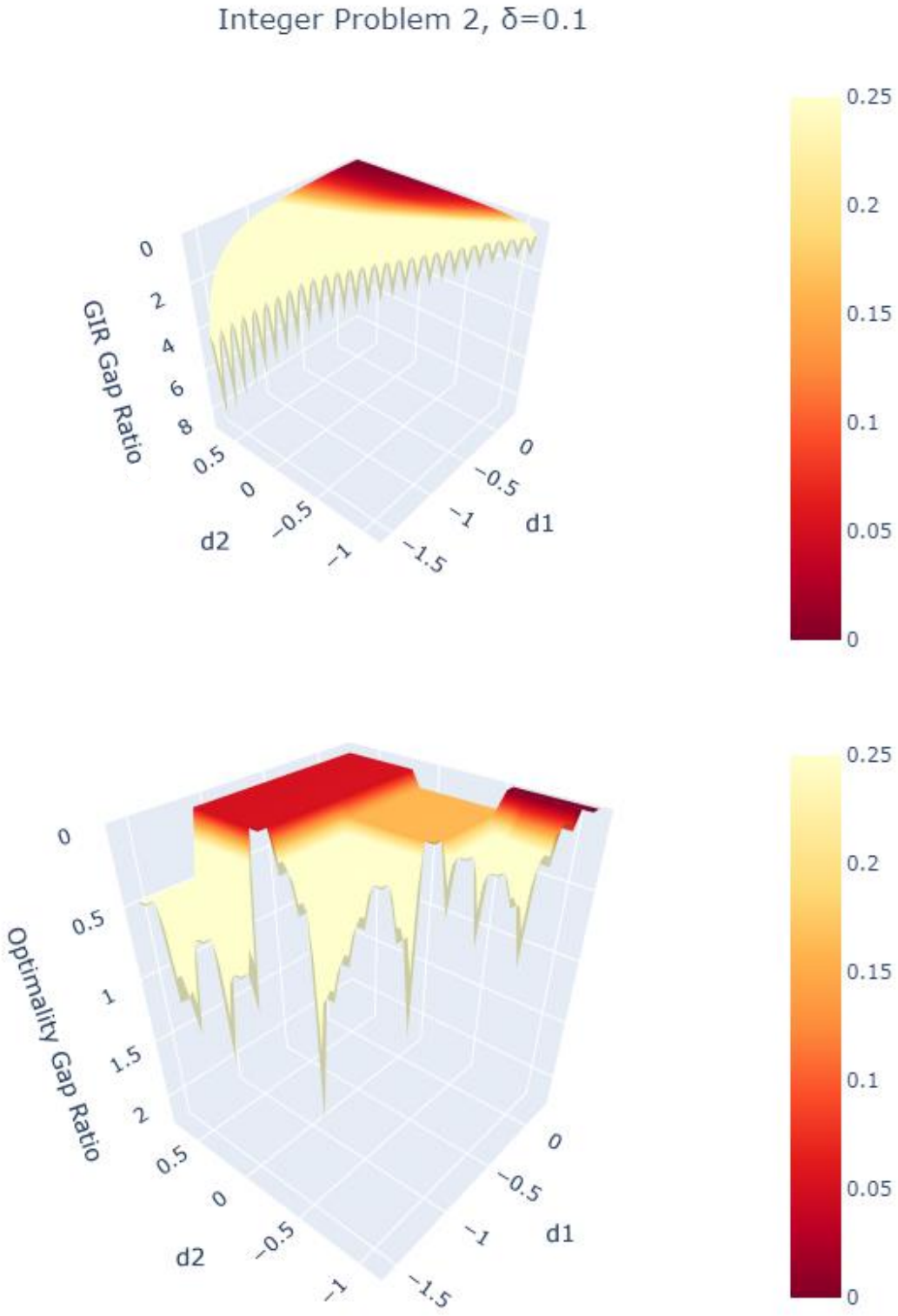

Figure 9.7 GIR Gap Ratio vs d and Optimality Gap Ratio vs d for Integer Problem 2

9.5 Overview of testing

In this section we show the results of directly applying RM to each of the problems listed in the appendix.

