A Bilevel Model for the Minimum Sudoku Clue Problem

vorgelegt von
Gennesaret Kharistio Tjusila
Berlin

der Fakultät II - Mathematik und Naturwissenschaften
der Technische Universität Berlin
zur Erlangung des akademisches Grades

Bachelor Of Science
- B. Sc. -

Prüfer: Prof. Dr. Thorsten Koch
Technische Universität Berlin
Prof. Dr. Sebastian Pokutta
Technische Universität Berlin

Berlin, den 08.08.2022

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 08.08.2022

Gennesaret Kharistio Tjusila

Zusammenfassung

Es ist bekannt, dass jedes beliebige 9-mal-9-Sudoku-Rätsel mindestens 17 Hinweise haben muss, um eine eindeutige Lösung zu haben. In dieser Arbeit wird die spezifische Frage untersucht: Was ist die maximale Anzahl von Einträgen, die wir von einem vollständigen Sudoku-Brett entfernen können, sodass das resultierende Sudoku-Rätsel eine eindeutige Lösung hat? Wir nennen dieses Problem das "Minumum Sudoku Clue" (MSC) Problem. In dieser Arbeit formulieren wir das MSP Problem als ein ganzzahliges, zweistufiges, lineares Optimierungsproblem (*Integer Bilevel Linear Program*) und verwenden ein *Branch-and-Cut*-Schema mit *Intersection Cuts*, um zweistufigen Optimierungsproblem-Instanzen zu lösen. Wir präsentieren auch global gültige Schnittebenen, die auf den Wurzelknoten anwendbar sind, bevor wir das *Branch-and-Cut*-Schema einführen. Wir nennen diese Schnittebenen (*Unavoidable Set Cuts*). Schließlich präsentieren wir rechnerisch, dass unser Modell zur Lösung von MSP taugt. Wir lösen 19 der 20 Instanzen, des von uns generierten 9 x 9-Sudoku-Bretts, bis zur Optimalität. Wir untersuchen auch die Variation der Anzahl der *Unavoidable Set Cuts*, die am Wurzelknoten angewendet werden, und die Verwendung von *Coupling Constraints* in der Formulierung des Problems. Insbesondere zeigen wir, dass die Anwendung einer moderaten Anzahl von Schnittebenen besser ist als eine extreme Anzahl und dass die Formulierung mit *Coupling Constraints* schneller eine Lösung findet als eine Formulierungen ohne *Coupling Constraint*, trotz der höheren *Final Gap* in den Instanzen, die nicht gelöst wird.

Abstract

It is well established that any given 9 by 9 sudoku puzzle needs to have at least 17 clues to possess a unique solution. This thesis investigates the more specific question: given a completed sudoku board, what is the maximum number of entries that we can remove from this board such that the resulting sudoku puzzle has a unique solution? We call this problem the "minimum sudoku clue" (MSC) problem. In this thesis, we formulate MSC as an integer bilevel linear programming problem and use a branch and cut scheme with intersection cuts to solve the bilevel instances. We also present a set of globally valid cutting planes applicable on the root node before initiating the branch and cut scheme. We call this set of cuts "unavoidable set cuts". Lastly, we include a computational study that shows the viability of our bilevel model in solving MSC. We solve to optimality 19 out of the 20 instances of the 9 by 9 sudoku board we generated. We also investigate the effect of varying the number of "unavoidable set cuts" applied at the root node and the effect of using coupling constraints in the formulation. In particular, we show that applying a moderate amount of cuts outperforms applying an extreme amounts of cuts, and that the formulation with coupling constraint solves faster than formulations without, albeit with a higher final gap on instances which do not solve.

Acknowledgements

I always thought that writing a bachelor's thesis would be stressful. To no one surprise, it is, but I never imagined it to be fun, and yet what a fun experience it has been indeed. I get to learn a lot and discover areas of mathematics that are very interesting that I would never have touched on had I not decided to write a thesis on this topic. Thus, I would like to first thank the people who helped me formulate this thesis topic, Professor Koch for hosting the seminar that gave rise to this thesis topic and allowing me to do a thesis on this particular topic, and Kai Hoppmann-Baum for helping me formulate my initial bilevel model for the minimum sudoku clue problem. It was never in my wildest dream that I would write a thesis about something as fun as sudoku puzzles. I also thank Markus Sinnl for providing me a license for their bilevel solver.

Most of all, I want to thank Mark Turner for being a super nice supervisor. Without you, this thesis would never have been finished. Thanks for guiding me to be a better mathematical writer, for the constant support when I stress myself out over this thesis, and for the constant reminder to make sure I learn new stuff and have fun as I write this thesis.

I also thank my colleagues, Gregorius Kevin, Michael Adipoetra, and Kristoforus Jason, who helped proofread this thesis and gave me very valuable insights. In particular, I thank Gerry for going through the burden of reading more than 50% of this thesis and helping me fix my grammatical mistakes.

Lastly, I thank my best friends, Georgius Kenneth, Nicole Charlene, and Owen Henson, along with my family, who have accompanied me throughout this journey and are always there to support me mentally and emotionally when I am stressed out. I am very lucky indeed to have such great accompaniments.

Contents

Introduction

A *sudoku puzzle* is a 9×9 grid with empty and non-empty cells. The grid is divided into nine subgrids, each of the size 3×3. The goal of the puzzle is to fill all cells with numbers from one to nine such that each number appears exactly once in each row, column, and subgrid. An example of a sudoku puzzle is provided in Figure 1. The non-empty cells of a sudoku puzzle are called clues. The first sudoku appeared in the May 1979 edition of Dell Pencil Puzzles and Word Games [12]. It was originally called "Number Place" before a Japanese magazine published it as "sudoku" in 1984.

Sudoku puzzles that have a unique solution are called valid puzzles. There exist 9×9 sudoku puzzles with 77 clues that have multiple solutions, but any puzzle with at least 78 clues will always have a unique solution. The question "what is the minimum number of clues on any valid puzzle" is a bit more elusive. McGuire et al. showed in [32] that there are no valid sudoku puzzles with 16 or less clues. Gordon Royle has kept a list of nearly 50000 sudoku puzzles with 17 clues. The result is proven with computer assistance, a proof method most famously used by Appel and Haken to prove the *four color theorem* [2]. As Cooper and Kirkpatrick pointed out in [9], verification of the results by McGuire is desirable as unlike a pure theoretical proof, computer-assisted proofs may be prone to errors.

This thesis investigates the adjacent question, "given a completed sudoku grid, what is the minimum number of clues that any valid puzzle whose unique solution is the given grid has". We call this problem the minimum sudoku clue (MSC) problem. For the 50000 sudoku puzzles with 17 clues that are maintained by Gordon Royle, the answer to this question is 17 (as McGuire shows that there are no valid sudoku puzzles with 16 clues or less). However, there are around 6.671×10^{21} possible completed sudoku grids, and, as we will see later in this thesis, for most of them, there are no valid puzzles with 17 clues. We note that the results we aim for in this thesis are sharper than the results by Gary McGuire, which only give a lower bound to the answer of MSC for any sudoku grid. In particular, solving MSC for all sudoku grids will directly verify the result by Gary McGuire.

To solve this problem, we formulate the MSC problem as a bilevel optimization problem. Optimization problems involving two decision-makers who make their decision in a hierarchical manner. The first bilevel models are Stackelberg games which appear in the field of economics [43]. A leader makes a decision that optimizes their objective and is then responded to by a follower, which optimizes their objective. The leader is assumed to have full knowledge of the follower's response and thus make their decision in anticipation of the follower's decision. The main advantage of modelling MSC as bilevel is that it allows the use of off-the-shelf general bilevel solvers to solve MSC. This is in stark contrast to the existing software created by the sudoku enthusiast community that are developed ad hoc. The drawback, however, is that bilevel is in general hard to solve, and even state-of-the-art bi-level solvers are not efficient enough to be used to investigate all sudoku grids.

This thesis consist of four chapters: In the first chapter, we will discuss linear programming and integer linear programming. The chapter will also act as a mathematical preliminary for this thesis work. Integer linear programs play a crucial role in this thesis as both our leader and follower optimization problems can be modelled as integer linear programs. We will highlight important theorems about linear and integer linear programming and discuss methods to solve them.

The next three chapters form the main contribution of this thesis. In the second chapter, we summarize existing literature on bilevel problems. We show some examples of bilevel problems and highlight features of bilevel optimization problems which make them hard to solve. In particular, we look at integer bilevel linear programs (IBLP), which are bilevel optimization problems in which the leader and follower optimization problems are integer linear programs.

9	8		7			6		
7	5			4				
		3			8		7	
5					7		3	
		9	4					
				2		1		
3								1
	9				5		8	
		5	2					6

9	8	1	7	5	2	6	4	3
7	5	2	6	4	3	8	1	9
4	6	3	1	9	8	2	7	5
5	1	4	8	6	7	9	3	2
6	2	9	4	3	1	7	5	8
8	3	7	5	2	9	1	6	4
3	4	8	9	7	6	5	2	1
2	9	6	3	1	5	4	8	7
1	7	5	2	8	4	3	9	6

Figure 1: A Sudoku Puzzle Along With Its Solution

In the third chapter, we formalize the MSC problem and model them as IBLP. We also introduce a set of cuts that can be applied at the root node to speed up the solving process. We will call these sets of cuts unavoidable set cuts.

In the fourth chapter, we present computational results that show the viability of our proposals in chapter 3 to solve MSC problems. We show that out of the sampled 20 sudoku grids of size 9×9, we are able to find an optimal solution to MSC on 19 of the grids.

Chapter 1

Linear and Integer Linear Programming

This chapter studies *linear programs* (LP), problems of maximizing or minimizing a linear function over the feasible set of a system of linear inequalities. Materials throughout Subsections 1.1–1.5 are usually covered of as a part of a course on linear programming or integer linear programming. Readers who are familiar with linear and integer linear programming may directly skip to Subsection 1.6. In Subsection 1.1, we investigate the feasible set of a system of linear inequalities and provide an algebraic method to find feasible solutions to a system of linear inequalities. In the next three subsections, we present LPs and important theorems about LPs. In Subsection 1.2, we define LP and its various equivalent forms. In Subsection 1.3, we discuss an essential feature of LP known as *duality*. In Subsection 1.4, we present an algorithm to solve LP, known as the *simplex algorithm*. Finally, in the last two subsections of this chapter we discuss integer linear programs (ILP), LPs with the additional constraint that all variables must be integral. In Subsection 1.5, we discuss ILP and introduce three methods for solving ILP: branch and bound, cutting plane, and branch and cut. In Subsection 1.6, we introduce intersection cuts, a class of cutting planes which will play a crucial role in this thesis work.

We assume that the reader has basic knowledge of linear algebra. As background reference for linear algebra, we mention Strang [40] and Lang [31]. This chapter is loosely based on the book by Conforti et al. [8]. For a more in-depth treatment of linear programming and integer linear programming, we mention the comprehensive books by Schrijver [39], and Korte and Vygen [29]. A more elementary treatment of the topic can be found in the work of Bertsimas [4], Papadimitriou and Steiglitz [36]. Finally, we mention the PhD thesis of Achterberg [1], from which we drew a lot of inspiration in writing this chapter.

1.1 Fourier Elimination

In this section, we investigate systems of linear inequalities. Consider a general system of linear inequalities

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\le b_1,\\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\le b_2,\\
&\vdots\\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\le b_m.
\end{aligned} \tag{1.1}$$

or equivalently $Ax \le b$ with

$$A := \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad \text{and} \quad b := \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

We call a vector $x \in \mathbb{R}^n$ a *feasible solution* of the system of linear inequalities if it satisfies all inequalities in the system. The set of all feasible solutions is called the *feasible set*. The system of linear inequalities is *feasible* if the system has at least one feasible solution and *infeasible* otherwise.

We introduce the *Fourier elimination* method to find a feasible solution to a system of linear inequalities if any exist. We consider a general system of linear inequalities as given in (1.1). Let $I := \{1, \dots, m\}$. We define the index sets

$$I^+ := \{i \in I \mid a_{in} > 0\},\ I^- := \{i \in I \mid a_{in} < 0\} \text{ and } I^0 := \{i \in I \mid a_{in} = 0\}.$$

For each $i \in I$, we divide the ith inequality by $|a_{in}|$. As a result, we have the following system of linear inequalities

$$\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|} x_j + x_n \leq \frac{b_i}{|a_{in}|}, \quad \forall i \in I^+ \tag{1.2a}$$

$$\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|} x_j - x_n \leq \frac{b_i}{|a_{in}|}, \quad \forall i \in I^- \tag{1.2b}$$

$$\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|} x_j \qquad \leq \frac{b_i}{|a_{in}|}, \quad \forall i \in I^0 \tag{1.2c}$$

Notice that the system of linear inequalities (1.2) is equivalent to the system (1.1), that is, a vector is a feasible solution to the system (1.1) if and only if it is a feasible solution to the system (1.2).

We now define a new system of linear inequalities as follows

$$\sum_{j=1}^{n-1} \left(\frac{a_{pj}}{|a_{pn}|} + \frac{a_{qj}}{|a_{qn}|} \right) x_j \leq \frac{b_p}{|a_{pn}|} + \frac{b_q}{|a_{qn}|}, \quad \forall p \in I^+, q \in I^- \tag{1.3a}$$

$$\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|} x_j \leq \frac{b_i}{|a_{in}|}, \qquad \forall i \in I^0. \tag{1.3b}$$

The inequalities in (1.3a) are obtained by summing up pairs of inequalities from (1.2a) and (1.2b). Note that for all of the inequalities in the system (1.3), the coefficient of x_n is zero. The variable x_n is thus 'eliminated' from the system of linear inequalities. We remark that each inequality in the system (1.3) is a non-negative linear combination of the inequalities in the system (1.1), that is, for all inequality $\alpha x \leq \beta$ in the system (1.3), there exists a $y \in \mathbb{R}^m$ such that $y \geq 0$, $y^T A = \alpha$ and $y^T b = \beta$. Furthermore, by swapping the variables, we can use the method described to eliminate any variable x_i with $i \in \{1, \dots, n\}$.

Theorem 1.1 (Fourier Elimination [21]). *A vector $(\bar{x}_1, \dots, \bar{x}_{n-1}) \in \mathbb{R}^{n-1}$ is a feasible solution to the system (1.3) if and only if there exist an $\bar{x}_n \in \mathbb{R}$ such that $(\bar{x}_1, \dots, \bar{x}_{n-1}, \bar{x}_n)$ is a feasible solution to the system (1.1).*

Proof. We first show that the condition is necessary. Let $\bar{x} = (\bar{x}_1, \dots, \bar{x}_{n-1}, \bar{x}_n) \in \mathbb{R}^n$ be a feasible solution to the system (1.1). Then it is also a feasible solution to the system (1.3) as every inequality in the system (1.3) is a non-negative linear combination of inequalities from (1.1). To show sufficiency, we rewrite the inequalities in (1.3a) as

$$\sum_{j=1}^{n-1} \frac{a_{qj}}{|a_{qn}|} x_j - \frac{b_q}{|a_{qn}|} \leq \frac{b_p}{|a_{pn}|} - \sum_{j=1}^{n-1} \frac{a_{pj}}{|a_{pn}|} x_j, \quad \forall p \in I^+, q \in I^-.$$

Let $(\bar{x}_1, \dots, \bar{x}_{n-1})$ be a feasible solution to the system (1.3). We define,

$$l := \max_{i \in I^-} \left(\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|} \bar{x}_j - \frac{b_i}{|a_{in}|} \right) \text{ and } u := \min_{i \in I^+} \left(\frac{b_i}{|a_{in}|} - \sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|} \bar{x}_j \right).$$

We have $l \leq u$. Letting $\bar{x}_n := l$, we have

$$\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|} \bar{x}_j - \frac{b_i}{|a_{in}|} \leq \bar{x}_n, \quad \forall i \in I^-$$

which is equivalent to

$$\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|}\bar{x}_j - \bar{x}_n \leq \frac{b_i}{|a_{in}|}, \quad \forall i \in I^-$$

and

$$\bar{x}_n \leq \frac{b_i}{|a_{in}|} - \sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|}\bar{x}_j, \quad \forall i \in I^+$$

which is equivalent to

$$\sum_{j=1}^{n-1} \frac{a_{ij}}{|a_{in}|}\bar{x}_j + \bar{x}_n \leq \frac{b_i}{|a_{in}|}, \quad \forall i \in I^+.$$

We see that $(\bar{x}_1, \ldots, \bar{x}_{n-1}, \bar{x}_n)$ is a feasible solution to the system (1.2) and thus to the system (1.1). □

Example 1.2. *Consider the following system of linear inequalities*

$$\begin{aligned} -x_1 - x_2 &\leq -8, \\ 3x_1 - x_2 &\leq 16, \\ -x_1 + x_2 &\leq 0. \end{aligned} \tag{1.4}$$

Using Fourier elimination to eliminate the variable x_1 yields the following system of linear inequalities

$$\begin{aligned} -\frac{4}{3}x_2 &\leq -\frac{8}{3} \\ \frac{2}{3}x_2 &\leq \frac{16}{3} \end{aligned}$$

which is equivalent to

$$2 \leq x_2 \leq 8.$$

Using Fourier elimination to eliminate the variable x_2 yields the following system of linear inequalities

$$\begin{aligned} 2x_1 &\leq 16 \\ -2x_1 &\leq -8 \end{aligned}$$

which is equivalent to

$$4 \leq x_1 \leq 8.$$

We illustrate this example in Figure 1.1. The feasible set to the system (1.4) is shaded in blue. Denoting with S the feasible set to the system (1.4) and S_i the feasible set to the system of linear inequalities that we get by eliminating x_i with $i \in \{1, 2\}$. We see that S_i is the projection of S onto the axis x_i for $i \in \{1, 2\}$.

Given a system of linear inequalities $Ax \leq b$ with n variables, we can use Fourier elimination repeatedly until we get a system of linear inequalities $A'x \leq b'$ with one variable. Finding a feasible solution to the system $A'x \leq b'$ is trivial, since the system contains only upper- and lower-bounds on the variable x_1. We can then use a solution for $A'x \leq b'$ to generate a feasible solution to the system $Ax \leq b$ according to Theorem 1.1 (or by backward substitution).

Each inequality in the system $A'x \leq b'$ has the form $\bar{a}x_1 \leq \bar{b}$ where we can assume without loss of generality that $\bar{a} \in \{-1, 1\}$. Using Fourier elimination one extra time to eliminate x_1 yields inequalities of the form $0 \leq \hat{b}$. If for any of these inequalities $\hat{b} < 0$, then $A'x \leq b'$ is infeasible. To see this, let $x_1 \leq \bar{b}_1$ and $-x_1 \leq \bar{b}_2$ be two inequalities in the system $A'x \leq b'$ such that $\bar{b}_1 + \bar{b}_2 = \hat{b} < 0$, that is, a pair of inequalities which generates the inequality $0 \leq \hat{b}$. Then, it applies that any feasible solution to $A'x \leq b'$ must satisfy

$$x_1 \leq \bar{b}_1 \text{ and } x_1 \geq -\bar{b}_2 > \bar{b}_1$$

which implies that the system $A'x \leq b'$ has no feasible solution and so neither does $Ax \leq b$. Conversely, if the system $A'x \leq b'$ is infeasible, then there exist inequalities of the form $-x_1 \leq \bar{b}_1$ and $x_1 \leq \bar{b}_2$ (possibly after division by a scalar) with $-\bar{b}_1 > \bar{b}_2$. Adding the two inequalities yields the inequality

$$0 \leq \hat{b}$$

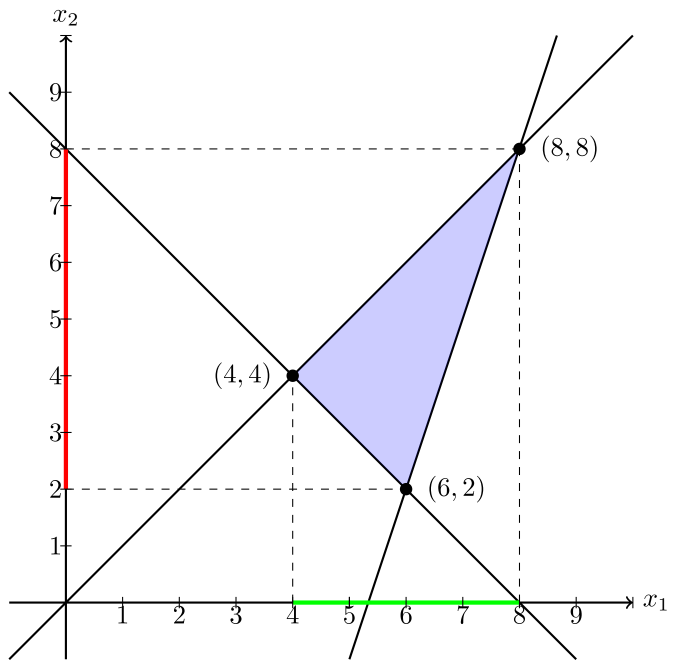


Figure 1.1: Illustration for Example 1.2

with

$$\hat{b} := \bar{b}_1 + \bar{b}_2 < \bar{b}_1 - \bar{b}_1 = 0.$$

This observation motivates the following theorem known as *Farkas' Lemma* [17].