Problem Name	RM solution	Optimal Solution	Optimality gap
Integer Problem 1	48	48	0
Integer Problem 2	18	19	0.05*
Integer Problem 3	23	23	0
Integer Problem 4	-76	-76	0
Binary Problem 1	22	42	0.48
Binary Problem 2	N/A	5780	N/A†
Binary Problem 3	-9	-4	1.25
MILP Problem 1	202.4	202.4	0
MILP Problem 2	28400	38000	0.25*

Table 9.2 Summary of applying RM directly to each problem

* RM can generate optimal solution but does not lie at maximal solution for RM.

† During the pre-processing stage, an orientation could not be found due to the constructed MILP being infeasible. As finding the orientation is an idealised version of finding the GIR for RM if this fails due to infeasibility RM is guaranteed to fail. This is almost certainly due to the problem containing a Boolean variable with a relatively large coefficient.

10 Future work

There are several areas of focus for further work that could both improve RM and better understand the mechanics behind it.

10.1 Proof of pivot postulate

RM heavily relies on the pivot postulate, which we have not yet proven for dimensions higher than two. While it appears, from all testing, to hold true, clearly further research into RM should be tentative until a formal proof is provided, or the postulate is disproved. RM could also benefit from a more general form of pivot postulate as the restriction of needing a pair of sub-cubes removes the ability to generate a wide range of potentially useful GIRs.

10.2 Pre-Processing

As noted in the literature review, pre-processing is an important component in solving MILPs. As RM has a different goal than current solution techniques, current pre-processing techniques must also be adapted to fit. The following example demonstrates the effect of unnecessary constraints.

The simplest integer problem is to solve a problem of the form

$$0 \leq x_i \leq 0, \qquad \forall i \in \{1, \dots, n\}$$

$$\vec{x} \in \mathbb{Z}^n$$

With a relaxation applied in the pre-processing of RM this would be changed to

$$\delta - 1 \leq x_i \leq 1 - \delta, \qquad \forall i \in \{1, \dots, n\}$$

$$\delta > 0$$

$$\vec{x} \in \mathbb{Z}^n$$

This corresponds to an n-cube with side length $2(1-\delta)$.

In this form the feasible space would have volume

$$\big(2(1-\delta)\big)^n = 2^n(1-\delta)^n$$

This is the greatest volume this feasible space could take; this can be verified using Minkowski's second theorem [52]. This is an important result as having a greater volume provides more GIRs that can fit within the region. It is vital that this volume is greater than one as otherwise no GIRs fit within the feasible space. For RM we believe that this is the best representation that can be used to solve this problem.

We construct what we believe is the worst representation of this problem, that can be implemented using linear constraints. Consider if we were to add infinitely many constraints corresponding to the tangents of a n-sphere with radius $(1-\delta)$. This feasible space would have volume

$$\frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{\mathrm{n}}{2}+1\right)}(1-\delta)^n$$

It should be noted that this volume achieves it maximum when $n = 5$. Even when δ is taken to its limit we would be unable to construct a GIR that fits inside the feasible space when $n \geq 13$ as

$$\lim_{\delta \to 0} \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{\mathrm{n}}{2}+1\right)}(1-\delta)^n < 1, \qquad \forall n \geq 13$$

This example shows the importance of removing unnecessary constraints. In current techniques adding unnecessary constraints only slows solving, the addition of unnecessary constraints can cause the inability to solve the problem using RM.

An additional complication is that, in theory, having constraints arbitrarily close to integer solutions outside the region increases performance. In practise, this is complicated by tolerance within solvers giving rise to GIRs that only contain integer solutions that lie outside the region. An area to investigate might be to individually calculate how much each constraint can be loosened and still allow the solvers' tolerances to act as intended.

10.3 Further extensions

10.3.1 Convex MIPs

None of the components of RM require the problem's constraints to be linear, they only need to be convex. This implies that RM could be extended to solve mixed integer convex programming (MICP) problems without much modification. The solving of MICP problems is an active area of research [53].

10.3.2 Metaheuristics

As shown in Section 7.4.3, when optimising the GIR for RM we can break the problem down into a master and sub-problem. When doing this the master problem only has simple constraints. These are bounds on d and one simple constraint involving d. While the sub-problem is an LP. The structure of Optimising the GIR lends itself to an iterative solving technique. However, finding the best GIR does not guarantee the best solution, as shown in the results. The results also show that the best method would be one that can quickly generate many optimal or near optimal GIRs, using a diversity of shape parameters. This is a problem that is well suited to metaheuristics. A flow diagram showing this approach is shown in Figure 10.1.

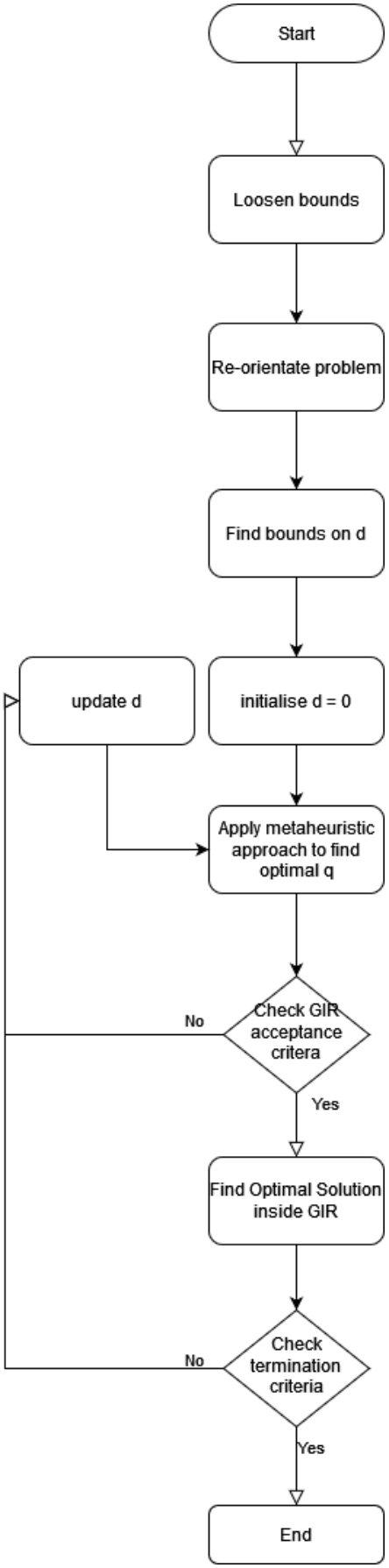

Figure 10.1 Process diagram for how RM can be converted to use metaheuristics

10.3.3 Transformation matrices

Although the transformation matrices currently used in RM appear to work well, a deeper understanding into the interactions between GIRs, pivot postulate, and the geometry of MILPs may result in better transformation matrices being found. The current transformation matrix may also be less suitable after implementation of the above extensions.

11 Conclusion

The aim of this research was to create a new algorithm for solving MILPs, based upon the concept of identifying and manipulating regions encapsulating integer solutions, removing the need for integer constraints.

A fundamental requirement to construct a method that works like this is to be able to construct and verify if a region is guaranteed to contain at least one integer point. We refer to such a region as a guaranteed integer region (GIR). If a GIR lies within a MILPs feasible space, an integer point that lies inside it also lies inside the feasible space. As a GIR always has at least one integer point that lies inside it, a GIR that lies inside the feasible space always contains a solution to the MILP.

In this thesis, such a method was developed and called the Region Method (RM). To be able to identify and manipulate regions they need to be parameterised in some way. This was done using a location, shape, and scale parameter for each dimension. This means when we reformulate the model, we modify all constraints that contain integer variables to instead contain a combination of the region parameters.

There are several stages to RM. The first stage is pre-processing, in this stage a MILP in standard form is relaxed without adding any integer solutions. This is done to create the largest volume in which these regions can be constructed. The method that is employed to do this a modified version of Gomory-Chvátal cuts. Next the problem is reoriented to take advantage of symmetries that occur when generating a region. The final step of pre-processing is calculating the bounds on the shape parameter. This can be done before any optimisation takes place.