Theorem 1.3 (Farkas' Lemma)**.** *A system of linear inequalities $Ax \leq b$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ has a feasible solution if and only if there exist no $y \in \mathbb{R}^m$ with $y^T b < 0$, $y^T A = 0$, and $y \geq 0$.*

Proof. We first show the sufficient part of the proof. Let $x^* \in \mathbb{R}^n$ be a feasible solution to the system $Ax \leq b$. For all $y \in \mathbb{R}^m$ with $y^T A = 0$ and $y \geq 0$, it applies

$$y^T b \geq y^T A x^* = 0.$$

To show necessity, we do a proof by contraposition. Suppose that the system $Ax \leq b$ is infeasible, then eliminating $x_2, \ldots, x_n$ using Fourier elimination yields the system $A'x \leq b'$ involving only the variable x_1. Since $Ax \leq b$ is infeasible, neither is $A'x \leq b'$ by Theorem 1.1. By the remark we made above, eliminating x_1 from the system $A'x \leq b'$ yields a system of inequalities, one of which has the form

$$0 \leq \hat{b}$$

with $\hat{b} < 0$. Additionally, we recall that the inequalities in the system $A'x \leq b'$ are obtained by taking a non-negative linear combination of inequalities in the system $Ax \leq b$. Thus, the inequality $0 \leq \hat{b}$ can be obtained by taking a non-negative linear combination of inequalities in the system $Ax \leq b$. This implies that there exists a $y \in \mathbb{R}^m$ such that $y \geq 0$, $y^T A = 0$, and $y^T b = \hat{b} < 0$. □

Theorem 1.4 (Farkas' Lemma Variant)**.** *A system of linear inequalities $Ax = b, x \geq 0$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ has a feasible solution if and only if $y^T b \leq 0$ for all $y \in \mathbb{R}^m$ with $y^T A \leq 0$*

Proof. Let x^* be a feasible solution to the system $Ax = b$, $x \geq 0$. Then for all $y \in \mathbb{R}^m$ with $y^T A \leq 0$, it applies

$$y^T b = y^T A x^* \leq 0 x^* = 0.$$

To show the converse direction, we use proof by contraposition. Suppose that the system $Ax = b, x \geq 0$ has no feasible solutions. Then, the system $Ax \leq b, -Ax \leq -b, -x \leq 0$ also has no feasible solutions. By Theorem 1.3, there exist vectors $u, v, w \in \mathbb{R}^m$ with $u, v, w \geq 0$ such that

$$\begin{bmatrix} u & v & w \end{bmatrix}^T \begin{bmatrix} A \\ -A \\ -I \end{bmatrix} = 0 \text{ and } \begin{bmatrix} u & v & w \end{bmatrix}^T \begin{bmatrix} b \\ -b \\ 0 \end{bmatrix} < 0.$$

By matrix multiplication, we have

$$u^T A - v^T A - w^T I = 0 \text{ and } u^T b - v^T b < 0$$

Letting $y := v - u$, we get

$$y^T A = -w^T I \leq 0 \text{ and } y^T b > 0$$

as desired. □

Example 1.5. *We present a geometric interpretation of Theorem 1.4. We consider the system of inequality*

$$\begin{aligned} x_1 + 2x_2 &= 1 \\ 2x_1 + x_2 &= 8 \\ x &\geq 0 \end{aligned}$$

The system is infeasible because the unique solution to the system

$$\begin{aligned} x_1 + 2x_2 &= 1 \\ 2x_1 + x_2 &= 8 \end{aligned}$$

is $x^* = (5, -2)$. We consider the set $\mathcal{S}$ of vector b such that the solution $x \in \mathbb{R}^2$ to the system

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} x = b$$

is non-negative. The set of vector in $\mathcal{S}$ is shaded blue in Figure 1.2. The vector $b = (1, 8)$ is shown as a green dot. Consider the vector $y = (-4, 1)$. The set of points such that $y^T x = 0$ are shown in red. It applies

$$y^T b = 4 > 0 \textit{ and } y^T \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} -2 & -6 \end{bmatrix} \leq 0$$

We see visually all the vectors of the set $\mathcal{S}$ lies on the right side of the line $y^T x = 0$ while the point b lies on the left side of the line.

1.2 Linear Programming

Given a matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, the linear program (LP) in *general form* is defined as

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned}$$

The LP is *infeasible*, if the set $\{x \in \mathbb{R}^n | Ax \leq b\}$ is empty, and *unbounded*, if for every $x \in \mathbb{R}^n$ with $Ax \leq b$ there exists an $x' \in \mathbb{R}^n$ such that $c^T x < c^T x'$ and $Ax' \leq b$. We introduce some basic linear programming terminologies:

(i) The set $X_{LP} := \{x \in \mathbb{R}^n | Ax \leq b\}$ is called the *feasible set*.

(ii) Vectors in the feasible set are called *feasible solutions*.

(iii) Variables in the vector x are called *decision variables*.

(iv) The function $c^T x$ is called the *objective function*.

(v) The value $z := \max\{c^T x \mid Ax \leq b,\, x \in \mathbb{R}^n\}$ is called the *optimum value*.

(vi) A feasible solution x^* with $c^T x^* = z$ is called an *optimum solution*.

(vii) Each inequality in the system of linear inequalities $Ax \leq b$ is called a *constraint*.

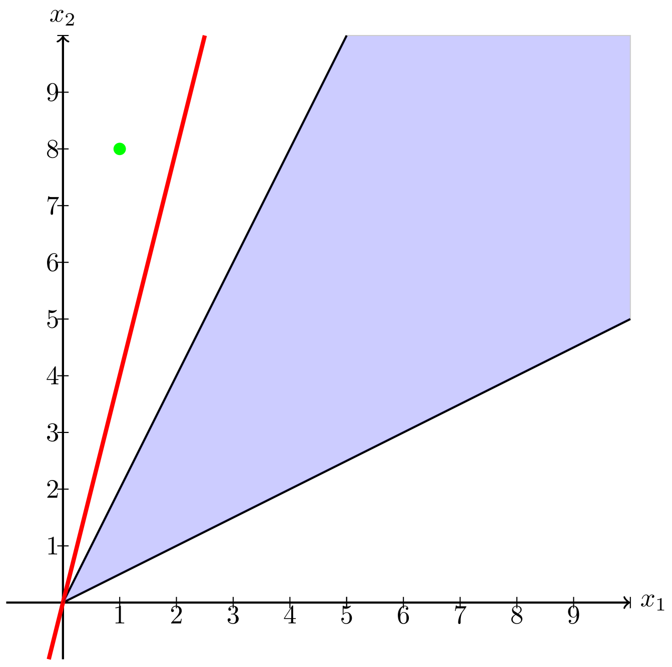


Figure 1.2: Illustration for Example 1.5

The following example illustrates an application of linear programming to real-world problems.

Example 1.6. *A coin factory produces two types of coins by mixing gold and silver. A single coin of type A requires 4 grams of silver and 1 gram of gold to produce, while a single coin of type B requires 3 grams of silver and 2 grams of gold. The company earns a profit of* \$10 *and* \$15 *for each coin of type A, and B produced, respectively. The factory currently has* 32 *kg of silver and* 13 *kg of gold as raw materials. Determine the number of each type of coin that the factory should produce to earn the maximum profit. For the purposes of this example, we assume that the factory can produce fractional coins.*

We denote with x_1 and x_2 the number of coins of types A and B to be made. These are the decision variables of our linear program (LP). We aim to maximize the factory's profit, that is, we try to maximize the following objective function

$$f(x_1, x_2) := 10x_1 + 15x_2.$$

The combinations of coins that we can make are limited by certain constraints*. For example, we cannot produce as many coins as we want because we only have a finite amount of material. If we were to produce x_1 coins of type A and x_2 coins of type B, we would need* $4x_1 + 3x_2$ *of silver. This amount must not exceed the 32 kg limit that we have. Thus, we have the constraint*

$$4x_1 + 3x_2 \leq 32000. \tag{1.5}$$

In the same way, we obtain another constraint that limits the amount of gold we can use

$$x_1 + 2x_2 \leq 13000. \tag{1.6}$$

Lastly, we have $x_1 \geq 0$ and $x_2 \geq 0$ because we cannot produce negative quantities. We call a constraint of the form $x \geq 0$ ***non-negativity constraint****. When combined, we have the following LP to represent the problem described.*

$$\max_{x_1,x_2\in\mathbb{R}} \quad 10x_1 + 15x_2 \tag{1.7a}$$

$$\text{subject to} \quad 4x_1 + 3x_2 \leq 32000 \tag{1.7b}$$

$$x_1 + 2x_2 \leq 13000 \tag{1.7c}$$

$$x_1, x_2 \geq 0. \tag{1.7d}$$

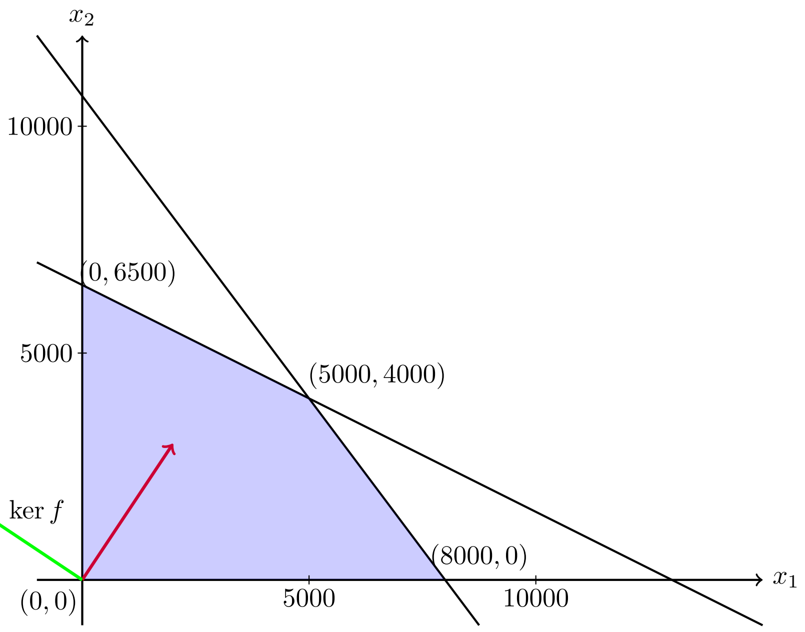


Figure 1.3: The Feasible Set of the LP (1.7)

The feasible solutions to LP (1.7) *are the vectors in* $\mathbb{R}^2$ *that satisfy* (1.7b)–(1.7d)*, this set is shaded blue in Figure 1.3. Observe that the feasible set of our LP is exactly the set of all production combinations which we have enough material to produce. The LP in* (1.7) *is feasible. We have an infeasible LP when, for example, we have no material at all, yet the factory boss still asks us to produce at least one coin. There is no feasible solution to such a linear program that satisfies all constraints, that is, that would both satisfy our material constraint and please our boss.*

Finally, we note that our objective function is a linear mapping from $\mathbb{R}^2$ *to* $\mathbb{R}$ *defined by the vector*

$$c := \begin{bmatrix} 10 & 15 \end{bmatrix}.$$

The green vector in Figure 1.3 is a basis of the null space of c*. From linear algebra, we know that moving in this direction does not change the value of our objective function. However, when we move in a direction that is not within the null space, such as the red vector in Figure 1.3, the value of our objective function changes. Further, we note that moving in the direction of the red vector increases the value of our objective function. We see that we can gain maximum profit by producing* 5000 *coins type A and* 4000 *coins type B with a profit of* \$110000*. Thus, the vector* $(5000, 4000) \in \mathbb{R}^2$ *is an optimum solution to our LP.*

Proposition 1.7. *Let* $\max\{c^T x \mid Ax \leq b,\ x \in \mathbb{R}^n\}$ *be an LP in general form with*

$$M := \sup\{c^T x \mid Ax \leq b,\ x \in \mathbb{R}^n\} < \infty$$

then there exists an $x^* \in \mathbb{R}^n$ *with* $Ax^* \leq b$ *such that* $c^T x^* = M$*.*

Proof. The proof for the proposition will be given in Section 1.3 □

Corollary 1.8. *An LP is either infeasible, unbounded or have an optimum solution.*

Proof. Follows directly from Proposition 1.7 □

An LP can be a maximization problem as well as a minimization problem. We can convert a maximization LP into a minimization LP or the other way around by multiplying the cost vector by minus one. For convenience, in this chapter we only discuss maximization problems.

Apart from the general form, an LP can also be in *canonical form*

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0, \end{aligned}$$

short $\max \{c^T x \mid Ax \leq b,\, x \geq 0,\, x \in \mathbb{R}^n\}$, or *standard form*

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

short $\max \{c^T x \mid Ax = b,\, x \geq 0,\, x \in \mathbb{R}^n\}$. The three forms are equivalent to each other, in the sense, that an LP of one form can be transformed into an LP of another form such that the two LPs have the same optimum value.

Let $\max \{c^T x \mid Ax \leq b,\, x \in \mathbb{R}^n\}$ be an LP in general form. To transform the LP into canonical form, we decompose x into x^+ and x^- with $x = x^+ - x^-$ and $x^+,\, x^- \geq 0$. An equivalent LP in canonical form is given by

$$\begin{aligned} \max_{x^+,\, x^- \in \mathbb{R}^n} \quad & c^T x^+ - c^T x^- \\ \text{subject to} \quad & Ax^+ - Ax^- \leq b \\ & x^+,\, x^- \geq 0 \end{aligned}$$

Now, let $\max \{c^T x \mid Ax \leq b,\, x \geq 0,\, x \in \mathbb{R}^n\}$ be an LP in canonical form. To transform the LP to standard form, we add slack variables $s \in \mathbb{R}^m$ with $s \geq 0$. An equivalent LP in standard form is given by

$$\begin{aligned} \max_{x \in \mathbb{R}^n,\, s \in \mathbb{R}^m} \quad & c^T x \\ \text{subject to} \quad & Ax + s = b \\ & x, s \geq 0 \end{aligned}$$

Finally, let $\max \{c^T x \mid Ax = b,\, x \geq 0,\, x \in \mathbb{R}^n\}$ be an LP in standard form. An equivalent LP in general form is given by

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & -Ax \leq -b \\ & -x \leq 0 \end{aligned}$$

Example 1.9. *Consider the following LP in general form*

$$\begin{aligned} \max_{x_1, x_2 \in \mathbb{R}} \quad & x_1 + x_2 \\ \text{subject to} \quad & 3x_1 + 2x_2 \leq 10 \\ & 4x_1 + 6x_2 \leq 20 \end{aligned}$$

An equivalent LP in canonical form is given by

$$\begin{aligned} \max_{x_1^+, x_2^+, x_1^-, x_2^- \in \mathbb{R}} \quad & x_1^+ + x_2^+ - x_1^- - x_2^- \\ \text{subject to} \quad & 3x_1^+ + 2x_2^+ - 3x_1^- - 2x_1^- \leq 10 \\ & 4x_1^+ + 6x_2^+ - 4x_1^- - 6x_2^- \leq 20 \\ & x_1^+, x_2^+, x_1^-, x_2^- \geq 0. \end{aligned}$$

An equivalent LP in standard form is given by

$$\begin{aligned}
\max_{x_1^+, x_2^+, x_1^-, x_2^-, s_1, s_2 \in \mathbb{R}} \quad & x_1^+ + x_2^+ - x_1^- - x_2^- \\
\text{subject to} \quad & 3x_1^+ + 2x_2^+ - 3x_1^- - 2x_1^- + s_1 = 10 \\
& 4x_1^+ + 6x_2^+ - 4x_1^- - 6x_2^- + s_2 = 20 \\
& x_1^+, x_2^+, x_1^-, x_2^-, s_1, s_2 \geq 0.
\end{aligned}$$

The final proposition in this section shows that if we are given a feasible LP in standard form, we can always assume that the matrix A has full row rank.

Proposition 1.10. *Let $\{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$ be a non-empty feasible region of an LP in standard form. Then there exist a matrix $A' \in \mathbb{R}^{m' \times n}$ of full row rank and a vector $b' \in \mathbb{R}^{m'}$ such that*

$$\{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\} = \{x \in \mathbb{R}^n \mid A'x = b',\ x \geq 0\}$$

Proof. If the Matrix A has full row rank, then we are done. Otherwise, then there exist a row in A that is a linear combination of the other rows of A. Assume without loss of generality that this row is the last row a_m (exchanging the order of the linear inequalities does not change the feasible set). Thus, we have

$$a_m = \sum_{i=1}^{m-1} \lambda_i a_i$$

where a_i denote the ith row of A. Note that it also applies

$$b_m = \sum_{i=1}^{m-1} \lambda_i b_i,$$

since the feasible region of the given LP is non-empty. Let $A' \in \mathbb{R}^{(m-1)\times n}$ be the matrix consisting of the first $m-1$ rows of A and b' be the column vector consisting of the first $m-1$ entries of b. We show

$$P := \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0,\ x \in \mathbb{R}^n\} = \{x \in \mathbb{R}^n \mid A'x = b',\ x \geq 0,\ x \in \mathbb{R}^n\} =: Q$$

Clearly P is a subset of Q because we are removing a constraint. To show the converse, let x be an arbitrary element of Q. It applies,

$$a_m x = \sum_{i=1}^{m-1} \lambda_i a_i x = \sum_{i=1}^{m-1} \lambda_i b_i = b_m$$

which shows that $x \in P$. Because m is finite, we can iterate this process until we get a matrix of full row rank. $\square$

1.3 Duality

Given an LP in general form $\max\{c^T x \mid Ax \leq b,\ x \in \mathbb{R}^n\}$ which we refer to as the *primal* LP. The *dual* LP of the primal LP is the LP

$$\begin{aligned}
\min_{y \in \mathbb{R}^m} \quad & y^T b \\
\text{subject to} \quad & y^T A = c^T \\
& y \geq 0.
\end{aligned}$$

We call the pair of LPs, a primal LP together with its dual LP, *a primal dual pair.* A feasible solution to the primal LP is called a *primal feasible solution* whereas a feasible solution to the dual LP is called a *dual feasible solution.*

Theorem 1.11 (Weak Duality). *Given an LP* $\max\{c^T x \mid Ax \leq b, x \in \mathbb{R}^n\}$ *and its corresponding dual LP* $\min\{y^T b \mid y^T A = c^T,\ y \geq 0,\ y \in \mathbb{R}^m\}$*. For any dual feasible solution y, it applies $c^T x \leq y^T b$ for any primal feasible solution x.*

Proof. Let y be an arbitrary but fixed dual feasible solution and x an arbitrary primal feasible solution. Since $Ax \leq b$ and $y^TA = c^T$ it applies

$$y^Tb \geq y^TAx = c^Tx. \qquad \square$$

Theorem 1.12 (Strong Duality). *Given a primal LP* $\max\{c^Tx \mid Ax \leq b,\, x \in \mathbb{R}^n\}$ *and its corresponding dual LP* $\min\{y^Tb \mid y^TA = c^T,\, y \geq 0,\, y \in \mathbb{R}^m\}$. *If both the primal LP and its dual LP are feasible then there exists a primal feasible solution* $\bar{x}$ *and a dual feasible solution* $\bar{y}$ *such that* $c^T\bar{x} = \bar{y}^Tb$.

Proof. Let $\hat{y}$ be a dual feasible solution. By Theorem 1.11, it applies

$$\sup\{c^Tx \mid Ax \leq b,\, x \in \mathbb{R}^n\} \leq \hat{y}^Tb < \infty$$

We rewrite the primal problem, it applies

$$\sup\{z \mid z \leq c^Tx, Ax \leq b,\, x \in \mathbb{R}^n,\, z \in \mathbb{R}\} = \sup\{c^Tx \mid Ax \leq b,\, x \in \mathbb{R}^n\}. \tag{1.8}$$

Using Fourier elimination to eliminate $x_1, \dots, x_n$ from the system of linear inequalities

$$Ax \leq b \text{ and } z - c^Tx \leq 0 \tag{1.9}$$

yields a new system

$$\bar{a}_i z \leq \bar{b}_i, \quad \forall i = 1, \dots m'$$

where $\bar{a}_i$ is either 1 or -1 for $i = 1, \dots, m'$. By Theorem 1.1, it applies

$$\sup\{z \mid \bar{a}_i z \leq \bar{b}_i\ \forall i = 1, \dots, m'\} = \sup\{z \mid z \leq c^Tx, Ax \leq b,\, x \in \mathbb{R}^n,\, z \in \mathbb{R}\} \tag{1.10}$$

It follows from Equation (1.8) and (1.10) that

$$\sup\{z \mid \bar{a}_i z \leq \bar{b}_i\ \forall i = 1, \dots, m'\} = \sup\{c^Tx \mid Ax \leq b,\, x \in \mathbb{R}^n\} < \infty$$

which implies $\bar{a}_i = 1$ for at least one $i \in \{1, \dots, m'\}$. Moreover, we have the relation

$$\sup\{z \mid \bar{a}_i z \leq \bar{b}_i\ \forall i = 1, \dots, m'\} = \min\{\bar{b}_i \mid i \in \{1, \dots, m'\}, \bar{a}_i = 1\}.$$

Let j be such that $\bar{b}_j = \min\{\bar{b}_i \mid i \in \{1, \dots, m'\},\, \bar{a}_i = 1\}$. Because we get the inequality $\bar{a}_j z \leq \bar{b}_j$ by using Fourier elimination on the system of inequalities (1.9), the inequality $\bar{a}_j z \leq \bar{b}_j$ can be obtained by taking a non-negative linear combination of inequalities in the system (1.9). This means that there exists a $(\lambda, \bar{y}) \in \mathbb{R} \times \mathbb{R}^m$ with $\lambda \geq 0$ and $\bar{y} \geq 0$ such that

$$\begin{bmatrix} \lambda & \bar{y} \end{bmatrix}^T \begin{bmatrix} 1 & -c^T \\ \mathbf{0} & A \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix} \text{ and } \begin{bmatrix} \lambda & \bar{y} \end{bmatrix}^T \begin{bmatrix} 0 \\ b \end{bmatrix} = \bar{b}_j$$

By multiplying the matrices, we have $\lambda = 1$, $-c^T + \bar{y}^TA = 0$ and $\bar{y}^Tb = \bar{b}_j$ which implies that $\bar{y}$ is a dual feasible solution. Thus, for all $x \in \mathbb{R}^n$ with $Ax \leq b$ it applies

$$c^Tx \leq \bar{y}^Tb$$

Fixing $z^* := \bar{b}_j$, we construct a vector $\bar{x} \in \mathbb{R}^n$ such that $(z^*, \bar{x})$ satisfies the system of linear inequalities (1.9) (by Theorem 1.1 such an $\bar{x}$ exists). It applies

$$c^T\bar{x} \geq z^* = \bar{b}_j = \bar{y}^Tb \text{ and } A\bar{x} \leq b$$

which shows that $\bar{x}$ is a primal feasible solution with $c^T\bar{x} = \bar{y}^Tb$. $\square$

Lemma 1.13. *Let* $\max\{c^Tx \mid Ax \leq b,\, x \in \mathbb{R}^n\}$ *be a feasible LP in general form. If*

$$M := \sup\{c^Tx \mid Ax \leq b,\, x \in \mathbb{R}^n\} < \infty,$$

then the dual LP of $\max\{c^Tx \mid Ax \leq b,\, x \in \mathbb{R}^n\}$ *is feasible*

Proof. We show by contraposition. Suppose the dual LP is infeasible, that is, the set

$$\{y \in \mathbb{R}^m \mid y^TA = c^T,\, y \geq 0,\, y \in \mathbb{R}^m\}$$

is empty. By Theorem 1.4, there exist a $\Delta x \in \mathbb{R}^n$ such that $c^T\Delta x > 0$ and $A\Delta x \leq 0$. Let x be an arbitrary but fixed feasible solution to the primal LP. It applies,

$$A(x + \Delta x) \leq Ax \leq b \text{ and } c^T(x + \Delta x) > c^Tx.$$

Thus, $(x + \Delta x)$ is a feasible solution with a value strictly larger than x. Since x is arbitrary, the primal LP is unbounded. $\square$

We can now prove Proposition 1.7.

Proposition 1.7. *Let* $\max\{c^T x \mid Ax \leq b,\, x \in \mathbb{R}^n\}$ *be an LP in general form with*

$$M := \sup\{c^T x \mid Ax \leq b,\, x \in \mathbb{R}^n\} < \infty$$

then there exists an $x^* \in \mathbb{R}^n$ *with* $Ax^* \leq b$ *such that* $c^T x^* = M$.

Proof. By Lemma 1.13, the dual of the LP is feasible and thus by Theorem 1.12 there exists a pair of primal and dual feasible solution $\bar{x}$ and $\bar{y}$ at which both the primal and dual LP attain their optimum value. □

We also give a characterization of optimum LP solutions. This result is often known as *complementary slackness.*

Corollary 1.14. *Given an LP* $\max\{c^T x \mid Ax \leq b, x \in \mathbb{R}^n\}$ *and its corresponding dual linear program* $\min\{y^T b \mid y^T A = c^T, y \geq 0, y \in \mathbb{R}^m\}$. *Let* x *and* y *be a pair of primal and dual feasible solution. Then the following are equivalent:*

(i) x *and* y *are optimum solutions to the primal and dual LPs*

(ii) $c^T x = y^T b$

(iii) $y^T(b - Ax) = 0$

Proof. The equivalence of (i) and (ii) follows from Theorem 1.12. Equivalence of (ii) and (iii) is shown by the following

$$c^T x = y^T b \Leftrightarrow y^T Ax = y^T b \Leftrightarrow y^T(b - Ax) = 0$$

□

Finally, we can also define dual LP for LP of other forms

Proposition 1.15.

(i) Given an LP in canonical form $\max\{c^T x \mid Ax \leq b, x \geq 0\}$, *the dual LP to the primal LP is* $\min\{y^T b \mid y^T A \geq c^T, y \geq 0\}$

(ii) Given an LP in standard form $\max\{c^T x \mid Ax = b, x \geq 0\}$, *the dual LP to the primal LP is* $\min\{y^T b \mid y^T A \geq c^T\}$

Proof. (i) The linear program $\max\{c^T x \mid Ax \leq b, x \geq 0\}$ is equivalent to the linear program

$$\max\left\{ c^T x \,\middle|\, \begin{bmatrix} A \\ -I \end{bmatrix} x \leq \begin{bmatrix} b \\ 0 \end{bmatrix} \right\}.$$

Taking the dual of this linear program yields

$$\min\left\{ \begin{bmatrix} y & s \end{bmatrix}^T b \,\middle|\, \begin{bmatrix} y & s \end{bmatrix}^T \begin{bmatrix} A \\ -I \end{bmatrix} = c^T, y, s \geq 0 \right\}$$

which is equivalent to $\min\{y^T b \mid y^T A \geq c^T, y \geq 0\}$
(ii) The linear program $\max\{c^T x \mid Ax = b, x \geq 0\}$ is equivalent to the linear program

$$\max\left\{ c^T x \,\middle|\, \begin{bmatrix} A \\ -A \end{bmatrix} x \leq \begin{bmatrix} b \\ -b \end{bmatrix}, x \geq 0 \right\}.$$

Using part (i) the dual of this linear program is

$$\min\left\{ \begin{bmatrix} y^+ & y^- \end{bmatrix}^T \begin{bmatrix} b \\ -b \end{bmatrix} \,\middle|\, \begin{bmatrix} y^+ & y^- \end{bmatrix}^T \begin{bmatrix} A \\ -A \end{bmatrix} = c^T \text{ and } y^+, y^- \geq 0 \right\}$$

which is equivalent to $\min\{y^T b \mid y^T A \geq c^T\}$ □

Since all transformations in Proposition 1.15 are equivalent transformation, all the theorem in this section generalizes to these other primal dual pairs.

1.4 Simplex Algorithm

We will now introduce the *simplex* algorithm to solve LPs in standard form. Because we have shown in Section 1.2 that LPs in general or canonical form can be transformed into an equivalent LP in standard form, the simplex method can be used to solve any LP. We omit the technical details of implementing the simplex algorithm in this section and focus instead on key theoretical results, those who are interested in the implementation details of the simplex method can refer to the classic work of Dantzig [11] or the work of Vanderbei [41] for a more modern treatment.

Consider an LP in standard form $\max\{c^T x \mid Ax = b,\ x \geq 0,\ x \in \mathbb{R}^n\}$ where A is an $m \times n$ matrix with full row rank, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$ vectors. We denote with $A_1, \ldots, A_n$ the columns of A. Let $B = \{B_1, \ldots, B_m\} \subseteq \{1, \ldots, n\}$ be an index set containing m elements. We call B a *basis* of A if the family of vectors $(A_{B_i})_{i \in \{1,\ldots,m\}}$ are linearly independent. Note that since A has rank m, the family of vectors $(A_{B_i})_{i \in \{1,\ldots,m\}}$ forms a basis of the column space of A. We define

$$A_B := \begin{bmatrix} A_{B_1} & \ldots & A_{B_m} \end{bmatrix}$$

to be the *basis matrix* corresponding to the basis B and

$$c_B := \begin{bmatrix} c_{B_1} & \ldots & c_{B_m} \end{bmatrix}^T$$

to be the *cost of the basic variable* of basis B. Note that A_B is an $m \times m$ matrix and is non-singular since its columns are by construction linearly independent. Let N be the index set of indices that are not in B, we denote

$$A_N := \begin{bmatrix} A_{N_1} & \ldots & A_{N_{n-m}} \end{bmatrix}$$

By permutation of the variables, in what follows we assume A have the form

$$A = \begin{bmatrix} A_B & A_N \end{bmatrix},$$

that is, the first m columns are those that are in the basis. We define the *basic solution* corresponding to basis B to be the vector

$$x^* := \begin{bmatrix} A_B^{-1} b \\ \mathbf{0} \end{bmatrix}.$$

By construction, we have

$$Ax^* = b \text{ and } c^T x^* = c_B^T A_B^{-1} b.$$

If x^* is a feasible solution to our LP, we call our basis B *feasible*. Additionally, B is *optimal* if x^* is an optimum solution to our LP. We call the variables x_i^* with $i \in B$ *basic* variables and the other variables x_i^* with $i \in N$ *non-basic* variables. A vector $x \in \mathbb{R}^n$ is a basic solution if there exist a basis B such that x is a basic solution corresponding to basis B. We call a basic solution a *basic feasible solution* (BFS), if additionally $x \geq 0$, that is, the vector is also feasible solution to our LP.

We note that there can be many bases whose BFS are the same as the following example shows:

Example 1.16. *Consider the LP in standard form*

$$\begin{array}{rl} \textit{maximize} & x_1 + x_2 + x_3 + x_4 + x_5 \\ \textit{subject to} & x_1 + \ x_2 + x_3 \qquad\qquad = 7 \\ & x_1 + 2x_2 + \quad + x_4 \qquad = 11 \\ & -2x_1 + x_2 + \qquad + x_5 = -2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{array}$$

The corresponding matrix A to this linear problem is

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ -2 & 1 & 0 & 0 & 1 \end{bmatrix}$$

which is clearly a matrix of full row rank. The bases $B_1 = \{1, 2, 3\}$ and $B_2 = \{1, 2, 4\}$ both have the same basic (feasible) solution correlated to it which is $x_1 = 3$, $x_2 = 4$, $x_3 = 0$, $x_4 = 0$ and $x_5 = 0$.

We call a basic solution that have more than $n - m$ zero entry *degenerate*, that is, at least one basic variables is zero.

Lemma 1.17. *Given an LP in standard form* $\max\{c^T x \mid Ax = b, x \geq 0, x \in \mathbb{R}^n\}$*. If a basic solution x^* has multiple bases associated to it, then x^* is degenerate,*

Proof. Let B_1 and B_2 be two different bases whose corresponding basic solution is x^*. Then there is an $i \in B_1$ with $i \notin B_2$. It applies that $x_j^* = 0$ for $j = 1, \ldots, k$ with $j \notin B_1$ and $x_i^* = 0$ because x_i^* is a non-basic variable with regard to B_2. Thus x_i have at least $n - m + 1$ zero entries. □

The converse direction does not apply. For example, it might be the case that the matrix A is invertible, thus only having one basis, yet the basic solution has zero as an entry.

We present several lemmas that will be useful in constructing the simplex algorithm. The first is a direct implication of Carathtéodory's Theorem. A discussion of Carathtéodory theorem can be found in Appendix C. We show that for any feasible LP in standard form, there is always at least one basic feasible solution.

Proposition 1.18. *Let* $\max\{c^T x \mid Ax = b,\ x \geq 0,\ x \in \mathbb{R}^n\}$ *be an LP in standard form, if the LP is feasible then the LP has at least one BFS.*

Proof. Since the LP is feasible, then $b \in cone(\{A_1, \ldots, A_n\})$. By Caratheodory's theorem, there exists an index set $B = \{B_1, \ldots, B_k\}$ such that $\{A_{B_1}, \ldots, A_{B_k}\}$ are linearly independent and $b \in cone(\{A_{B_1}, \ldots, A_{B_k}\})$. Without lost of generality, assume $|B| = m$ (otherwise, we can augment B using the full-row rank assumption of A). It follows by construction that B is a basis and its corresponding basic solution x_B is a BFS. □

The second lemma gives a characterization for optimal bases.

Lemma 1.19. *Let* $\max\{c^T x \mid Ax = b,\ x \geq 0,\ x \in \mathbb{R}^n\}$ *be an LP in standard form and x^* a BFS with corresponding basis B of A. If*

$$\bar{c}_B := c^T - c_B A_B^{-1} A \leq 0$$

then x^* is optimal. We call $\bar{c}_B$ the ***reduced cost*** *of basis B.*

Proof. Note that the dual LP is given by

$$\min\{y^T b \mid y^T A \geq c^T,\ y \in \mathbb{R}^m\}.$$

We define $y := \left(c_B^T A_B^{-1}\right)^T \in \mathbb{R}^m$. Since $\bar{c}_B \leq 0$ it applies

$$y^T A = c_B^T A_B^{-1} A \geq c^T.$$

which shows that y is feasible. Furthermore, it applies

$$c^T x^* = c_B^T A_B^{-1} b = y^T b$$

Thus x^* and y is a pair of feasible primal and dual feasible solution with the same objective values and are optimal according to Corollary 1.14 □

The third lemma gives a characterization of unbounded LP.

Lemma 1.20. *Let* $\max\{c^T x \mid Ax = b,\ x \geq 0,\ x \in \mathbb{R}^n\}$ *be an LP in standard form and x^* a BFS with corresponding basis B of A. If there exist a $j \in N$ such that*

$$(\bar{c}_B)_j > 0 \text{ and } u = A_B^{-1} A_j \leq 0,$$

then the LP is unbounded.

Proof. Let $N' = N \setminus \{j\}$. Without loss of generality, assume that the matrix A has the form

$$A = \begin{bmatrix} A_B & A_{N'} & A_j \end{bmatrix}.$$

We define,

$$\Delta x := \begin{bmatrix} -u \\ \mathbf{0} \\ 1 \end{bmatrix}.$$

Note that $\Delta x \geq 0$. For all $\lambda \in \mathbb{R}$ with $\lambda \geq 0$, it applies

$$x^* + \lambda \Delta x \geq 0.$$

Furthermore, it applies

$$\begin{aligned} A(x^* + \lambda\Delta x) &= \begin{bmatrix} A_B & A'_N & A_j \end{bmatrix} \begin{bmatrix} A_B^{-1}b - \lambda u \\ \mathbf{0} \\ \lambda \end{bmatrix} \\ &= A_B A_B^{-1} b - \lambda A_B u + \lambda A_j \\ \\ &= A_B A_B^{-1} b - \lambda A_B A_B^{-1} A_j + \lambda A_j \\ &= b - \lambda A_j + \lambda A_j \\ &= b. \end{aligned}$$

Thus $x^* + \lambda \Delta x$ is a feasible solution for all $\lambda \in \mathbb{R}$ with $\lambda \geq 0$. Finally, we have

$$c^T \Delta x = -c_B^T u + c_j A_j = c_j A_j - c_B A_B^{-1} A_j = (\bar{c}_B)_j > 0,$$

which implies that we can move indefinitely in direction Δx while increasing our objective value. Thus, the LP is unbounded. □

Note that for any basis B and $j \in N$,

$$A_B u = A_B A_B^{-1} A_j = A_j$$

that is, the vector u is the coordinate vector of A_j with regard to the basis $\{A_{B_1}, \ldots, A_{B_m}\}$ of the column space of A. From linear algebra, we know that if $u_\ell \neq 0$ for any, $\ell \in \{1, \ldots, m\}$ then the set of vectors

$$A_{B_1}, \ldots, A_{B_{\ell-1}}, A_j, A_{A_{B_{\ell+1}}}, \ldots, A_{B_m}$$

are linearly independent.

Lemma 1.21. *Let* $\max\{c^T x \mid Ax = b,\, x \geq 0,\, x \in \mathbb{R}^n\}$ *be an LP in standard form and* x^* *a BFS corresponding to basis* B *of* A*. For any* $j \in N$ *such that* $\bar{c}_j > 0$ *and* $u = A_B^{-1} A_j$ *have at least one positive entry, we let*

$$\theta^* := \min \left\{ \frac{x^*_{B_i}}{u_i} \,\middle|\, u_i > 0, i = 1, \ldots, m \right\}$$

and $\ell \in \{1, \ldots, m\}$ *to be an index such that* $\theta^* := \frac{x^*_{B_\ell}}{u_\ell}$*. Then*

$$B' := \{B_1, B_{\ell-1}, j, B_{\ell+1}, \ldots, B_m\}$$

is a feasible basis whose corresponding basic solution x' *satisfies,* $c^T x' \geq c^T x^*$ *with strict inequality if* x^* *is non-degenerate.*

Proof. Since $u_\ell > 0$, by the remark above the vectors

$$A_{B_1}, \ldots, A_{B_{\ell-1}}, A_j, A_{A_{B_{\ell+1}}}, \ldots, A_{B_m}$$

are linearly independent and thus B' is a basis. We claim that the basic solution x' corresponding to B' is given by

$$\begin{aligned} x'_{B_i} &= x^*_{B_i} - \theta^* u_i, \quad \forall i \in \{1, \ldots, m\} \\ x'_j &= \theta^* \\ x'_i &= 0 \quad \forall i \in N \setminus \{j\} \end{aligned}$$

Note that $x' \geq 0$, $x'_{B_\ell} = 0$ and $x' = x^*$ if and only if $\theta^* = 0$. Further, we have

$$\begin{aligned} Ax' &= \sum_{i=1}^{m} A_{B_i}(x^*_{B_i} - \theta^* u_i) + \theta^* A_j \\ &= A_B A_B^{-1} b - \theta^* A_B u_i + \theta^* A_j \\ &= b - \theta^* A_B A_B^{-1} A_j + \theta A_j \\ &= b \end{aligned}$$

Finally, we have

$$c^T x' - c^T x^* = \theta^* c_j - \theta^* c_B^T u = \theta^* c_j - \theta^* c_B^T A_B^{-1} A_j = \theta^* \bar{c}_j \geq 0.$$

The strictness of the inequality follows from the observation that if θ^* is zero, x^* is degenerate. □

We present the simplex algorithm in Algorithm 1.