The next stage in RM is to reformulate the problem so that we are now optimising a region instead of a point. There are two major sets of constraints that are added to accomplish this. The first set of constraints is added to ensure the region is a GIR. The second set of constraints ensure that for every constraint the closest vertex remains the same no matter which region is generated. This second set of constraints is important as it allows the original constraints to be easily modified to be based on the closest vertex. By doing so, a quadratic non-convex problem is formed, composed of location, shape and scale parameters that define the region.

The next stage is to solve this quadratic problem. Currently this is done using a non-linear solver. It should be noted that only the shape and scale parameters are linked non-linearly, if either of these variables are fixed the problem becomes linear. Once optimal values are attained, the region parameters are used with the original problem's constraints to construct a MILP that, once solved, contains the output to RM. This new MILP is much smaller than the original problem and should be much faster to solve.

Through this thesis RM is developed and implemented on small test problems and can be seen to generate valid solutions. However, due to relaxation approaches applied and solving tolerances invalid solutions can be returned. On inspection of these occurrences, RM was found to return a valid GIR contained within the feasible space.

In its current form RM does not present many advantages over current solution techniques. However, current solution techniques are very mature and required extensive research before they were considered useful. Additionally, there are many areas which have been identified where RM could be developed, including the possible integration of metaheuristic techniques.

An advantage of RM is that for some classes of MILP, highlighted in Integer Problem 3 and MILP problem 1, the full version of RM does not need to be used. For these problems it is enough to find any solution to the quadratic problem generated by RM. This solution can be optimised by holding the shape parameter constant giving a linear problem and then solving the MILP this generates. This simplified version of RM does not need to solve a quadratic problem and therefore has the best potential of competing with current solution techniques.

A weakness of RM is that it currently relies on a postulate that is only proven up to two dimensions. Further research is required to verify this postulate with mathematical certainty.

It was hoped that a method of this nature would be able to function as a trade-off between branch and cut, and metaheuristics. In its current form this is not the case, however, there appear to be opportunities to combine RM with metaheuristics. We feel that the concept of regions is well suited to metaheuristics.

Overall, we feel that the concept of using a region as a proxy for an integer solution is a valid proposition for a solution technique and should be further researched.

12 Appendices

12.1 Appendix A. Considered IP Models

12.1.1 Integer Problems

12.1.1.1 Integer Problem 1

Maximize $z = 4x + 12y$
s.t.
$4x + 15y \leq 59$
$7x + 5y \leq 53$
$-3x + 4y \leq 11$
$x, y\ int$

Solution: $x = 3, y = 3, z = 48$

12.1.1.2 Integer Problem 2

Max $z = 2x + 5y$
s.t.
$12x + 5y \leq 60$
$2x + 10y \leq 35$
$x, y\ int$

Solution: $x = 2, y = 3, z = 19$

12.1.1.3 Integer Problem 3

Max $z = 5x + 1y$
s.t.
$6x - 2y \leq 70$
$-3x + 11y \leq 32$
$10x + 5y \leq 55$
$13x + 6y \leq 23$
$x, y\ int$

Solution: $x = 7, y = -12, z = 23$

12.1.1.4 Integer Problem 4

Min $z = 7x + 9y$
s.t.
$11x + 7y \geq 87$
$3x + 11y \geq 53$
$5x - 4y \geq 17$
$x, y\ int$

Solution: $x = 7, y = 3, z = 76$

12.1.2 Binary Problems

12.1.2.1 Binary Problem 1

Max $z = 16x_1 + 22x_2 + 12x_3 + 8x_4$
s.t.
$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$
$x_i\ binary$

Solution: $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1, z = 42$

12.1.2.2 Binary Problem 2

Max $z = 130x_1 + 280x_2 + 70x_3$
s.t.
$x_2 \geq 0.5(x_1 + x_2 + x_3)$
$2.5x_1 + 6x_2 + 1.5x_3 \leq 2000$
$1.5x_1 + 2x_2 + x_3 \leq 800$
$1.5x_1 + 2x_2 + x_3 \leq 800 + 1000y$
$x_1 + 2.5x_2 + 0.5x_3 \leq 800 + 1000(1 - y)$
$x_i\ int$
$y\ bin$

Solution: $x_1 = 3, x_2 = 16, x_3 = 13, y = 0, z = 5780$

12.1.2.3 Binary Problem 3

Min $z = 3x_1 + 5x_2 + x_3 + 2x_4 + x_5 + 4x_6 + 3x_7 + x_8 + 2x_9 + 2x_{10}$
s.t.
$x_1 + x_2 + x_4 + x_5 + x_8 + x_9 \geq 1$
$x_1 + x_3 \geq 1$
$x_2 + x_5 + x_7 + x_{10} \geq 1$
$x_3 + x_6 + x_8 \geq 1$
$x_1 + x_2 + x_4 + x_6 + x_7 + x_9 + x_{10} \geq 1$
x_i bin

Solution: $x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 0, x_8 = 0, x_9 = 0, x_{10} = 1, z = 4$

12.1.3 MILPs

12.1.3.1 MILP Problem 1

Max $0z_1 + 120z_2 + 180z_3 + 201z_4 + 0.2x_2 + 0.8x_3$
s.t.
$0z_1 + 200z_2 + 350z_3 + 420z_4 + 0.5x_2 + 0.2x_3 \leq 420$
$0z_1 + 400z_2 + 700z_3 + 840z_4 + 2x_2 + 5x_3 \leq 840$
$0z_1 + 200z_2 + 350z_3 + 420z_4 + 0.5x_2 + x_3 \leq 420$
$x_3 \geq 10$
$z_1 + z_2 + z_3 + z_4 = 1$
$y_1 + y_2 + y_3 = 1$

$z_1 \leq y_1$
$z_2 \leq y_1 + y_2$
$z_3 \leq y_2 + y_3$
$z_4 \leq y_3$
$x_2, x_3, z_1, z_2, z_3, z_4 \geq 0$
$y_1, y_2, y_3\ bin$

Solution $x_2 = 0, x_3 = 28, z_1 = 0, z_2 = 0, z_3 = 1, z_4 = 0, y_1 = 0, y_2 = 0, y_3 = 1, z = 202.4$

Note that to allow for an orientation to be found, the problem had have the redundant constraint

$$-z_1 - z_2 - z_3 - z_4 \leq -1$$

Removed
This results in

Max $0z_1 + 120z_2 + 180z_3 + 201z_4 + 0.2x_2 + 0.8x_3$
s.t.
$0z_1 + 200z_2 + 350z_3 + 420z_4 + 0.5x_2 + 0.2x_3 \leq 420$
$0z_1 + 400z_2 + 700z_3 + 840z_4 + 2x_2 + 5x_3 \leq 840$
$0z_1 + 200z_2 + 350z_3 + 420z_4 + 0.5x_2 + x_3 \leq 420$
$-x_3 \leq -10$
$z_1 + z_2 + z_3 + z_4 \leq 1$
$y_1 + y_2 + y_3 \leq 1$
$-y_1 - y_2 - y_3 \leq -1$
$z_1 - y_1 \leq 0$
$z_2 - y_1 - y_2 \leq 0$
$z_3 - y_2 - y_3 \leq 0$
$z_4 - y_3 \leq 0$
$-x_2, -x_3, -z_1, -z_2, -z_3, -z_4 \leq 0$
$y_1, y_2, y_3\ bin$

Solution $x_2 = 0, x_3 = 28, z_1 = 0, z_2 = 0, z_3 = 1, z_4 = 0, y_1 = 0, y_2 = 0, y_3 = 1, z = 202.4$

12.1.3.2 MILP Problem 2

3. Max $3.5(x_1 + x_2 + x_3 + x_4) - 1.7x_1 - 2.3x_2 - 2.9x_3 - (0z_1 + 10000z_2 + 14500z_3 + 16500z_4) - 5000y_1 - 3000y_2 - 1000y_3 - (6000w_1 + 8000w_2 + 9000w_3) + 600(1 - v)$
s.t.
$x_1 + x_2 + x_3 + x_4 \leq 20000$
$x_1 \leq 8000y_1$
$x_2 \leq 9000y_2$
$x_3 \leq 6000y_3$
$x_4 \leq 10000v$
$z_1 + z_2 + z_3 + z_4 = 1$
$w_1 + w_2 + w_3 = 1$
$z_1 \leq w_1$
$z_2 \leq w_1 + w_2$
$z_3 \leq w_2 + w_3$
$z_4 \leq w_3$
$x_i \geq 0$
$z_i \geq 0$
$y_i\ bin$
$v\ bin$

$$y_1 = 1, y_2 = 0, y_3 = 1, v = 1, x_1 = 8000, x_2 = 0, x_3 = 2000, x_4 = 10000, z_1 = 1, z_2 = 0, z_3 = 0, z_4 = 0, w_1 = 1, w_2 = 0, w_3 = 0\ z = 38000$$