Algorithm 1 Simplex Algorithm (Dantzig [10])

Input: The matrix $A \in \mathbb{R}^{m \times n}$, column vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ defining our linear program $\max \{c^T x \mid Ax = b, x \geq 0\}$, a starting BFS x_0 with corresponding basis B_0

Output: A BFS x^* which is optimal or declare that the linear program is unbounded

```
1. Set incumbent solution $x^* := x_0$ and incumbent basis $B := B_0$
2. Compute the reduced cost of the incumbent basis $\bar{c}_B = c - c_B A_B^{-1} A$. If $\bar{c}_B \leq 0$
   the incumbent BFS is optimal (by Lemma 1.19). Otherwise, go to step 3.
3. Select the smallest $j \in N$ with $\bar{c}_j > 0$ and compute $u = A_B^{-1} A_j$. If $u \leq 0$, LP is
   unbounded (by Lemma 1.20). Otherwise, go to step 4.
4. Let $\theta^* = \min \left\{ \frac{x^*_{B_i}}{u_i} \middle| u_i > 0, i = 1, \dots, m \right\}$ and $\ell \in \{1, \dots, n\}$ be an index such
   that $\theta^* = \frac{x^*_{B_\ell}}{u_\ell}$. Generate new incumbent basis by replacing $B_\ell$ from $B$ with
   $j$ according to Lemma 1.21 and update the incumbent solution $x^*$ to the BFS
   corresponding to the new basis.
```

The choice of indices j in step 3 and ℓ in step 4 may not be unique. The *bland pivoting rule* proposed by Bland in [6] states that when multiple options are available, one should always choose the smallest index. Bland also proved the following theorem:

Theorem 1.22. *Let* $\max \{c^T x \mid Ax = b,\ x \geq 0,\ x \in \mathbb{R}^n\}$ *be an LP in standard form and* x_0 *a BFS with basis* B *of* A*. The simplex algorithm with* x_0 *as a starting BFS and bland pivoting rule terminates after at most* $\binom{n}{m}$ *iterations.*

Corollary 1.23. *Let* $\max \{c^T x \mid Ax = b,\ x \geq 0,\ x \in \mathbb{R}^n\}$ *be a feasible LP. If the LP is bounded, then there exist a BFS that is an optimum solution.*

We give a method to find an initial BFS for the simplex algorithm.

Proposition 1.24. *Given a linear program in standard form* $\max \{c^T x \mid Ax = b, x \geq 0\}$ *with* $b \geq 0$*. We add* m *artificial variables and define the* ***phase one*** *linear program to be*

$$\begin{aligned} \max_{x,a} \quad & -\mathbf{1}^T a \\ \textit{subject to} \quad & Ax + I_m a = b \\ & x, a \geq 0 \end{aligned}$$

where a *is a vector of the artificial decision variables. It applies:*

(i) The phase one linear program is feasible

(ii) The phase one linear program has an optimum solution

(iii) The phase one linear program has an optimum value of 0 *if and only if the original linear program is feasible*

Proof.

(i) The phase one linear program is feasible because $x = 0$ and $a = b$ is a feasible solution.

(ii) The phase one linear is bounded because its optimum value is at most 0.

(iii) We first show sufficiency. If (x^*, a^*) is an optimum solution of cost 0, then $a^* = 0$. It follows that $Ax^* = b$. Thus x^* is a feasible solution to our original linear program. To show necessity, if x^* is a feasible solution to our original linear program then $x = x^*$ and $a = 0$ is a feasible solution to the phase one linear program with a value of 0.

□

We note that $x = 0$ and $a = b$ is also a basic feasible solution for the phase one linear program where as basis we select the columns of I_m. We can use simplex algorithm to solve the phase one problem and get an optimal basic feasible solution (x^*, a^*) with $a^* = 0$ and the associated optimal basis B^*. Note that x^* is a basic feasible solution to our original linear program. To get the corresponding basis, we remove from B^* all indices corresponding to columns of I_m and augment it into a basis of the column space of A.

1.5 Integer Linear Programming

Given a matrix $A \in \mathbb{Q}^{m\times n}$, vectors $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$ vectors and an index set $I \subseteq \{1, \dots, n\}$, the *mixed integer linear program* (MILP) in standard form is to solve

$$\begin{aligned} \max_{x\in\mathbb{R}^n} \quad & c^T x \\ \text{subject to} \quad & Ax \le b \\ & x_i \in \mathbb{Z}, \quad \forall i \in I \end{aligned}$$

Similar to LPs, the MILP *infeasible* if the set

$$\{x \in \mathbb{R}^n | Ax \le b,\ x_i \in \mathbb{Z}\ \forall i \in I\}$$

is empty and *unbounded* if for every $x \in \mathbb{R}^n$ with $Ax \le b$ there exists an $x' \in \mathbb{R}^n$ such that $c^T x < c^T x'$, $Ax' \le b$, and $x'_i \in \mathbb{Z}$ for all $i \in I$. We call constraints of the form $x_j \in \mathbb{Z}$ *integrality constraints*. Similar to LPs, MILP can also be in general form or canonical form. A MILP is called an *Integer Linear Program* (ILP) if all variables are constrained to be integer. For the rest of this section, we discuss exclusively ILPs. Furthermore, we assume that upper- and lower-bound vectors $u \in \mathbb{Q}^n$ and $\ell \in \mathbb{Q}^n$ are given such that $\ell \le x \le u$.

We denote with $ILP(A, b, c, u, \ell)$ to be the optimization problem

$$\begin{aligned} \max_{x\in\mathbb{Z}^n} \quad & c^T x \\ \text{subject to} \quad & Ax \le b \\ & \ell \le x \le u \end{aligned}$$

.

We define the *LP relaxation* $\overline{ILP}(A, b, c, u, \ell)$ of $ILP(A, b, c, u, \ell)$ to be the LP that we get by dropping the integrality constraints and optimizing over $\mathbb{R}^n$ instead. We observe that the optimum value of $\overline{ILP}(A, b, c, u, \ell)$ is always greater or equal to $ILP(A, b, c, u, \ell)$ because feasible solutions of $ILP(A, b, c, u, \ell)$ are also feasible solutions to $\overline{ILP}(A, b, c, u, \ell)$. The optimum value of $\overline{ILP}(A, b, c, u, \ell)$ is called the *dual bound* of $ILP(A, b, c, u, \ell)$. Note that since we have upper- and lower-bounds for all our variable, both the ILP and its LP relaxation are bounded (that is, the dual bound always exist). Furthermore, the ILP also attains a maximum value because there are only finitely many integer points in the feasible set.

Example 1.25 (Hitting Set Problem). *Let $\mathcal{S}$ be a set with finite number of elements and $S_1, \dots, S_k$ be finitely many subsets of $\mathcal{S}$. A set $H \subseteq \mathcal{S}$ hits a subset S_i with $i \in \{1, \dots, k\}$ if $H \cap S_i \ne \emptyset$. The* hitting set problem *is to find a set $\bar{H}$ of minimum cardinality that hits all set S_i with $i \in \{1, \dots, k\}$.*

Let $m = |\mathcal{S}|$. We label each item in $\mathcal{S}$ with a number $1, \dots, m$. *The following is an IP formulation of the hitting set problem*

$$\begin{aligned} \min_{x\in\mathbb{Z}^m} \quad & \sum_{i=1}^{m} x_i \\ \text{subject to} \quad & \sum_{i\in S_j} x_i \ge 1, \forall j \in \{1, \dots, k\} \end{aligned}$$

The variables x_i determines whether or not item i is included in the hitting set $\bar{H}$. The constraint of the integer program requires that each set S_j with $j \in \{1, \dots, k\}$ is hit by $\bar{H}$. The desicion problem associated to the hitting set problem is shown by Karp to be NP-Complete [26].

There exist numerous solvers to solve MILPs such as GUROBI [22], CPLEX [25], and SCIP [5]. The first two are commercial software, while the last one is open source. An exhaustive discussion of how these softwares work is beyond the scope of this thesis. We will consider these softwares as black boxes for the rest of this thesis. However, we will in this section give a sketch of how ILP can be solved.

Lemma 1.26. *Let $ILP(A, b, c, u, \ell)$ be an integer linear program. If the optimum solution to $\overline{ILP}(A, b, c, u, \ell)$ is integral, then it is also an optimum solution to $ILP(A, b, c, u, \ell)$.*

Proof. This follows from the fact that any feasible solution x of $ILP(A, b, c, u, \ell)$ is also a feasible solution of $\overline{ILP}(A, b, c, u, \ell)$. □

Suppose that x^* is the optimum solution to $\overline{ILP}(A, b, c, u, \ell)$ and there exists an index $j \in \{1, \dots, n\}$ such that x^*_j is non-integral. We describe two strategy to handle this case. The first strategy is to *branch*.

Lemma 1.27. *Let $ILP(A, b, c, u, \ell)$ be an ILP, $j \in \{1, \dots, n\}$, $q \in \mathbb{R}$ with $\ell_j \leq q \leq u_j$. One can solve $ILP(A, b, c, u, \ell)$ by solving the two subproblems, $ILP(A, b, c, u', \ell)$ and $ILP(A, b, c, u, \ell')$ where*

$$u'_i = \begin{cases} u_i, \forall i \in \{1, \dots, n\}, i \neq j \\ \lfloor q \rfloor, i = j \end{cases} \quad \text{and } \ell'_i = \begin{cases} \ell_i, \forall i \in \{1, \dots, n\}, i \neq j \\ \lceil q \rceil, i = j \end{cases}$$

and taking the maximum of the two solutions. We call this operation branching *on variable x_j.*

Proof. The sets $\{x \in \mathbb{Z}^n \mid Ax \leq b, \ell \leq x \leq u'\}$ and $\{x \in \mathbb{Z}^n \mid Ax \leq b, \ell' \leq x \leq u\}$ form a partitioning of the set $\{x \in \mathbb{Z}^n \mid Ax \leq b, \ell \leq x \leq u\}$. Thus, optimizing over each partition individually and taking the maximum is the same as optimizing over the whole set directly. □

Note that in particular, we can branch on variable x_j with $q = x^*_j$ and x^* is not a feasible solution of the LP relaxation of both subproblems. This strategy motivates Algorithm 2.

Algorithm 2 is an example of a branch and bound algorithm. Branch and bound algorithms enumerate the whole feasible set of an ILP using 'divide and conquer'. It successively partitions the feasible set of an ILP into smaller subsets and solves over each set individually.

When considering a given subset, a *bounding strategy* is used to determine an upper bound of the objective function over the subset. For example, the bounding strategy in Algorithm 2 is to use LP relaxation. If the upper bound of the objective function over the subset is lower than that of an already known feasible solution of our original ILP, the subset does not need to be considered further and can be discarded. We call this *fathom by bounding*. The upper bounds given by the bounding strategy are called *dual bounds*.

If we cannot fathom a subset by bounding, then we have two options, either to solve it to optimality over the subset or divide it into smaller sets. The former can be very hard. Algorithm 2 does this opportunistically when the optimal solution to our LP relaxation is integral, in which case by Lemma 1.26 we have an optimal solution over the currently evaluated subset. If an optimal solution over the subset can be found, the solution can be kept and the rest of the set can be discarded. We call this *fathom by optimality*. The solution can then be checked against the previously best known feasible solution and, if better, can be kept. Such a locally optimal solution is a lower bound to the optimal solution of the ILP. We call such bounds *primal bounds*.

The other option is to decide against solving to optimality and partition a subset into two smaller subsets instead by branching. Note that in Algorithm 2 we cannot branch indefinitely, as eventually our feasible set will only consist of a single integer point which we can fathom by optimality.

Lastly, we note that we also have the flexibility to determine which set to investigate first. To summarize, there are 4 defining aspects of a branch and bound strategy:

1. A branching strategy to partition a subset into smaller subsets

Algorithm 2 LP-Based Branch And Bound

```
Input: An integer linear program $ILP(A, b, c, u, \ell)$
Output: An optimum solution $x^*$ for $ILP(A, b, c, u, \ell)$

1. Initialize: We initiate an empty list $N$ of subproblems, set the incumbent
   lower bound $\underline{Z} = -\infty$, and the incumbent solution $x^*$ to void. We add
   $ILP(A, b, c, u, \ell)$ to the list $N$.
2. Select Node: If $N$ is empty, go to step 6. Otherwise, pick a subproblem
   $ILP(A, b, c, u^i, \ell^i)$ from $N$.
3. Fathom By Bounding: Compute the dual bound $\bar{Z}^i := \overline{ILP}(A, b, c, u^i, \ell^i)$ and
   let $\bar{x}^i$ be the optimum solution. If $\bar{Z}^i \leq \underline{Z}$ or $\overline{ILP}(A, b, c, u^i, \ell^i)$ is infeasible,
   return to step 2. Otherwise, go to step 4.
4. Fathom By Optimality: If $\bar{x}^i$ is an integral vector, then update the incumbent
   lower bound $\underline{Z} = \bar{Z}^i$, incumbent solution $x^* = \bar{x}^i$ and return to step 2. Other-
   wise, continue to step 5.
5. Branch: Select an index $j \in \{1, \ldots, n\}$ where $\bar{x}_j$ is not an integer. Branch on
   variable $j$ by Lemma 1.27 with $q = \bar{x}^i_j$. Add the two generated subproblems to
   $N$. Return to step 2.
6. Terminate: If $x^*$ is void, $ILP(A, b, c, u, \ell)$ is infeasible. Otherwise, $x^*$ is the
   optimal solution to $ILP(A, b, c, u, \ell)$
```

2. A bounding strategy to find an upper bound of the objective function over a given set
3. A method to find an optimal solution for some subsets
4. A node selection strategy to determine which set to be investigated first.

A second strategy to handle fractional variables x^*_j is to *cut*. For a set $S \subseteq \mathbb{R}^n$, we call an inequality $\alpha^T x \leq \beta$ with $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$ *valid* for S if it is satisfied by all vectors S. We call a valid inequality $\alpha^T x \leq \beta$ for S a *cutting plane* that separates $x_0 \in \mathbb{R}^n$ from S if additionally $\alpha^T x_0 > \beta$, that is, the inequality is violated by x_0.

Example 1.28. *Consider the ILP*

$$\begin{aligned} \max_{x,y \in \mathbb{Z}} \quad & 3x + 4y \\ \text{subject to} \quad & 5x + y \leq 15 \\ & x + 5y \leq 15 \\ & x \geq 0.5 \\ & y \geq 0.5 \end{aligned}$$

The feasible set of the LP relaxation is marked blue in Figure 1.4. The optimal solution of the LP relaxation is the point $(2.5, 2.5)$. *The ILP has 4 feasible solutions marked in blue dots. The inequality*

$$13x + 5y \leq 39$$

is a valid cutting plane that separates the point $(2.5, 2.5)$ *from the feasible set of the ILP. The area that is removed from the feasible set of the LP by adding this cutting plane as a constraint is marked in dark blue shade.*

This motivates Algorithm 3. The problem of finding good cutting planes is a research topic in itself. Whether Algorithm 3 converge depends on the method for finding a cutting plane. To close this section, we highlight two possibilities for mixing branching and cut. The first possibility is to *cut and branch*. Using cutting planes to reduce the size of the feasible ILP set before starting branch and bound. This requires that the cut added should be *globally* valid, that is, it needs to be valid with regard to the entire feasible set of our ILP. Another alternative would be to mix the two, resulting in a *branch and cut* scheme as shown in Algorithm 4. The benefit of *branch and cut* is we can also employ *locally* valid cuts, cuts

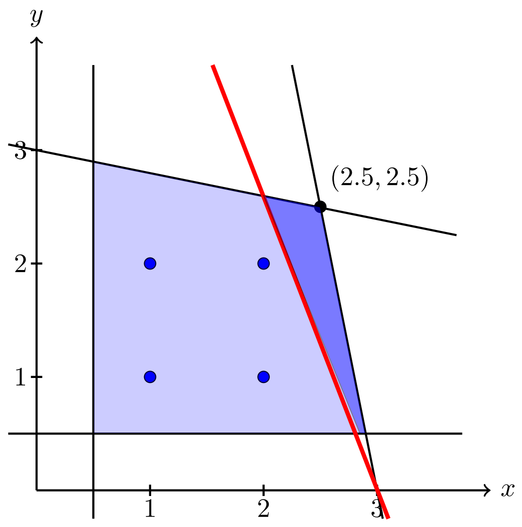


Figure 1.4: Illustration For Example 1.28

Algorithm 3 Cutting Plane Algorithm

Input: An ILP $ILP(A, b, c, u, \ell)$
Output: An optimum solution x^* to $ILP(A, b, c, u, \ell)$ (if any)

```
1. Initialize: Let $S := \{x \in \mathbb{R}^n \mid Ax \leq b, \ell \leq x \leq u\}$.
2. Solve LP: Solve the LP $\max\{c^T x \mid x \in S\}$. If the LP is infeasible, then the ILP
   is infeasible. Otherwise, let $x^*$ be the optimum solution. If $x^*$ is integral, then
   $x^*$ is an optimum ILP solution. Otherwise, go to step 3.
3. Cut: Find a cutting plane $\alpha^T x \leq \beta$ that separates $x^*$ from
       $\{x \in \mathbb{Z}^n \mid Ax \leq b, \ell \leq x \leq u\}$
   and redefine $S$ to be
       $S := \{x \in \mathbb{R}^n \mid A_x \leq b, \ell \leq x \leq u, \alpha x \leq \beta\}$
   Go back to step 2.
```

that are valid only for the feasible set of a subproblem $ILP(A^i, b^i, c, u^i, \ell^i)$ rather than the original ILP.

1.6 Intersection Cut

In this section, we introduce a method to generate valid cutting planes for ILP known as *intersection cuts*. We consider an ILP in standard form

$$\begin{aligned} \max_{x \in \mathbb{Z}^n} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0. \end{aligned} \tag{1.11}$$

Let B be a feasible basis of A and x^* the associated BFS. We denote as before N the set of indices that are not in the basis. We can rewrite the system of linear equations $Ax = b$ as

$$A_B x_B + A_N x_N = b$$

Algorithm 4 LP-Based Branch And Cut

Input: An integer linear program $ILP(A, b, c, u, \ell)$
Output: An optimum solution x^* for $ILP(A, b, c, u, \ell)$

1. *Initialize*: Let $ILP(A, b, c, u, \ell)$ be the integer linear program given as an input. We initiate an empty list N of subproblems, set the incumbent lower bound $\underline{Z} = -\infty$, and the incumbent solution x^* to void. We finally add $ILP(A, b, c, u, \ell)$ to the list N.

2. *Select Node*: If N is empty, go to step 7. Otherwise, remove a subproblem $ILP(A^i, b^i, c, u^i, \ell^i)$ from N.