12.2 Appendix B. Code

https://github.com/JamieRobertOwen/RegionMethod

13 References

[1] 'Mixed-Integer Programming (MIP) - A Primer on the Basics', *Gurobi*. https://www.gurobi.com/resource/mip-basics/ (accessed Aug. 02, 2021).

[2] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming*, vol. 271. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-11008-0.

[3] J. P. Vielma, 'Mixed Integer Linear Programming Formulation Techniques', *SIAM Review*, vol. 57, no. 1, pp. 3–57, 2015.

[4] H. P. Williams, *Model Building in Mathematical Programming*. New York, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2013. Accessed: Aug. 16, 2021. [Online]. Available: http://ebookcentral.proquest.com/lib/qut/detail.action?docID=1120846

[5] R. Grimes and S. Jabbour, 'The DYNAMICS model for measuring dynamic operating benefits', Electric Power Research Inst., Palo Alto, CA (USA); Decision Focus, Inc …, 1989.

[6] J. Linderoth, 'Mixed-Integer (Linear) Programming: Recent Advances, and Future Research Directions', 2017. [Online]. Available: https://www.focapo-cpc.org/pdf/Linderoth.pdf

[7] E. L. Lawler and D. E. Wood, 'Branch-And-Bound Methods: A Survey', *Operations Research*, vol. 14, no. 4, pp. 699–719, 1966, doi: 10.1287/opre.14.4.699.

[8] T. Achterberg, R. Bixby, Z. Gu, E. Rothberg, and D. Weninger, 'Presolve Reductions in Mixed Integer Programming', *INFORMS Journal on Computing*, vol. 32, Nov. 2019, doi: 10.1287/ijoc.2018.0857.

[9] T. Achterberg and R. Wunderling, 'Mixed Integer Programming: Analyzing 12 Years of Progress', in *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*, M. Jünger and G. Reinelt, Eds. Berlin, Heidelberg: Springer, 2013, pp. 449–481. doi: 10.1007/978-3-642-38189-8_18.

[10] E. Klotz and A. M. Newman, 'Practical guidelines for solving difficult mixed integer linear programs', *Surveys in Operations Research and Management Science*, vol. 18, no. 1, pp. 18–32, Oct. 2013, doi: 10.1016/j.sorms.2012.12.001.

[11] M. W. P. Savelsbergh, ‘Preprocessing and Probing Techniques for Mixed Integer Programming Problems’, *ORSA Journal on Computing*, vol. 6, no. 4, pp. 445–454, Nov. 1994, doi: 10.1287/ijoc.6.4.445.

[12] M. Guzelsoy and T. K. Ralphs, ‘Duality for Mixed-Integer Linear Programs’, p. 34.

[13] A. M. Geoffrion, ‘Duality in Nonlinear Programming: A Simplified Applications-Oriented Development’, *SIAM Review*, vol. 13, no. 1, pp. 1–37, 1971.

[14] M. Yagiura and S. Umetani, ‘Relaxation Heuristics for the Set Covering Problem’, *Journal of the Operations Research Society of Japan*, vol. 50, no. 4, pp. 350–375, 2007.

[15] Oregon State University, ‘Penalty Methods’. http://web.engr.oregonstate.edu/~paasch/classes/me517/week7/penalty.html (accessed Sep. 06, 2021).

[16] K. Kemal, ‘Interior and Exterior Penalty Methods to solve Nonlinear Optimization Problems’, MPhil, Addis Ababa Unversity, 2017.

[17] K. Willcox, ‘Multidisciplinary System Design Optimization’, Massachusetts Institute of Technology. Accessed: Sep. 07, 2021. [Online]. Available: https://ocw.mit.edu/courses/institute-for-data-systems-and-society/ids-338j-multidisciplinary-system-design-optimization-spring-2010/lecture-notes/MITESD_77S10_lec08.pdf

[18] S. Kia, ‘Penalty Function Method’, University of California Irvine. Accessed: Sep. 07, 2021. [Online]. Available: http://solmaz.eng.uci.edu/Teaching/MAE206/Lecture14.pdf

[19] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*. Somerset, UNITED STATES: John Wiley & Sons, Incorporated, 2013. Accessed: Sep. 07, 2021. [Online]. Available: http://ebookcentral.proquest.com/lib/qut/detail.action?docID=1124000

[20] M. Fisher, ‘The Lagrangian Relaxation Method for Solving Integer Programming Problems’, *Manag. Sci.*, 2004, doi: 10.1287/mnsc.1040.0263.

[21] C. Deperrois (Grisoni), ‘Lagrangian relaxation can solve your optimization problem much, much faster’, *BCG GAMMA*, Feb. 03, 2020. https://medium.com/bcggamma/lagrangian-relaxation-can-solve-your-optimization-problem-much-much-faster-daa9edc47cc9 (accessed Sep. 13, 2021).

[22] J. Nocedal and S. J. Wright, Eds., ‘Penalty and Augmented Lagrangian Methods’, in *Numerical Optimization*, New York, NY: Springer, 2006, pp. 497–528. doi: 10.1007/978-0-387-40065-5_17.

[23] A. H. Land and A. G. Doig, 'An Automatic Method of Solving Discrete Programming Problems', *Econometrica*, vol. 28, no. 3, p. 497, Jul. 1960, doi: 10.2307/1910129.

[24] J. Clausen, 'Branch and Bound Algorithms-Principles and Examples', *Department of Computer Science, University of Copenhagen*, pp. 1–30, 2003.

[25] S. Boyd and J. Mattingley, 'Branch and bound methods', *Notes for EE364b, Stanford University*, pp. 2006–07, 2007.

[26] S. Haseen, S. Sadia, A. Bari, and Q. Ali, 'Integer Programming: NAZ cut and A-T cut', *International Journal of Engineering Science and Technology*, vol. 06, pp. 128–137, Apr. 2014.

[27] A. Bounceur, M. Bezoui, and R. Euler, *Boundaries and Hulls of Euclidean Graphs: From Theory to Practice*. New York: Chapman and Hall/CRC, 2018. doi: 10.1201/9781315169897.

[28] R. E. Gomory, 'Outline of an algorithm for integer solutions to linear programs', *Bull. Amer. Math. Soc.*, vol. 64, no. 5, pp. 275–279, Sep. 1958, doi: 10.1090/S0002-9904-1958-10224-4.

[29] G. Cornuéjols, 'Revival of the Gomory cuts in the 1990's', *Ann Oper Res*, vol. 149, no. 1, pp. 63–66, Feb. 2007, doi: 10.1007/s10479-006-0100-1.

[30] R. E. Gomory, 'Outline of an Algorithm for Integer Solutions to Linear Programs and An Algorithm for the Mixed Integer Problem', in *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Berlin, Heidelberg: Springer, 2010, pp. 77–103. doi: 10.1007/978-3-540-68279-0_4.

[31] 'MIT15_083JF09_lec17.pdf'. Accessed: Aug. 11, 2021. [Online]. Available: https://ocw.mit.edu/courses/sloan-school-of-management/15-083j-integer-programming-and-combinatorial-optimization-fall-2009/lecture-notes/MIT15_083JF09_lec17.pdf

[32] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, 'A survey on metaheuristics for stochastic combinatorial optimization', *Nat Comput*, vol. 8, no. 2, pp. 239–287, Jun. 2009, doi: 10.1007/s11047-008-9098-4.

[33] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, 'A unified solution framework for multi-attribute vehicle routing problems', *European Journal of Operational Research*, vol. 234, no. 3, pp. 658–673, May 2014, doi: 10.1016/j.ejor.2013.09.045.