3. *Fathom By Bounding*: Compute the dual bound $\bar{Z}^i := \overline{ILP}(A^i, b^i, c, u^i, \ell^i)$ and let $\bar{x}^i$ be the optimum solution. If $\bar{Z}^i \leq \underline{Z}$ or $\overline{ILP}(A^i, b^i, c, u^i, \ell^i)$ is infeasible, return to step 2. Otherwise, go to step 4.

4. *Fathom By Optimality*: If $\bar{x}^i$ is an integral vector, then update the incumbent lower bound $\underline{Z} = \bar{Z}^i$ and incumbent solution $x^* = \bar{x}^i$ and return to step 2. Otherwise, continue to step 5.

5. *Cut:* Decide whether to cut or branch. If we decide to branch, go to step 6. If we decide to cut, find a cutting plane $\alpha^T x \leq \beta$ that separates $\bar{x}^i$ from $\{x \in \mathbb{Z}^n \mid A^i_x \leq b^i, \ell^i \leq x \leq u^i\}$. Generate the augmented ILP by adding the constraint $\alpha^T x \leq \beta$ to $ILP(A^i, b^i, c, u^i, \ell^i)$ and add the augmented ILP to N. Return to step 2.

6. *Branch*: Select an index $j \in \{1, \ldots, n\}$ where $\bar{x}^i_j$ is not an integer. Branch on variable j by Lemma 1.27 with $q = \bar{x}_j$. Add the two generated subproblems to N. Return to step 2.

7. *Terminate*: If x^* is void, $ILP(A, b, c, u, \ell)$ is infeasible. Otherwise, x^* is the optimal solution to $ILP(A, b, c, u, \ell)$

where x_B is the vector of basic variables and x_N of non-basic variables. Equivalently, we have

$$A_B x_B = b - A_N x_N$$

Using the fact that A_B is non-singular, this is equivalent to

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N = \bar{x} - A_B^{-1} A_N x_N$$

with $\bar{x} := A_B^{-1} b$. Thus the LP relaxation of ILP 1.11 can be rewritten as

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{subject to} \quad & x_B = \bar{x} - A_B^{-1} A_N x_N \\ & x_B, x_N \geq 0 \end{aligned} \tag{1.12}$$

We define the set

$$P(B) := \{(x_B, x_N) \in \mathbb{R}^n \mid x_B = \bar{x} - A_B^{-1} A_N x_N, x_B \in \mathbb{R}^m, x_N \in \mathbb{R}^{n-m}, x_N \geq 0\}$$

that is, the set of feasible solutions by relaxing the constraint $x_B \geq 0$ from LP 1.12. Each point $x \in P(B)$ can be written as

$$x = x^* + \bar{A} x_N$$

with some $x_N \in \mathbb{R}^{n-m}, x_N \geq 0$ and

$$\bar{A} = \begin{bmatrix} -A_B^{-1} A_N \\ I_{n-m} \end{bmatrix}.$$

Note that by construction, the columns of $\bar{A}$ are linearly independent by construction. For each $i \in N$, we denote $\bar{A}_i$ to be the unique column of $\bar{A}$ whose first m entries are defined by $-A_B^{-1} A_i$. We introduce *intersection cuts* first proposed by Balas [3]. The proof of this theorem can be found in the original paper.

Theorem 1.29 (Balas [3])**.** *Let $C \subseteq \mathbb{R}^n$ be a closed and convex set. We define for all $i \in N$*

$$\alpha_i := \max\{\alpha \geq 0 : x^* + \alpha \bar{A}_i \in C\}$$

and $\frac{1}{+\infty} := 0$. It applies that the set

$$S := \left\{ x \in P(B) \;\middle|\; \sum_{i \in N} \frac{x_i}{\alpha_i} < 1 \right\}$$

is contained in the interior of C.

Corollary 1.30. *Let $C \subseteq \mathbb{R}^n$ be a closed and convex set such that x^* is an interior point of C and C contains no integer point in its interior. We define α_i for $i \in N$ as in Theorem 1.29 and $\frac{1}{+\infty} = 0$. The inequality*

$$\sum_{i \in N} \frac{x_i}{\alpha_i} \geq 1$$

is a valid cutting plane that separates x^* from the feasible set of ILP (1.11).

We call cutting planes generated according to Corollary 1.30 *intersection cuts*. An interesting property of intersection cut, the closed and convex set C can have any arbitrary property and the generated intersection cut will cut from our feasible set points elements which fulfill that property. For example, in Corollary 1.30, the property is non-integral points. In the next chapter, we will consider closed and convex sets which contains no bilevel feasible solution.

Chapter 2

Bilevel Programming

In this chapter, we introduce *bilevel optimization*, optimization problems involving two decision makers, commonly called the *leader* and the *follower*, who make their decisions in a hierarchical manner: the leader first makes their decision that maximizes their objective, and then the follower makes their decision that maximizes their objective subjected to the decision of the leader. The leader is assumed to be able to predict the follower's response. Thus, the leader's task is a nested optimization problem that takes into account the follower's response. The first three sections introduce mixed integer bilevel linear program (MIBLP) and cover some examples of MIBLP commonly used in the literature to demonstrate properties of MIBLP that make it difficult to solve. We also show a reformulation of bilevel linear programs into single-level mixed integer programs. These three sections follow closely from the work of Kleinert et al.[27]. In the last two sections, we present a branch and cut scheme for solving integer bilevel linear programs proposed by Fischetti et al. [20].

2.1 Definitions

Definition 2.1 (Mixed Integer Bilevel Linear Program). *Given rational matrices $A \in \mathbb{Q}^{\ell \times n_x}$, $B \in \mathbb{Q}^{\ell \times n_y}$, $G_x \in \mathbb{Q}^{m \times n_x}$, and $G_y \in \mathbb{Q}^{m \times n_y}$, vectors $q \in \mathbb{R}^m$, $b \in \mathbb{R}^\ell$, $c_x \in \mathbb{R}^{n_x}$, $c_y \in \mathbb{R}^{n_y}$, and $d \in \mathbb{R}^{n_y}$, and index sets $I_x \subseteq \{1, \ldots, n_x\}$, $I_y \subseteq \{1, \ldots, n_y\}$, and $I_y^L \subseteq \{1, \ldots, n_y\}$, the mixed integer bilevel linear program (MIBLP) is to solve the problem*

$$\begin{aligned}
\min_{x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}} \quad & c_x^T x + c_y^T y && \text{(2.1a)}\\
\textit{subject to} \quad & G_x x + G_y y \geq q && \text{(2.1b)}\\
& x_j \in \mathbb{Z}, \quad \textit{for all } j \in I_x && \text{(2.1c)}\\
& y_j \in \mathbb{Z}, \quad \textit{for all } j \in I_y && \text{(2.1d)}\\
& y \in S(x) && \text{(2.1e)}
\end{aligned}$$

where $S(x)$ is the set of optimal solutions of the x-parameterized MILP

$$\begin{aligned}
\min_{y \in \mathbb{R}^{n_y}} \quad & d^T y && \text{(2.2a)}\\
\textit{subject to} \quad & Ax + By \geq b && \text{(2.2b)}\\
& y_j \in \mathbb{Z}, \quad \textit{for all } j \in I_y^L && \text{(2.2c)}
\end{aligned}$$

We call the problem a bilevel linear program (BLP) if there are no integrality constraints and an integer linear bilevel linear program (IBLP) if all variables are constrained to be integer.

Problem 2.1 is called the *upper-level* (or the *leader's* problem) and Problem 2.2 is called the *lower-level* (or the *follower's*) problem. We call the variables $x \in \mathbb{R}^{n_x}$ the *upper-level* (or *leader's decision*) variables and $y \in \mathbb{R}^{n_y}$ the lower-level (or follower's decision) variables. Upper-level variables which appears on lower-level constraints are called *linking variables.* Moreover, we call an upper-level constraint a *coupling constraint* if lower-level variables have nonzero coefficients, that is, the constraint involves lower-level variables.

For a given $x \in \mathbb{R}^{n_x}$, we define the *follower value function*

$$\Phi(x) = \min_{y \in \mathbb{R}^{n_y}} \{d^T y \mid Ax + By \geq b, y_j \in \mathbb{Z} \text{ for all } j \in I_y^L\}$$

Using the follower value function we can rewrite any MIBLP as a single level optimization problem as follows

$$
\begin{aligned}
\min_{x\in\mathbb{R}^{n_x}, y\in\mathbb{R}^{n_y}} \quad & c_x^T x + c_y^T y && \text{(2.3a)}\\
\text{subject to} \quad & G_x x + G_y y \geq q && \text{(2.3b)}\\
& Ax + By \geq b && \text{(2.3c)}\\
& x_j \in \mathbb{Z}, \quad \text{for all } j \in I_x && \text{(2.3d)}\\
& y_j \in \mathbb{Z}, \quad \text{for all } j \in I_y && \text{(2.3e)}\\
& y_j \in \mathbb{Z}, \quad \text{for all } j \in I_y^L && \text{(2.3f)}\\
& d^T y = \Phi(x) && \text{(2.3g)}
\end{aligned}
$$

We call this reformulation the *value-function* reformulation. A vector $(x, y) \in \mathbb{R}^{n_x+n_y}$ is *bilevel feasible* if it is a feasible solution of the value-function reformulation (and thus a feasible solution to our original bilevel optimization problem). The set of all bilevel feasible solution is called the *bilevel feasible set*. We call the MILP that we obtain by dropping constraint (2.3g) from Problem 2.3 the *high point relaxation* (HPR) of the MIBLP problem and the LP relaxation of the HPR to be the *LP high point relaxation* ($\overline{HPR}$). Note that both HPR and $\overline{HPR}$ give valid bounds to our MIBLP.

2.2 Some Examples Of MIBLP

We show some properties of MIBLP, which make them difficult to solve by virtue of examples that exist in the literature. The first example shows that bilevel feasible sets are non-convex.

Example 2.2 (Kleinert et al. [27]). *Consider the following BLP*

$$
\begin{aligned}
\min_{(x,y)\in\mathbb{R}^2} \quad & y\\
\textit{subject to} \quad & y \in S(x)
\end{aligned}
$$

where $S(x)$ is the set of optimal solution of the x-parameterized linear program

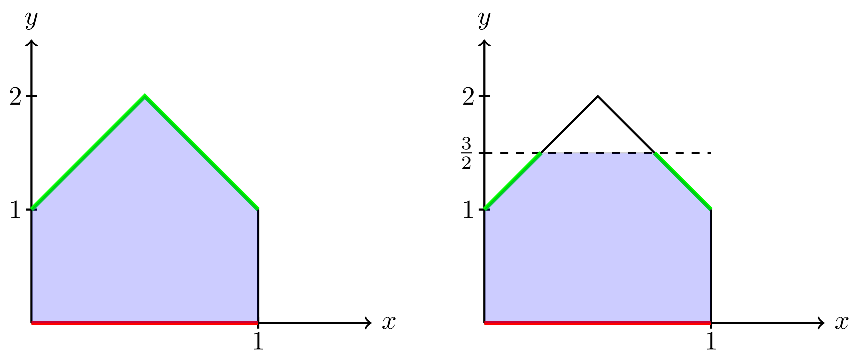
$$
\begin{aligned}
\min_{y\in\mathbb{R}} \quad & -y\\
\textit{subject to} \quad & y \leq 1 + x\\
& y \leq 3 - x\\
& 0 \leq x \leq 1\\
& y \geq 0
\end{aligned}
$$

The feasible set of the lower-level problem is colored blue in Figure 2.1a. Since there is no upper-level constraint apart from the lower-level optimality constraint, the feasible set of high point relaxation coincides with the feasible set of the lower-level problem. The optimum solutions to our high point relaxation is the line segment marked in red, with all solutions on the line segment having an optimum value of 0. The bilevel feasible set is the union of the two line segments marked in green. Visually, we see that the optimal solution to the BLP are the vectors $(1, 1)$ and $(0, 1)$ with an optimum value of 1. Note that this value is greater than the optimum value of our high point relaxation. In this example, we also observe that the bilevel feasible set is non-convex.

Mersha and Dempe discussed that in the case where coupling constraints exist, the bilevel feasible set can be disconnected [33].

Example 2.3 (Continuation of Example 2.2 from Kleinert et al. [27]). *We now add the coupling constraint $y \leq 3/2$ to the BLP from Example 2.2. The upper-level optimization problem is now*

$$
\begin{aligned}
\min_{(x,y)\in\mathbb{R}^2} \quad & y\\
\textit{subject to} \quad & y \leq \frac{3}{2}\\
& y \in S(x)
\end{aligned}
$$



(a) Illustration of Example 2.2 (b) Illustration of Example 2.3

Figure 2.1: An Example of a BLP from Kleinert et al. [27]

while the lower-level LP stays the same. The new bilevel feasible set is shown in Figure 2.1b. We observe that the set is disconnected.

Xu and Wang in [45] also points out that coupling constraints make it more difficult to find bilevel feasible solutions. If there are no coupling constraints, then we can acquire bilevel feasible solutions by fixing an x^* and solving the follower program to find an optimum solution y^*. If the optimum solution y^* exist, then (x^*, y^*) will always be a bilevel feasible solution. However, if there are coupling constraints, then (x^*, y^*) might not be bilevel feasible because it violates a coupling constraint. For these reasons, many algorithms that are proposed in the literature for solving MIPBLP such as DeNegre and Ralphs [14] and Bard and Moore [34] only work without coupling constraints.
The next example shows that even if the upper-level objective is bounded over the bilevel feasible set, it does not necessarily attain a minimum.

Example 2.4 (Köppe, Queyranne, and Ryan [30]). *Consider the following MIPLB*

$$\begin{aligned} \min_{(x,y)\in\mathbb{R}^2} \quad & x - y \\ \textit{subject to} \quad & 0 \leq x \leq 1 \\ & y \in S(x) \end{aligned}$$

where $S(x)$ is the set of optimal solutions to the x-parameterized linear program

$$\begin{aligned} \min_{y\in\mathbb{R}} \quad & y \\ \textit{subject to} \quad & y \geq x \\ & 0 \leq y \leq 1 \\ & y \in \mathbb{Z} \end{aligned}$$

The bilevel feasible set is shown in Figure 2.2 marked in green. We observe that the infimum of the upper-level objective over the bilevel feasible set is -1. However, this value cannot be attained by any bilevel feasible solution.

We call an MIBLP *bounded* if the upper level objective is bounded from below over the bilevel feasible set. Vicente, Savard and Judice showed in [42] the following theorem.

Theorem 2.5. *A feasible and bounded MIBLP has an optimum solution if any one of the following holds:*

(i) All upper- and lower-level variables are continuous

(ii) All upper- and lower-level variables are discrete

(iii) Only upper-level variables are discrete

In what follows we only discuss MIBLP for which the upper- and lower-level variables are either all continuous or all integer, that is, we only consider BLP and IBLP.

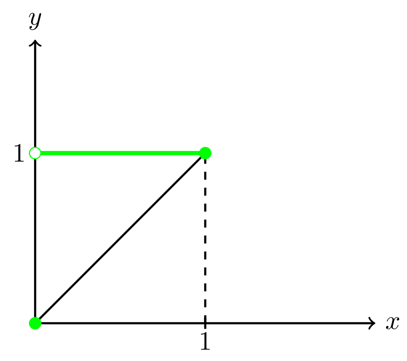


Figure 2.2: An attainability counterexample from Köppe, Queyranne, and Ryan [30]

Lastly, we have the following notorious example from Bard and Moore [34], which shows that relaxing the integrality constraints of an MIBLP does not give a bound to the optimal value of the MIBLP.

Example 2.6 (Bard and Moore [34])**.** *Consider the following bilevel ILP*

$$\begin{aligned} \min_{(x,y)\in\mathbb{R}^2} \quad & -x-10y \\ \textit{subject to} \quad & x \in \mathbb{Z} \\ & y \in S(x) \end{aligned}$$

where $S(x)$ is the set of optimal solutions of the x-parameterized linear program

$$\begin{aligned} \min_{y\in\mathbb{R}} \quad & y \\ \textit{subject to} \quad & -25x+20y \le 30 \\ & x+2y \le 10 \\ & 2x-y \le 15 \\ & 2x+10y \ge 15 \\ & y \in \mathbb{Z} \end{aligned}$$

The feasible set of the LP high point relaxation is the marked blue in Figure 2.3 and the bilevel feasible set is marked green. The optimum value of the high point relaxation is -44 with the optimum solution being $(2,4)$. If we relax all integrality constraints, the bilevel feasible set becomes the union of the line segments marked in red and the optimum solution of the bilevel is $(8,1)$ with an optimum value of -18. However, the optimum value of the original bilevel is -22 which is attained at $(2,2)$. Thus, relaxing the integrality constraint does not give a bound to our MIBLP.

To conclude, we list the four challanges of solving MIBLIPs:

(i) The bilevel feasible set can be non-convex even in the continuous case.

(ii) If coupling constraints are present, the bilevel feasible set can be disconnected.

(iii) The infimum of the upper-level objective function over the bilevel feasible set may not be attainable even if it is finite.

(iv) Relaxing integrality constraints does not give a lower bound to the optimum value.

2.3 Single Level Reformulation For BLPs

We show how BLPs can be reformulated as a single level MILP. Let $A \in \mathbb{R}^{\ell\times n_x}$, $B \in \mathbb{R}^{\ell\times n_y}$, $G_x \in \mathbb{R}^{m\times n_x}$, and $G_y \in \mathbb{R}^{m\times n_y}$ be rational matrices and $q \in \mathbb{R}^m$, $b \in \mathbb{R}^\ell$, $c_x \in \mathbb{R}^{n_x}$, $c_y \in \mathbb{R}^{n_y}$, and $d \in \mathbb{R}^{n_y}$ be vectors. We consider the BLP

$$\min_{x\in\mathbb{R}^{n_x},y\in\mathbb{R}^{n_y}} \quad c_x^T x + c_y^T y \tag{2.4a}$$

$$\text{subject to} \quad G_x x + G_y y \ge q \tag{2.4b}$$

$$y \in S(x) \tag{2.4c}$$

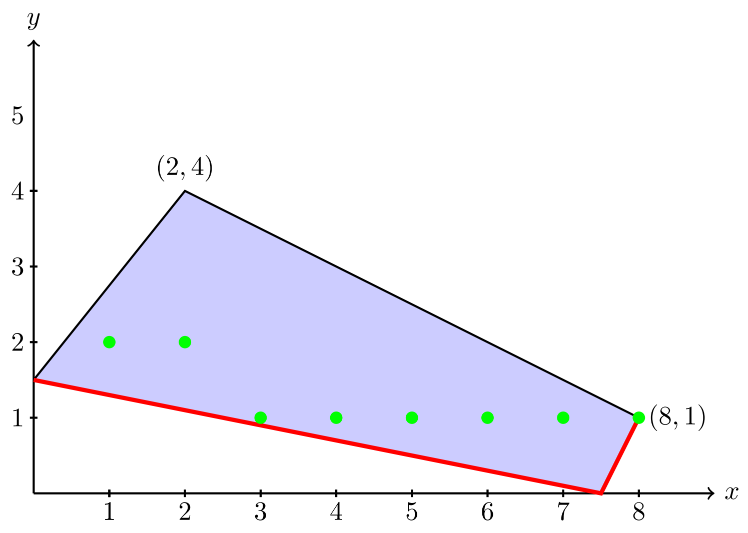


Figure 2.3: An Example From Bard and Moore [34]

where $S(x)$ is the set of optimal solution of the x-parameterized LP

$$\min_{y \in \mathbb{R}^{n_y}} \quad d^T y \tag{2.5a}$$

$$\text{subject to} \quad Ax + By \geq b \tag{2.5b}$$

The inequality (2.5b) can be rewritten as

$$By \geq b - Ax \tag{2.5c}$$

Let $x \in \mathbb{R}^{n_x}$ be arbitrary but fixed. The right hand side of Inequality (2.5c) can be considered as a constant. The dual LP to (2.5) becomes

$$\min_{z \in \mathbb{R}^{\ell}} \quad z^T(b - Ax) \tag{2.6a}$$

$$\text{subject to} \quad z^T B = d^T \tag{2.6b}$$

$$z \geq 0 \tag{2.6c}$$

Using complementary slackness criterion, we have that $y \in \mathbb{R}^n_y$ is an optimum feasible solution to (2.5), if and only if there exists a feasible solution $z \in \mathbb{R}^{\ell}$ to (2.6) such that

$$z^T(b - Ax - By) = 0$$

We can thus reformulate our BLP as the following single level optimization problem

$$\min_{x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}, z \in \mathbb{R}^{\ell}} \quad c_x^T x + c_y^T y \tag{2.7a}$$

$$\text{subject to} \quad G_x x + G_y y \geq q \tag{2.7b}$$

$$Ax + By \geq b \tag{2.7c}$$

$$z^T B = d^T \tag{2.7d}$$

$$z \geq 0 \tag{2.7e}$$

$$z^T(b - Ax - By) = 0 \tag{2.7f}$$

If we drop Equation (2.7d)–(2.7f), we obtain the HPR of our BLP. The constraint $d^T y = \Phi(x)$ is implicitly defined by the constraints (2.7d)–(2.7f), since the existence of a $z \in \mathbb{R}^{\ell}$ that satisfies (2.7d)–(2.7f) shows the optimality of y for the follower LP.

Evaluating constraint (2.7f) row-wise yields the equivalent constraint

$$z_i(b_i - A_i x - B_i y) = 0, \quad \text{for all } i = 1, \ldots, \ell$$

where A_i and B_i denote the *ith* row of A and B respectively. Rather than seeing this constraint as a product, we can see it as a disjunction

$$\text{either } z_i = 0 \text{ or } b_i - A_i x - B_i y = 0 \text{ for all } i = 1, \dots, \ell$$

Let M be a sufficiently large constant. We define binary variables s_i for all $i = 1, \dots, \ell$. The optimization problem (2.7) can be rewritten into the following MILP

$$\begin{aligned}
\min_{x\in\mathbb{R}^{n_x}, y\in\mathbb{R}^{n_y}, z\in\mathbb{R}^{\ell}, s\in\{0,1\}^n} \quad & c_x^T x + c_y^T y && (2.8a)\\
\text{subject to} \quad & G_x x + G_y y \geq q && (2.8b)\\
& Ax + By \geq b && (2.8c)\\
& z^T B = d^T && (2.8d)\\
& z \geq 0 && (2.8e)\\
& z_i \leq M s_i, \quad \text{for all } i = 1, \dots, \ell && (2.8f)\\
A_i x + B_i y - M(1 - s_i) \leq b_i, & \quad \text{for all } i = 1, \dots, \ell && (2.8g)\\
& s_i \in \{0, 1\}, \quad \text{for all } i = 1, \dots, \ell && (2.8h)
\end{aligned}$$

Constraints (2.8f)–(2.8h) can be interpreted as follows. If $s_i = 0$, then $Ms_i = 0$ which implies $z_i = 0$. On the other hand, $M(1 - s_i) = M$ thus the constraint (2.8g) will always be fulfilled by assumption that M is sufficiently large. If $s_i = 1$ then $Ms_i = M$ and constraint (2.8f) will always be fulfilled. In this case, however, $M(1 - s_i) = 0$ which implies $A_i x + B_i y \leq b_i$ for all $i \in \{1, \dots, \ell\}$ and thus together with contraint (2.8c), we get $b_i - A_i x - B_i y = 0$.

2.4 Solving IBLP with Branch and Bound

Our goal in this section is to modify the branch and bound procedure for ILP to solve IBLP. To the best of our knowledge, all algorithms for solving IBLP require some simplifying assumptions about the upper and lower level problem, either to guarantee convergence of the algorithm or to guarantee attainability of supremum. The first branch and bound scheme for solving IBLP was proposed by Baard and Moore [34]. Their scheme assumes that the IBLP has no coupling constraints and that all variables have upper- and lower-bounds. The scheme utilizes the relaxed BLP to generate primal bounds and high point relaxations to generate dual bounds. Xu and Wang developed in [45] a branch and bound scheme that works with coupling constraints. We present in this section a branch and bound scheme by Fischetti et al. [20], which builds upon the work by Xu and Wang. As in the scheme by Xu and Wang, this scheme allows coupling constraints while also improving it by allowing weaker assumptions for the general MIBLP case, which is beyond the scope of this thesis. To guarantee finiteness of the branch and bound scheme upper- and lower-bounds all variables are still required. We denote with $\bar{y}$ and $\underline{y}$ the upper and lower bounds of y in the upper-level program and $\bar{x}$ and $\underline{x}$ the upper and lower bounds of x in the upper-level program. Furthermore, we denote with l and u the upper- and lower-bounds of y in the follower program.

We consider an instance of IBLP in its value-function reformulation

$$\begin{aligned}
\min_{x\in\mathbb{Z}^{n_x}, y\in\mathbb{Z}^{n_y}} \quad & c_x^T x + c_y^T y && (2.9a)\\
\text{subject to} \quad & G_x x + G_y y \geq q && (2.9b)\\
& Ax + By \geq b && (2.9c)\\
& \underline{x} \leq x \leq \bar{x} && (2.9d)\\
& \underline{y} \leq y \leq \bar{y} && (2.9e)\\
& d^T y = \Phi(x) && (2.9f)
\end{aligned}$$

where we define for a given $x^* \in \mathbb{R}^{n_x}$

$$\begin{aligned}
\Phi(x^*) := \min_{y\in\mathbb{Z}^{n_y}} \quad & d^T y && (2.10a)\\
\text{subject to} \quad & Ax^* + By \geq b && (2.10b)\\
& \ell \leq y \leq u. && (2.10c)
\end{aligned}$$

For any feasible solution (x^*, y^*) of HPR, y^* is always a feasible solution of the x^*-parameterized follower problem (2.10), since constraints of the follower problem are also constraints of the HPR. Since y is bounded by l and u in the follower program, the follower program is bounded and has an optimal solution. Thus, the function $\Phi(x)$ is well-defined for all HPR solution (x, y). Additionally, note that any HPR solution $(x^*, \hat{y})$ is bilevel feasible if $d^T\hat{y} \le \Phi(x)$. We introduce the following subroutine.

Algorithm 5 Get Bilevel Feasible Solution (Fischetti et al. [20, Algorithm 1])

```
Input: An HPR Solution $(x^*, y^*)$
Output: A bilevel feasible solution $(x^*, \hat{y})$ (if any)

1. Compute $\Phi(x^*)$ by solving the $x^*$-parameterized follower ILP
2. Define the restricted HPR by temporarily adding the constraint $x = x^*$ and
   $d^T y \le \Phi(x^*)$
3. Solve the restricted HPR
4. If the restricted HPR is infeasible, then there exists no bilevel solution in the
   form $(x^*, \cdot)$. Otherwise, return the optimal solution to the restricted HPR $(x^*, \hat{y})$
```

We present in Algorithm 6 the full branch and bound scheme for IBLP. For $j \in \{1, \dots, n_x\}$ and a given node S^i, we denote with $\bar{x}^i_j$ the upper bound of x_j at node S^i and $\underline{x}^i_j$ the lower bound of x_j at node S^i.

Theorem 2.7 (Fischetti et al. [20, Theorem 2]). *Algorithm 6 correctly solves an IBLP and terminates after a finite number of iteration.*

Proof. Since all variables are integral and bounded, there exist at most finitely many possible nodes. At each iteration, a new node is evaluated. Thus, the algorithm is finite. Step (5a) and (6) compute the best bilevel feasible solution for the current node. Thus, the solution is optimal. □

Notice that the scheme also allows branching on lower-level variables. This is non-standard in IBLP literature (cf. Bard and Moore [34] and Xu and Wang [45]). Fischetti et al. in [20] noted that we exploit the underlying MILP solver as much as possible by allowing branching on lower-level variables. Whenever bilevel feasibility is checked, $\Phi(x^*)$ is evaluated with the upper- and lower-bounds for y being u and l respectively. Thus, the calculation of $\Phi(x^*)$ is completely "blind" to the branching.

2.5 Intersection Cut For Bilevel Programming

DeNegre remarked in [13] that the dual bounding method proposed by Baard and Moore is too weak to be effective on interesting problems. To tackle this, DeNegre and Ralphs in [14] expanded on the result by Baard and Moore into a branch and cut framework for solving IBLP, the first of such a proposal. In this section, we present a method for generating intersection cut (IC) introduced by Fischetti et al. in [20]. As in the case of ILPs, the generated cuts are locally valid. To generate IC cuts, we need a closed and convex set C such that the bilevel infeasible solution (x^*, y^*) that we want to cut lies in the interior of the set C and all interior points of the set C are bilevel infeasible. We consider the same IBLP as in Section 2.4. Additionally, we assume that all the entries of matrix A, B and vector b are integer (this can easily be fulfilled by scaling the constraints). Let $\mathbf{1} = (1, \dots, 1)^T$ be a column vector of ones of suitable size. We introduce the following *bilevel-free polyhedra*, that is, polyhedra that contain no bilevel feasible sets in their interior.

Theorem 2.8 (Fischetti et al. [20, Theorem 4]). *If all the entries of matrix A, B and vector b are integer, then for any $\hat{y} \in \mathbb{Z}^{n_y}$ that satisfies $l \le \hat{y} \le u$, the set*

$$S^+(\hat{y}) = \{(x, y) \in \mathbb{R}^n \mid d^T y \ge d^T \hat{y}, Ax + B\hat{y} \le b + \mathbf{1}\}$$

does not contain any bilevel feasible point in its interior.

Algorithm 6 Branch And Bound For IBLP (Fischetti et al. [20, Algorithm 2])

Input: An IBLP instance

Output: An optimal solution for the given IBLP (if any)

1. *Initialize:* Let S^0 be the HPR of our original IBLP. Set the incumbent lower bound $\underline{Z} = -\infty$, and the incumbent solution (x^*, y^*) to void. We initiate an empty list N of ILP and add S^0 to the list N.

2. *Select Node:* If N is empty, go to step 8. Otherwise, pick an S^i from N

3. *Solve LP Relaxation:* If the LP relaxation of S^i is infeasible, return to step 2. Otherwise, let $\bar{Z}^i$ be the dual bound of S^i.

4. *Fathom By Bounding:* If $\bar{Z}^i \leq \underline{Z}$, fathom node by bound. Return to step 2.

5. *LP Relaxation Is Unbounded:* If $\bar{Z}^i = \infty$, we divide the following two cases:

   (a) *Find Bilevel Feasible Solution:* If $\bar{x}^i = \underline{x}^i$, then compute $\Phi(\bar{x}^i)$ by solving the $\bar{x}$-parameterized follower ILP. If the problem is infeasible, fathom node by infeasibility. Otherwise, solve HPR with additional constraints $x = \bar{x}^i$ and $d^T y \leq \Phi(\bar{x}^i)$. If the problem is unbounded, IBLP is unbounded. If the problem is infeasible, fathom node by infeasibility. Otherwise, the optimum solution $(\tilde{x}, \tilde{y})$ is a bilevel feasible solution with value $Z := c^T\tilde{x} + c^T\tilde{y}$. If $Z < \underline{Z}$, update the incumbent solution to $(\tilde{x}, \tilde{y})$ and set $\underline{Z} = Z$. Fathom the node by optimality.

   (b) *Branch:* If $\bar{x}^i \neq \underline{x}^i$, select any $j \in \{1, \ldots, n_x\}$ with $\bar{x}^i_j \neq \underline{x}^i_j$. We define sub-problems S^i_1 to be the ILP S^i with the additional constraint $x_j \leq \lfloor \frac{\bar{x}^i_j + \underline{x}^i_j}{2} \rfloor$ and S^i_2 to be the MILP S^i with the additional constraint $x_j \geq \lceil \frac{\bar{x}^i_j + \underline{x}^i_j}{2} \rceil$, that is, we branch on variable x_j.

   Return to step 2.

6. *Fathom By Optimality:* If $\bar{Z}^i < \infty$, let $(\tilde{x}, \tilde{y})$ be the optimal solution corresponding to the dual bound. If any entry of $(\tilde{x}, \tilde{y})$ is nonintegeral, go to step 7. Otherwise, compute $\Phi(\tilde{x})$ by solving the $\tilde{x}$-parameterized follower ILP. One of the following two cases may happen:

   (a) If $d^T\tilde{y} \leq \Phi(\tilde{x})$, then $(\tilde{x}, \tilde{y})$ is a bilevel feasible solution with value $Z := c^T\tilde{x} + c^T\tilde{y}$.

   (b) If $d^T\tilde{y} > \Phi(\tilde{x})$, use Algorithm 5 to try to get a bilevel feasible solution $(\tilde{x}, \hat{y})$. If no bilevel feasible solution exist, set $Z = \infty$. Otherwise, we define $(\tilde{x}, \tilde{y}) := (\tilde{x}, \hat{y})$ and $Z := c^T\tilde{x} + c^T\hat{y}$

   If $Z < \underline{Z}$, update the incumbent solution to $(\tilde{x}, \tilde{y})$ and set $\underline{Z} = Z$. Fathom the node by optimality. Return to step 2.

7. *Branch On Fractional Variable:* Select a fractional variable $\tilde{x}_j \notin \mathbb{Z}$ or $\tilde{y}_k \notin \mathbb{Z}$. We define sub-problems S^i_1 to be the ILP S^i with the additional constraint $x_j \leq \lfloor \tilde{x}_j \rfloor$ (or $y_k \geq \lfloor \tilde{y}_k \rfloor$) and S^i_2 to be the MILP S^i with the additional constraint $x_j \geq \lceil \tilde{x}_j \rceil$ (or $y_k \geq \lceil \tilde{y}_k \rceil$). We add S^i_1 and S^i_2 to N and return to step 2.

8. *Terminate:* If (x^*, y^*) is void, $IBLP$ is infeasible. Otherwise, (x^*, y^*) is the optimal solution to the $IBLP$.

Proof. Let $\hat{y} \in \mathbb{Z}^{n_2}$ with $l \leq \hat{y} \leq u$ be arbitrary but fixed and (x', y') be an interior point of $S^+(\hat{y})$. It applies $d^T y' > d^T \hat{y}$ and $Ax' + B\hat{y} < b + \mathbf{1}$. If any entries of x' are non integral, then it is not bilevel feasible. Otherwise, $Ax' + B\hat{y}$ is integral because all entries of A and B are integral. Since b is integral, it applies that $Ax' + B\hat{y} \leq b$. Because $l \leq \hat{y} \leq u$ and $Ax' + B\hat{y} \leq b$, it follows that $\hat{y}$ is a feasible solution to the x'-parameterized follower ILP. Thus, we get

$$\Phi(x') \leq d^T \hat{y} < d^T y'$$

which shows that (x', y') is not bilevel feasible. □

Given an HPR optimum solution (x^*, y^*) that is bilevel infeasible, we want to select a $\hat{y} \in \mathbb{Z}^{n_y}$ with $l \leq \hat{y} \leq u$ such that the bilevel free polyhedra $S^+(\hat{y})$ generates deep IC cuts. Fischetti et al. [20] provided two method of choosing $\hat{y}$.

The first method, which we will call `SEP 1`, is to choose $\hat{y}$ as the optimum solution to the ILP

$$\min_{y \in \mathbb{Z}^{n_y}} d^T y \tag{2.11a}$$

$$Ax^* + By \leq b \tag{2.11b}$$

$$l \leq y \leq u, \tag{2.11c}$$

which is the follower problem with parameter x^*. Since (x^*, y^*) is a feasible HPR solution, the bilevel infeasibility of (x^*, y^*) implies $d^T \hat{y} < d^T y^*$. Furthermore, it applies $Ax^* + B\hat{y} \leq b$ and thus $Ax^* + B\hat{y} < b + \mathbf{1}$. Thus, (x^*, y^*) is an inner point of the $S^+(\hat{y})$. This method maximizes the distance of (x^*, y^*) to the facet $d^T y \geq d^T \hat{y}$. However, other facets can be close to (x^*, y^*)

The second method, which we call `SEP 2` relies on the following observations.

Lemma 2.9 (Fischetti et al. [20, Theorem 5]). *Let*

$$S = \{(x, y) \in \mathbb{R}^n \mid \alpha_i^T x + \beta_i^T y \leq \gamma_i,\ i = 1, \dots, k\}$$

be any polyhedron not containing bilevel-feasible points in its interior. For any $j \in \{1, \dots, k\}$ such that the halfspace $\{(x, y) \in \mathbb{R}^n \mid \alpha_j^T x + \beta_j^T y \geq \gamma_j\}$ does not contain any bilevel-feasible solution, it applies that the set

$$S' := \{(x, y) \in \mathbb{R}^n \mid \alpha_i^T x + \beta_i^T y \leq \gamma_i,\ i = 1, \dots j-1, j+1, \dots, k\}$$

does not contain any bilevel-feasible solution in its interior.

Proof. Since the half-space $\{(x, y) \in \mathbb{R}^n \mid \alpha_j^T x + \beta_j^T \geq \gamma_j\ \}$ does not contain any bilevel-feasible solution, neither does the half-line $\{(x, y) \in \mathbb{R}^n \mid \alpha_j^T x + \beta_j^T = \gamma_j\ \}$. The result follows from the equation

$$S' = \{(x, y) \in \mathbb{R}^n \mid \alpha_i^T x + \beta_i^T \leq \gamma_i,\ i = 1, \dots, k\} \cup \{(x, y) \in \mathbb{R}^n \mid \alpha_j^T x + \beta_j^T \geq \gamma_j\}$$ □

Lemma 2.10 (Fischetti et al. [20, Corollary 1]). *Let*

$$S = \{(x, y) \in \mathbb{R}^n \mid \alpha_i^T x + \beta_i^T y \leq \gamma_i,\ i = 1, \dots, k\}$$

be any polyhedron not containing bilevel-feasible points in its interior and $\bar{x}, \underline{x}, \bar{y}, \underline{y}$ be the upper and lower bound of the variables in our bilevel program. For any $j \in \{1, \dots, k\}$ with

$$\sum_{k=1}^{n_x} \max\{\alpha_{jk}\bar{x}_j, \alpha_{jk}\underline{x}_j\} + \sum_{k=1}^{n_y} \max\{\beta_{jk}\bar{y}_j, \beta_{jk}\underline{y}_j\} < \gamma_k, \tag{$\star$}$$

it applies that

$$S' := \{(x, y) \in \mathbb{R}^n \mid \alpha_i^T x + \beta_i^T \leq \gamma_i,\ i = 1, \dots j-1, j+1, \dots, k\}$$

does not contain any bilevel-feasible solution in its interior.

Proof. Let $j \in \{1, \dots, k\}$ be such that the condition $\star$ is fulfilled, we show that the halfspace

$$\{(x, y) \in \mathbb{R}^n \mid \alpha_j^T x + \beta_j^T y \geq \gamma_j\}$$

does not contain any bilevel-feasible solution. Any feasible solution (x^*, y^*) must satisfy $\underline{x} \leq x \leq \bar{x}$ and $\underline{y} \leq y \leq \bar{y}$ thus

$$\alpha_j^T x^* + \beta_j^T y^* \leq \sum_{k=1}^{n_x} \max\{a_{jk}\bar{x}_j, a_{jk}\underline{x}_j\} + \sum_{k=1}^{n_y} \max\{\beta_{jk}\bar{y}_j, \beta_{jk}\underline{y}_j\} < \gamma_k.$$

The result follows from Lemma 2.9 □

The two lemmas give us a sufficient condition in which we can remove constraints from our polyhedron while maintaining the bilevel free interior property.