[34] R. L. Burdett, P. Corry, C. Eustace, and S. Smith, ‘A flexible job shop scheduling approach with operators for coal export terminals – A mature approach’, *Computers & Operations Research*, vol. 115, p. 104834, Mar. 2020, doi: 10.1016/j.cor.2019.104834.

[35] R. L. Burdett and E. Kozan, ‘A sequencing approach for creating new train timetables’, *OR Spectrum*, vol. 32, no. 1, pp. 163–193, Jan. 2010, doi: 10.1007/s00291-008-0143-6.

[36] E. Zunic, A. Beširević, R. Skrobo, H. Hasić, K. Hodžić, and A. Djedović, ‘Design of Optimization System for Warehouse Order Picking in Real Environment’, Oct. 2017. doi: 10.1109/ICAT.2017.8171630.

[37] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, ‘Metaheuristic research: a comprehensive survey’, *Artif Intell Rev*, vol. 52, no. 4, pp. 2191–2233, Dec. 2019, doi: 10.1007/s10462-017-9605-z.

[38] X.-S. Yang, *Nature-inspired metaheuristic algorithms*, 2. ed. Frome: Luniver Press, 2010.

[39] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. New York: Cambridge University Press, 2011.

[40] K. Jain and V. V. Vazirani, ‘Approximation Algorithms for Metric Facility Location . . .’, *J. ACM*, vol. 48, no. 2, pp. 274–296, Mar. 2001, doi: 10.1145/375827.375845.

[41] T. Berthold, ‘RENS-relaxation enforced neighborhood search’, Aug. 2009.

[42] M. X. Goemans and D. P. Williamson, ‘The primal-dual method for approximation algorithms and its application to network design problems’, in *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, Ed. USA: PWS Publishing Co., 1996.

[43] C. Musco, A. Vakilian, T. Kolokotrones, M. Sridhar, and V. Tjeng, ‘Lecture : LP Relaxations for Combinatorial Relaxation’, p. 11.

[44] C. Neumann, O. Stein, and N. Sudermann-Merx, ‘A feasible rounding approach for mixed-integer optimization problems’, *Comput Optim Appl*, vol. 72, no. 2, pp. 309–337, Mar. 2019, doi: 10.1007/s10589-018-0042-y.

[45] C. Neumann and O. Stein, ‘Feasible rounding approaches for equality constrained mixed-integer optimization problems’, *Optimization*, pp. 1–26, Oct. 2021, doi: 10.1080/02331934.2021.1981894.

[46] J. Sherman and W. J. Morrison, ‘Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or a Given Row of the Original Matrix’, *The Annals of Mathematical Statistics*, vol. 20, no. 4, pp. 620–624, Dec. 1949, doi: 10.1214/aoms/1177729959.

[47] J. Ding and A. Zhou, ‘Eigenvalues of rank-one updated matrices with some applications’, *Applied Mathematics Letters*, vol. 20, no. 12, pp. 1223–1226, Dec. 2007, doi: 10.1016/j.aml.2006.11.016.

[48] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, ‘Julia: A Fresh Approach to Numerical Computing’, *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, Jan. 2017, doi: 10.1137/141000671.

[49] I. Dunning, J. Huchette, and M. Lubin, ‘JuMP: A Modeling Language for Mathematical Optimization’, *SIAM Rev.*, vol. 59, no. 2, pp. 295–320, Jan. 2017, doi: 10.1137/15M1020575.

[50] S. G. Johnson, *The NLopt nonlinear-optimization package*. [Online]. Available: http://ab-initio.mit.edu/nlopt

[51] M. J. D. Powell, ‘A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation’, in *Advances in Optimization and Numerical Analysis*, S. Gomez and J.-P. Hennart, Eds. Dordrecht: Springer Netherlands, 1994, pp. 51–67. doi: 10.1007/978-94-015-8330-5_4.

[52] J. W. S. Cassels, *An Introduction to the Geometry of Numbers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. Accessed: Apr. 22, 2022. [Online]. Available: https://doi.org/10.1007/978-3-642-62035-5

[53] M. Lubin, ‘Mixed-integer convex optimization: outer approximation algorithms and modeling power’, p. 143, Aug. 2017.