Theorem 2.11. *Assume that all entries of matrix A, B and vector b are integer. We denote with A_i the ith row of A and B_i the ith row of B. Let $l \leq \hat{y} \leq u$ and $S^+(\hat{y})$ be defined as in Theorem 2.8 and $i \in \{1, \dots, \ell\}$ an index that satisfies*

$$\sum_{j=1}^{n_x} \max\{A_{ij}\bar{x}_j, A_{ij}\underline{x}_j\} + B_i\hat{y} \leq b_i$$

where $\bar{x}$ and $\underline{x}$ are the upper- and lower- bounds of our leader variables. The set

$$\bar{S}^+(\hat{y}) = \{(x, y) \in \mathbb{R}^n \mid d^T y \geq d^T \hat{y},\ A_i x + B_i \hat{y} \leq b_i + 1 \text{ for all } i = 1, \dots, j-1, j+1, \dots, n\}$$

does not contain any bilevel feasible set in its interior.

Proof. By Theorem 2.8 $S^+(\hat{y})$ does not contain any bilevel feasible solution in its interior. Let i be an index that satisfies the required property. Consider the constraint

$$A_i x + B_i \hat{y} \leq b_i + 1.$$

Since $\hat{y}$ is given $B_i\hat{y}$ is a constant and we can write equivalently

$$A_i x \leq b_i + 1 - B_i\hat{y}$$

By Lemma 2.10 this constraint can be removed if

$$A_i x < b_i + 1 - B_i\hat{y}$$

However, we have assumption that

$$A_i x + B_i\hat{y} \leq \sum_{j=1}^{n_x} \max\{A_{ij}\bar{x}_j, A_{ij}\underline{x}_j\} + B_i\hat{y} \leq b_i < b_i + 1.$$

Thus, the constraint can be removed. □

We observe that using upper- and lower-bounds of the upper-level variables on the current node yields locally valid cut. Assume that the cost vector of the lower level program d is integer (which can be achieved by scaling). Given a feasible HPR solution (x^*, y^*), the method `SEP 2` is to choose a $\hat{y}$ such that we can remove as much constraints as possible from $S^+(\hat{y})$ according to Theorem 2.11, that is, we want to select $\hat{y}$ such that the condition

$$\sum_{j=1}^{n_x} \max\{A_{ij}\bar{x}_j, A_{ij}\underline{x}_j\} + B_i\hat{y} \leq b_i \tag{2.12}$$

to be fulfilled by as much index i as possible. This can be done by solving the following MILP.

$$\begin{aligned}
\min_{\hat{y} \in \mathbb{Z}^n, w \in \{0,1\}^\ell} \quad & \sum_{i=1}^{\ell} w_i && (2.13a)\\
\text{subject to} \quad & d^T \hat{y} \leq d^T y^* - 1 && (2.13b)\\
& B\hat{y} + s = b && (2.13c)\\
& s_i + (L_i^{\max} - L_i^*) w_i \geq L_i^{\max}, \quad \text{for all } i = 1, \dots, m && (2.13d)\\
& l \leq \hat{y} \leq u && (2.13e)
\end{aligned}$$

where,

$$L_i^* := A_i x^* \text{ and } L_i^{\max} := \sum_{j=1}^{n_x} \max\{A_{ij}\bar{x}_j, A_{ij}\underline{x}_j\}$$

for each $i = 1, \dots, \ell$.

If $w_i = 1$ then $s_i \geq L_i^*$ by Equation (2.13d) and $B_i\hat{y} + L_i^* = A_i x^* + B_i\hat{y} \leq b_i$ by Equation (2.13c). On the other hand, if $w_i = 0$ then $s_i \geq L_i^{\max}$ by Equation (2.13d) and $B_i\hat{y} + L_i^{\max} \leq b_i$ by Equation (2.13c). Notice that since, we have a minimization problem $w_i = 0$ if and only if Equation (2.12) is fulfilled for index i. Since $L_i^* \leq L_i^{\max}$, it applies that $A_i x^* + B_i\hat{y} \leq b_i$ for all $i = 1, \dots, \ell$ together with Equation 2.13b, we get that (x^*, y^*) is an interior point of $S^+(\hat{y})$.

Chapter 3

Minimum Sudoku Clue Problem

3.1 Problem Description

Given a square number $n \in \mathbb{N}$ and $s := \sqrt{n}$, an $n \times n$ *sudoku* grid is a matrix $G \in \{1, \dots, n\}^{n \times n}$ with the following properties:

(G1) Every number between 1 and n appear *exactly once* in each row of G

(G2) Every number between 1 and n appear *exactly once* in each column of G

(G3) If we partition the matrix G into n submatrices of size $s \times s$, every number between 1 and n appear *exactly once* in each submatrix.

An example of a 9×9 grid with 3×3 submatrices is given in Figure 3.1b. A position $(i, j) \in \{1, \dots, n\}^2$ is called a *cell.* A group of s submatrices on the same row is called a *band.* A 9×9 grid have 3 band with row 1–3, row 4–6, and row 7–9 each forming a band. Analogously, a group of s submatrices on the same column is called a *stack.* There exist variations of sudoku with non-square submatrices. The models introduced in this chapter are constrained to grids with square submatrices. However, they can easily be generalized for the non-square variations. Given two sudoku grids G_1 and G_2 the *distance* between G_1 and G_2 is the number of cells where G_1 and G_2 differ. A *sudoku puzzle* is a matrix $P \in \{0, \dots, n\}^{n \times n}$ with the following properties:

(P1) Every number between 1 and n appear *at most once* in each row of P

(P2) Every number between 1 and n appear *at most once* in each column of P

(P3) If we partition the matrix P into n submatrices of size $s \times s$, every number between 1 and n appear *at most once* in each submatrix.

The non-zero entries of a puzzle are called *clues* while the zero entries are called *empty* cells. An example of a sudoku puzzle is given in Figure 3.1a. The cells with no numbers are empty cells. By definition a sudoku grid is also a sudoku puzzle (albeit with zero empty cells), we call puzzles that have at least one empty cell a *proper* sudoku puzzle.

We call an $n \times n$ sudoku grid G a *completion* of an $n \times n$ sudoku puzzle P if the clues of P coincide with G, that is, for all $i, j = 1, \dots, n$ with $P_{ij} \neq 0$ it applies $P_{ij} = G_{ij}$. We call a puzzle P *valid*, if it has exactly one completion, *infeasible* if it has no completion, and *invalid* if it has multiple completions.

The *sudoku solving* problem is the following: given a sudoku puzzle P, find a sudoku grid G that is a completion of P or determine that no such grid exists. We will formulate an integer program to solve this problem on Section 3.3.

The *minimum sudoku clue* (MSC) problem, which will be the main focus of the chapter, asks the "reverse" question as an optimization problem: Given a sudoku grid G, find a sudoku puzzle P with the minimum number of clues such that P is a valid puzzle and the unique completion of P is G. The optimization problem is always feasible because G itself will always be a puzzle whose only completion is itself. The extreme answer of a puzzle with no clues is an infeasible solution to the optimization problem because although G is a completion for a puzzle with no clues, any other sudoku grid G' is also a completion of the

			6	4		2		
1		8					3	
		7		1	8			
	6					5		
3							1	
4			2					
	2		5					

(a) A Sudoku Puzzle with 17 Clue

7	9	3	6	4	5	2	8	1
1	5	8	7	9	2	4	3	6
6	4	2	1	8	3	7	9	5
5	3	7	4	1	8	6	2	9
9	6	1	3	2	7	5	4	8
2	8	4	9	5	6	1	7	3
3	7	5	8	6	4	9	1	2
4	1	6	2	3	9	8	5	7
8	2	9	5	7	1	3	6	4

(b) A Completion Of The Puzzle 3.1a

Figure 3.1: A 17 Clue Sudoku Puzzle

puzzle. An example instance is given in Figure 3.1. The given grid is shown in Figure 3.1b. The puzzle shown in Figure 3.1a is a feasible solution to the optimization and as we will soon see, an optimum solution. The optimum value of MSC problem in this example is 17.

The rest of this chapter will focus on 9×9 sudoku grids. However, the models proposed generalizes to sudoku grids of other sizes.

3.2 Literature Review

Yato and Seta showed that solving a general $n \times n$ sudoku puzzle is NP-Complete [46]. Despite this complexity result, numerous methods for solving puzzles exist such as those proposed by Koch [28], and Norvig [35]. Felgenhauer and Jarvis showed in [18] that the number of possible 9×9 sudoku grids is $6,670,903,752,021,072,936,960 \approx 6.671 \times 10^{21}$. Many of these grids are interchangeable with each other and are identical under transformation through permuting the numbers, rotating the board, and other symmetries. Taking these transformations into account, Russell and Jarvis reduced the number of possible 9×9 sudoku grids to $5,472,730,538$ *essentially different* grids [38].

The minimum number of givens that any valid 9×9 puzzle has is long conjectured to be 17. Gordon Royle has collected nearly 50000 instances of valid 9×9 puzzles with 17 clues these instances available on his homepage [37]. In their landmark paper [32], McGuire et Al. prove this result via a computer-assisted proof taking over 7.1 million core hours with Intel X5650 hex-core processors being used. This result is of key interest to this paper for two reasons: The first it shows that the optimum value for any minimum sudoku clue problem instance with 9×9 grid size is bounded by 17. The second is that the method used in the paper to check a given sudoku grid for 16 clue puzzle inspired the strengthening that we will introduce in Section 3.5. To the author's knowledge at the time of writing, there is no purely mathematical proof which shows that any valid puzzle for 9×9 grid have at least 17 clues.

Although theoretically brute forcing all possible puzzles is a worse idea than solving a bilevel formulation, there exist software for brute force which are often faster. Examples of such software are the one used by Gary McGuire [32] and the one developed by Mladen Dobrichev[1]. These softwares also uses the concept of unavoidable sets that we will elaborate later in Section 3.5. Since they are brute force approaches, there exist corner cases in which these software perform badly. Unfortunately, these are often anecdotal and not documented properly. There are three reasons why these software often perform better than our proposal: First, the brute force approaches first convert sudoku grids into normal forms to break symmetries, discussion of symmetry breaking techniques is beyond the scope of this thesis. Second, they use pattern matching algorithm to identify unavoidable sets, which is much faster than the ILP program we suggested in this thesis (albeit at the cost of requiring more work to generate the patterns in the first place). Lastly, they have better hitting set

[1]https://github.com/dobrichev/gridchecker

enumerators because of the use of additional specialised heuristics as discussed by McGuire in [32].

3.3 Solving Sudoku Using ILP

We now introduce a integer linear programming model for solving sudoku puzzles. This section is loosely based on the work of Hürlimann [24]. We are given an $n \times n$ sudoku puzzle P and want to find a sudoku grid G such that G is a completion of P. We define binary decision variables

$$x_{ijk} := \begin{cases} 1, & \text{if } G_{ij} = k \\ 0, & \text{otherwise} \end{cases}$$

which will be the characteristic vector of the grid G. Since G_{ij} must have exactly one entry for all $i, j \in \{1, \dots, n\}$, we have the constraint

$$\sum_{k=1}^{n} x_{ijk} = 1 \quad \text{for all } i, j \in \{1, \dots, n\} \tag{$G0$}$$

We now add the constraints which requires G to be a valid sudoku grid. For each row $i \in \{1, \dots, n\}$ of G, each number $k \in \{1, \dots, n\}$ appear exactly once.

$$\sum_{j=1}^{n} x_{ijk} = 1 \quad \text{for all } i, k \in \{1, \dots, n\} \tag{$G1$}$$

Analogously for each column $j \in \{1, \dots, n\}$ of G, each number $k \in \{1, \dots n\}$ appear exactly once.

$$\sum_{i=1}^{n} x_{ijk} = 1 \quad \text{for all } j, k \in \{1, \dots, n\} \tag{$G2$}$$

Lastly, we model the requirement that each submatrix contains each number $k \in \{1, \dots, n\}$ exactly once (remember that $s := \sqrt{n}$).

$$\sum_{i=sp-s+1}^{sp} \sum_{j=sq-s+1}^{sq} x_{ijk} = 1 \quad \text{for all } p, q \in \{1, \dots, s\} \text{ and } k \in \{1, \dots, n\} \tag{$G3$}$$

To give an intuition for the last constraint, we consider a single constraint of this type on a 9×9 puzzle. For $p = 2$ and $q = 3$, the constraint corresponds to the right most submatrix of the middle band. Our summation goes over all i and j with

$$sp - s + 1 = 4 \leq i \leq 6 = sp \text{ and } sq - s + 1 = 7 \leq j \leq 9 = sq$$

which is exactly the summation that we want. Together we call constraints $(G0)$ – $(G3)$ the *grid constraints*. A binary vector x is a characteristic vector of a sudoku grid if and only if it fulfills all the grid constraints.

Additionally, we have the constraints that entries given in P as clues must be fixed.

$$x_{ijP_{ij}} = 1 \quad \text{for all } i, j \in \{1, \dots, n\} \text{ with } P_{ij} \neq 0 \tag{F1}$$

We call constraint $(F1)$ the *fixed constraint*. For example, in Figure 3.1a the cell in the second row and third column has a value of 8, thus $x_{2,3,8} = 1$. We can set the objective function of our ILP to be $0^T x$ because any binary vector x which satisfies $(G0)$–$(G3)$ and $(F1)$ is the characteristic vector of a sudoku grid that completes P. To summarize, we have the following integer linear program, which we will call `SUDOKU`.

$$
\begin{array}{llll}
\max\limits_{x} & 0^T x & & \\
\text{subject to} & \displaystyle\sum_{k=1}^{n} x_{ijk} = 1, & \text{for all } i,j \in \{1,\dots,n\} & (G0) \\
& \displaystyle\sum_{j=1}^{n} x_{ijk} = 1, & \text{for all } i,k \in \{1,\dots,n\} & (G1) \\
& \displaystyle\sum_{i=1}^{n} x_{ijk} = 1, & \text{for all } j,k \in \{1,\dots,n\} & (G2) \\
& \displaystyle\sum_{i=sp-s+1}^{sp}\sum_{j=sq-s+1}^{sq} x_{ijk} = 1, & \text{for all } p,q \in \{1,\dots,s\} \text{ and } k \in \{1,\dots,n\} & (G3) \\
& x_{ijP_{ij}} = 1, & \text{for all } i,j \in \{1,\dots,n\} \text{ with } P_{ij} \neq 0 & (F1) \\
& x_{ijk} \in \{0,1\}, & \text{for all } i,j,k \in \{1,\dots,n\} &
\end{array}
$$

The ILP `SUDOKU` is feasible if there exists a grid G that completes P and infeasible if P is an infeasible puzzle. The ILP `SUDOKU` can also be equivalently written as

$$
\begin{array}{llll}
\max\limits_{x} & 0^T x & & \\
\text{subject to} & \displaystyle\sum_{k=1}^{n} x_{ijk} \geq 1, & \text{for all } i,j \in \{1,\dots,n\} & (G'0) \\
& \displaystyle\sum_{j=1}^{n} x_{ijk} \leq 1, & \text{for all } i,k \in \{1,\dots,n\} & (G'1) \\
& \displaystyle\sum_{i=1}^{n} x_{ijk} \leq 1, & \text{for all } j,k \in \{1,\dots,n\} & (G'2) \\
& \displaystyle\sum_{i=sp-s+1}^{sp}\sum_{j=sq-s+1}^{sq} x_{ijk} \leq 1, & \text{for all } p,q \in \{1,\dots,s\} \text{ and } k \in \{1,\dots,n\} & (G'3) \\
& x_{ijP_{ij}} \geq 1, & \text{for all } i,j \in \{1,\dots,n\} \text{ with } P_{ij} \neq 0 & (F1) \\
& x_{ijk} \in \{0,1\}, & \text{for all } i,j,k \in \{1,\dots,n\}, &
\end{array}
$$

This reformulation would be useful in later sections. Notice that x is a feasible solution to our original formulation if and only if x is also a feasible solution to this reformulation. This follows from the observation that if all squares are filled, and a number between 1 until n does not appear in a particular column, row, or submatrix, then another number must appear twice in that column, row, or submatrix.

3.4 A Simple Bilevel Model For MSC

We introduce an IBLP model for MSC. Firstly, we introduce the *sudoku puzzle validity* problem: given an $n \times n$ sudoku puzzle P determine if P is valid, invalid or infeasible. To solve this problem, we first solve the ILP `SUDOKU` with input P. We get a grid G that completes P or determine that P is infeasible. In the former case, it remains to determine if P is valid or invalid. We do this by solving the following integer linear program, which we will refer to as `VALID`:

$$
\begin{aligned}
\min_{x,z} \quad & z \\
\text{subject to} \quad & \sum_{k=1}^{n} x_{ijk} + z \geq 1, \quad \text{for all } i,j \in \{1,\dots,n\} && (G'0)\\
& \sum_{j=1}^{n} x_{ijk} \leq 1, \quad \text{for all } i,k \in \{1,\dots,n\} && (G'1)\\
& \sum_{i=1}^{n} x_{ijk} \leq 1, \quad \text{for all } j,k \in \{1,\dots,n\} && (G'2)\\
& \sum_{i=sp-s+1}^{sp} \sum_{j=sq-s+1}^{sq} x_{ijk} \leq 1, \quad \text{for all } p,q \in \{1,\dots,s\} \text{ and } k \in \{1,\dots,n\} && (G'3)\\
& x_{ijP_{ij}} \geq 1, \quad \text{for all } i,j \in \{1,\dots,n\} \text{ with } P_{ij} \neq 0 && (F1)\\
& \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijG_{ij}} \leq n^2 - 1 && (N1)\\
& x_{ijk}, z \in \{0,1\}, \quad \text{for all } i,j,k \in \{1,\dots,n\}.
\end{aligned}
$$

The ILP `VALID` tries to construct another grid G' which completes P. If $z = 1$, then cells are allowed to be empty. Note that in particular, puzzle P is a feasible solution to the ILP with $z = 1$. If $z = 0$, then ILP `VALID` has the same constraints as ILP `SUDOKU` with the additional constraint $(N1)$. Thus, the binary vector x defines a sudoku grid that is a completion of P. The constraint $(N1)$ requires that the sudoku grid defined by x have at most $n^2 - 1$ same entries as G. We call constraint $(N1)$ the *no-good cut* constraint. Consequentially if the ILP `VALID` has an optimum solution of 0, that is, if there exists a pair (x, z) that satisfies all constraints with $z = 0$, we can construct another sudoku grid G' with $G \neq G'$ yet G' completes P, which implies that our sudoku puzzle P is invalid. On the other hand, if the ILP `VALID` has an optimum solution of 1, then such a construction is not possible and G is a unique completion of P, which implies that P is a valid puzzle.

We now construct an IBLP model for the MSC problem. Given as input a sudoku grid G, we formulate MSC as the following IBLP:

$$
\begin{aligned}
\min_{x,y,z} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} \\
\text{subject to} \quad & z = 1 && (V1)\\
& y_{ij} \in \{0,1\}, \quad \text{for all } i,j \in \{1,\dots,n\}\\
& (x,z) \in S(y)
\end{aligned}
$$

where $S(y)$ is the set of optimum solutions to the y-parameterized follower problem

$$
\begin{aligned}
\min_{x,z} \quad & z \\
\text{subject to} \quad & \sum_{k=1}^{n} x_{ijk} + z \geq 1, \quad \text{for all } i,j \in \{1,\dots,n\} && (G'0)\\
& \sum_{j=1}^{n} x_{ijk} \leq 1, \quad \text{for all } i,k \in \{1,\dots,n\} && (G'1)\\
& \sum_{i=1}^{n} x_{ijk} \leq 1, \quad \text{for all } j,k \in \{1,\dots,n\} && (G'2)\\
& \sum_{i=sp-s+1}^{sp} \sum_{j=sq-s+1}^{sq} x_{ijk} \leq 1, \quad \text{for all } p,q \in \{1,\dots,s\} \text{ and } k \in \{1,\dots,n\} && (G'3)\\
& x_{ijG_{ij}} \geq y_{ij}, \quad \text{for all } i,j \in \{1,\dots,n\} && (F'1)\\
& \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijG_{ij}} \leq n^2 - 1 && (N1)\\
& x_{ijk}, z \in \{0,1\}, \quad \text{for all } i,j,k \in \{1,\dots,n\}.
\end{aligned}
$$

The upper-level decision variables of our IBLP y_{ij} determines which cells are given as clues. The constraints $(F'1)$ is a modification of the constraint $(F1)$ of ILP `VALID` that is dependent y. If $y_{ij} = 1$, then the cell (i, j) is given as a clue and the constraint $(F'1)$ is fulfilled if and only if $x_{ijG_{ij}}$ is fixed. Otherwise, if $y_{ij} = 0$, the constraint $(F'1)$ is always fulfilled. The y parameterized follower problem has an optimum value of 1 if and only if the puzzle defined by y is a valid puzzle. Thus, a solution vector (x, y, z) is bilevel feasible if and only if y defines a valid puzzle since the constraint $(V1)$ requires that $z = 1$. It follows that optimum solution of the IBLP constructs a valid puzzle in which the entry $(i, j) \in \{1, \ldots, n\}^2$ is given as a clue if and only if $y_{ij} = 1$ (note that the objective function of the leader problem is the number of clues given).

3.5 Unavoidable Sets

Consider the sudoku grid given in Figure 3.2. We observe that we can swap the 3 and 8 in the green marked cell to get a new sudoku grid G' that has the same entries except for the cells marked in green. Thus, any valid puzzle P that is completed by G must have at least 1 clue in one of the green marked cell otherwise both G and G' are completions of P and P becomes an invalid puzzle. Similarly, we observe that it is possible to change the entries of cells marked in blue or red to get a new sudoku grid with the same entries except for the cells marked in blue or red respectively. Thus, there must also be at least one clue in the cells marked red and one clue in the cells marked blue.

For a given sudoku grid G, we call a set of cell U an *unavoidable set* of G if there exists a sudoku grid $G' \neq G'$ that differs from G exactly on cells in U, that is, entries of the cells in U can be permuted to get a new sudoku grid. We call an unavoidable set *minimally unavoidable* if it contains no subset that is again unavoidable. The following proposition formalized the observation we made earlier.

Proposition 3.1. *Let G be a sudoku grid and P a puzzle such that G is a completion of P. Then P is a valid puzzle, if and only if, for every minimally unavoidable set U of G there exist a cell $(i, j) \in U$ which is given as a clue, that is, $P_{ij} \neq 0$.*

Proof. To show sufficiency suppose that there exist a minimally unavoidable set U such that $P_{ij} = 0$ for all cells $(i, j) \in U$. By definition of unavoidable set, there exist a sudoku grid $G' \neq G$ which differs from G only in the entries of cells that are in U. Since $P_{ij} = 0$ for all cells $(i, j) \in U$ then G' is also a completion of P. Thus P is not a valid puzzle.

To show neccessity suppose that P is not a valid puzzle and there exists G and G' which complete P, where $G \neq G'$. We define

$$U := \{(i, j) \in \{1, \ldots, n\}^2 \mid G_{ij} \neq G'_{ij}\}$$

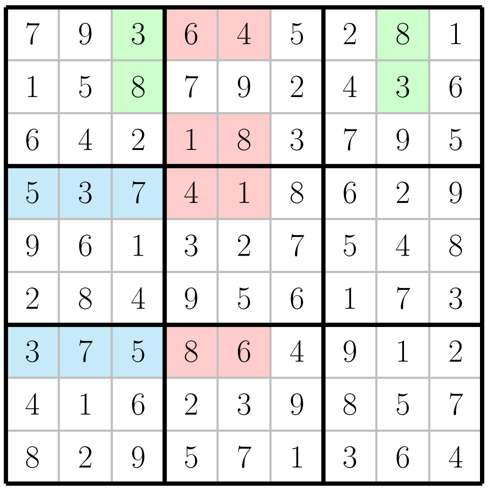
the set of cells whose entry in G is different from its entry in G'. It applies that U is by construction an unavoidable set and $P_{ij} = 0$ for all $(i, j) \in U$. If U is minimally unavoidable then we are done. Otherwise select any $\tilde{U} \subset U$ that is minimally unavoidable. Then $\tilde{U}$ is a minimally unavoidable set of G with $P_{ij} = 0$ for all $(i, j) \in \tilde{U}$ □

Let G be a sudoku grid. If we are given *apriori* all minimal unavoidable set of G then the minimum sudoku clue problem can be solved by finding a set of cells that hits all unavoidable sets, that is, the minimum sudoku clue problem becomes a minimum hitting set problem. McGuire et al. [32] uses this idea to find a lower bound for the number of clues in any valid sudoku puzzle. They search each sudoku grid (up to equivalent classes) for all unavoidable sets of size 12 or below. They then enumerate for each grid all possible hitting sets of size 16 and see if any of them is a valid sudoku puzzle. Proposition 3.1 give rise to Algorithm 7 to solve the MSC problem. We introduce a method to find a minimal unavoidable set in step 3.

Proposition 3.2. *Let G be a sudoku grid, $\mathcal{U}$ a set of minimally unavoidable sets of G, and H a hitting set that hits all elements of $\mathcal{U}$. Suppose that G' is a sudoku grid with $G' \neq G$ but $G_{ij} = G'_{ij}$ for all cells $(i, j) \in H$. We define*

$$U := \{(i, j) \in \{1, \ldots, n\}^2 \mid G_{ij} \neq G'_{ij}\}.$$

Then U is an unavoidable set that is not hit by any elements of H. If we further require that the grid G' have a minimum distance from G, then U is minimally unavoidable.



7	9	3	6	4	5	2	8	1
1	5	8	7	9	2	4	3	6
6	4	2	1	8	3	7	9	5
5	3	7	4	1	8	6	2	9
9	6	1	3	2	7	5	4	8
2	8	4	9	5	6	1	7	3
3	7	5	8	6	4	9	1	2
4	1	6	2	3	9	8	5	7
8	2	9	5	7	1	3	6	4

Figure 3.2: Unavoidable sets of the sudoku grid from 3.1b

Algorithm 7 Cutting Plane Algorithm For MSC

Input: A sudoku grid G

Output: An optimum solution to MSC

1. *Initialize:* Let $\mathcal{U}$ be an empty list of unavoidable sets of G
2. *Find Hitting Set:* Find a set of cells H with the minimum number of element that hits all of $\mathcal{U}$
3. *Cut:* Find a minimal unavoidable set U that is not hit by any elements of H, add it to $\mathcal{U}$ and go back to step 2. If no unavoidable set can be found, go to step 4.
4. *Terminate:* The puzzle in which the cells in H are given as clues is an optimal solution to MSC.

Proof. U is by construction an unavoidable set and is not hit by H because of the construction of G'. To show minimality, suppose that $\hat{U} \subset U$ is an unavoidable set with $\hat{U} \neq U$ then there exist G'' such that G'' and G differs only in cells that are in $\hat{U}$. Since $\hat{U} \subset U$ it applies that no elements of H hit $\hat{U}$. Thus $G''_{ij} = G_{ij}$ for all cells $(i,j) \in H$ but the number of entries of G'' that are different from G is lower than the number of entries of G' that are different from G contradicting the minimality requirement of G'. □

Suppose, we are now at an iteration of our algorithm with an $n \times n$ sudoku grid G as an input, a set $\mathcal{U}$ of minimally unavoidably sets of G and a hitting set H that hits all elements of $\mathcal{U}$. We consider the integer linear program

$$\begin{aligned}
\min_{x,d} \quad & d \\
\text{subject to} \quad & \sum_{k=1}^{n} x_{ijk} = 1, \quad \text{for all } i,j \in \{1,\dots,n\} && (G0)\\
& \sum_{j=1}^{n} x_{ijk} = 1, \quad \text{for all } i,k \in \{1,\dots,n\} && (G1)\\
& \sum_{i=1}^{n} x_{ijk} = 1, \quad \text{for all } j,k \in \{1,\dots,n\} && (G2)\\
& \sum_{i=sp-s+1}^{sp} \sum_{j=sq-s+1}^{sq} x_{ijk} = 1, \quad \text{for all } p,q \in \{1,\dots,s\} \text{ and } k \in \{1,\dots,n\} && (G3)\\
& x_{ijG_{ij}} = 1, \quad \text{for all } (i,j) \in H && (S1)\\
& \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijG_{ij}} + d = n^2 && (D1)\\
& d \geq 1\\
& x_{ijk} \in \{0,1\}, \quad \text{for all } i,j,k \in \{1,\dots,n\}\\
& d \in \mathbb{Z}
\end{aligned}$$

The vector x in the mixed integer linear program is the characteristic vector of some sudoku grid G' with d being the distance between G and G' because of constraint $(D1)$. Since $d \geq 1$, it applies $G \neq G'$. The constraint $(S1)$ enforces that $G'_{ij} = G_{ij}$ for all $(i,j) \in H$. Solving the mixed integer linear program gives us a grid G' of minimal distance to G among all grids G'' with $G''_{ij} = G_{ij}$ for all $(i,j) \in H$. Using Proposition 3.2, we get that

$$U := \{(i,j) \in \{1,\dots,n\}^2 \mid G_{ij} \neq G'_{ij}\}.$$

is a minimally unavoidable set that is not hit by any element of H. Thus, $U \notin \mathcal{U}$ and adding U to $\mathcal{U}$ increase the cardinality of our set of unavoidable sets $\mathcal{U}$. Note that if the ILP is infeasible, then H hits all minimal hitting sets of G.

Since there is only a finite amount of unavoidable sets Algorithm 7 will always terminate. The number of unavoidable sets, however, might still be extremely large. Moreover, the time needed to find a hitting set H will increase as the number of unavoidable sets increase. Although in practice it might be the case that $\mathcal{U}$ does not need to contain all unavoidable set before the algorithm terminates, there is no guarantee for this. For this reason, this algorithm is impractical for sudoku board of size 9×9 (It can, however, be used for board of size 4×4 or 6×6). However, it gives us an idea to improve our bilevel formulation.

Let G be a sudoku grid and U an arbitrary minimal unavoidable set of G. We remind ourselves that the upper-level problem to our integer bilevel linear program is

$$\begin{aligned}
\min_{x,y,z} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} \\
\text{subject to} \quad & z = 1 && (V1)\\
& (x,z) \in S(y)
\end{aligned}$$

where the vector y determines the entries of G that are given in a puzzle and $S(y)$ is the set of optimum solutions to the follower problem. By Proposition 3.1 for a puzzle to be a bilevel feasible solution, that is, for a puzzle to be valid, it must hit every minimal unavoidable set of G in particular it must hit U.

Corollary 3.3. *Let G be a sudoku grid and U an arbitrary minimal unavoidable set. Then, the cutting plane*

$$\sum_{(i,j)\in U} y_{ij} \geq 1 \tag{U}$$

is a valid cutting plane for the upper level program of the minimum sudoku bilevel program. We call this cut the ***unavoidable set cut*** *corresponding to U*

Proof. The proof follows directly from Proposition 3.1. □

The algorithm we introduced earlier can thus be seen as a form of cutting plane algorithm to solve our bilevel problem. Rather than using cutting planes exclusively until we reach an optimal solution, we propose a mixed approach: we first generate a set of minimally unavoidable sets $\mathcal{U}$ and then use bilevel optimization to optimize over all hitting sets of $\mathcal{U}$. This is equivalent to adding the unavoidable set cuts corresponding to U for all $U \in \mathcal{U}$ to our upper level program of the minimum sudoku bilevel program.

We give a method to generate the set $\mathcal{U}$. Let $m \in \mathbb{N}$ with $m \geq 1$. Consider the m-parameterized integer linear program,

$$\begin{aligned}
\min_x \quad & 0^T x \\
\text{subject to} \quad & \sum_{k=1}^{n} x_{ijk} = 1, \quad \text{for all } i,j \in \{1,\dots,n\} && (G0)\\
& \sum_{j=1}^{n} x_{ijk} = 1, \quad \text{for all } i,k \in \{1,\dots,n\} && (G1)\\
& \sum_{i=1}^{n} x_{ijk} = 1, \quad \text{for all } j,k \in \{1,\dots,n\} && (G2)\\
& \sum_{i=sp-s+1}^{sp} \sum_{j=sq-s+1}^{sq} x_{ijk} = 1, \quad \text{for all } p,q \in \{1,\dots,s\} \text{ and } k \in \{1,\dots,n\} && (G3)\\
& \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijG_{ij}} = n^2 - m && (D1)\\
& x_{ijk} \in \{0,1\}, \quad \text{for all } i,j,k \in \{1,\dots,n\},
\end{aligned}$$

which we will refer to as `GENERATE`. The integer linear program gives us a sudoku grid $G' \neq G$ of distance m from G. We get that

$$U := \{(i,j) \in \{1,\dots,n\}^2 \mid G_{ij} \neq G'_{ij}\}.$$

is an unavoidable set of G by construction. We start with $m = 1$ and repeatedly solve `GENERATE`, adding in each iteration the no-good cut constraint

$$\sum_{(i,j)\in U} x_{ijG_{ij}} \geq 1 \tag{N2}$$

which bars `GENERATE` from returning any G', whose associated unavoidable set is a superset of U. `GENERATE` will thus return a different unavoidable set of size m in each iteration. Once all unavoidable set of size m has been generated, `GENERATE` will be infeasible and we can move on to $m + 1$.

Proposition 3.4. *For the procedure described above it applies*

(i) At each iteration, the resulting unavoidable set will always be a minimally unavoidable set

(ii) Repeating the procedure eventually yields all minimal unavoidable sets

Proof. To show (i), let $\hat{U}$ be an unavoidable set that is not minimal, $U \subset \hat{U}$ a minimal unavoidable set and $\tilde{m} := |U|$. When generating all unavoidable sets of size $\tilde{m}$, a no-good cut for U will also be added to the formulation. Thus, any G' which generates $\hat{U}$ will be infeasible because $U \subset \hat{U}$. To show (ii), let U be a minimally unavoidable set and $\tilde{m} = |U|$. Any grid G' which differs from G on all (and only on) cells that are in U is a feasible solution of `GENERATE` with parameter $\tilde{m}$. Further since no subset of U is an unavoidable set, no no-good cut will block any of these grids. Thus eventually one of these grids will be iterated over since there are only finitely many grids. □

We finally give the modified upper-level bilevel program, given a grid G and a set $\mathcal{U}$ of unavoidable sets of G, we can add unavoidable set cuts to the upper level problem of our bilevel program

$$\begin{aligned}
\min_{x,y,z} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij} & & \\
\text{subject to} \quad & z = 1 & & \qquad (V1)\\
& \sum_{(i,j)\in U} y_{ij} \geq 1, & & \text{for all } U \in \mathcal{U}\\
& y_{ij} \in \{0,1\}, & & \text{for all } i,j \in \{1,\ldots,n\}\\
& (x,z) \in S(y). & &
\end{aligned}$$

We call this the `STANDARD` formulation.

3.6 Coupling Free Bilevel Formulation

We have seen in Section 2.2 that the existence of coupling constraints between the upper and lower level program may lead to the disconnected bilevel feasible region. We also note that the special case of bilevel without coupling constraint has been studied more extensively than ones with coupling constraints. Thus we propose in this section a reformulation of the bilevel model `STANDARD` without the coupling constraint $(V1)$. To do this, we add a penalty term to our upper level objective. Given a grid G, a large constant M and a set $\mathcal{U}$ of unavoidable sets of G, we have the bilevel program:

$$\begin{aligned}
\min_{x,y,z} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij} - Mz & & \\
\text{subject to} \quad & \sum_{(i,j)\in U} y_{ij} \geq 1, & & \text{for all } U \in \mathcal{U}\\
& y_{ij} \in \{0,1\}, & & \text{for all } i,j \in \{1,\ldots,n\}\\
& (x,z) \in S(y). & &
\end{aligned}$$

where $S(y)$ is the set of optimum solutions to the y-parameterized follower problem

$$\begin{aligned}
\min_{x,z} \quad & z & & \\
\text{subject to} \quad & \sum_{k=1}^{n} x_{ijk} + z \geq 1, & & \text{for all } i,j \in \{1,\ldots,n\} \qquad (G'0)\\
& \sum_{j=1}^{n} x_{ijk} \leq 1, & & \text{for all } i,k \in \{1,\ldots,n\} \qquad (G'1)\\
& \sum_{i=1}^{n} x_{ijk} \leq 1, & & \text{for all } j,k \in \{1,\ldots,n\} \qquad (G'2)\\
& \sum_{i=sp-s+1}^{sp}\ \sum_{j=sq-q+1}^{sq} x_{ijk} \leq 1, & & \text{for all } p,q \in \{1,\ldots,s\} \text{ and } k \in \{1,\ldots,n\} \qquad (G'3)\\
& x_{ijG_{ij}} \geq y_{ij}, & & \text{for all } i,j \in \{1,\ldots,n\} \qquad (F'1)\\
& \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijG_{ij}} \leq n^2 - 1 & & \qquad (N1)\\
& x_{ijk}, z \in \{0,1\}, & & \text{for all } i,j,k \in \{1,\ldots,n\}.
\end{aligned}$$

The optimal solution to this bilevel program is equal to the optimal solution of our original bilevel program given M is big enough. It can be shown that $M = n^2$ is large enough. We call this formulation the `COUPLING FREE` formulation.

Chapter 4

Computational Results

In this chapter, we show the applicability of the models we proposed in the previous chapter to solve the minimum sudoku clue (MSC) problem. We select 100 sudoku grids of size 6×6 and 20 sudoku grids of size 9×9 for our MSC instances. Although the sample size is not sufficient to represent the whole set of possible sudoku grids, we note that it is sufficient to spot trends in the performance of various models for the MSC problem. In Section 4.1, we describe the setup of the computational environment in which the experiments are run. Section 4.2 describes the performance of our unavoidable set generation algorithm. We separate this part of the experiment from the others because we see that the unavoidable set generation can be improved somewhat independently from the rest of the procedure (such as by using a pattern matching algorithm like the one proposed by McGuire et al. in [32]). In the rest of the chapter, we use the models proposed in Chapter 3 to solve the MSC instances: First, we run our model on random 6×6 instances to determine the best solver setting for our experiment. Second, we investigate the effect of the number of unavoidable set cuts on solver's performance. Lastly, we investigate the effect of transforming coupling constraints into a penalty term as we do in the `COUPLING FREE` formulation.

4.1 Experimental Setup

We run all our instances on the high-performance computing cluster owned by the Mathematics and Natural Sciences Faculty at the Technicsche Universität Berlin. The cluster runs AlmaLinux 8.6 with an x86-64 bit architecture. All our instances run on a single core of an Intel Xeon E5-2630V4 2.2 GHz. The cores run a single thread and hyper-threading is not activated. The computing time is measured in wall-clock seconds, the time limit for each run is set to 270000 seconds (around 3 days) and the memory limit is set to 16 GB. All the instances described in this chapter can be provided by the author upon request.

The algorithm `GENERATE` is implemented in Python 3.9 and utilizes Gurobi version 9.5.1 [22] with default settings to solve the MILPs. To solve our bilevel experiments, we use the bilevel solver by Sinnl and Fischetti in [19]. License for the solver is provided upon request to the author. The solver is distributed as binary compiled under Ubuntu 14.04 64bit with g++ 4.8.4 and uses CPLEX [25] version 12.7 to solve MILP.

For our 6×6 experiment, we generate 100 random instances. To generate these instances, we repeatedly solve the sudoku ILP on a blank puzzle and add no good cut constraints between solves to prevent the same sudoku grid from being generated twice. We also ensure that the grids generated are not mere relabels of one another by bringing them into a normal form where the first row is $1, \ldots, 6$ and generating additional ones if necessary. We randomly select 100 grids out of the total 5000 generated.

The 9×9 instances consist of 20 experiments that are split into two groups of ten. The first group of 10 sudoku grids — instance 0–9 — are randomly picked from Gordon Royle's list of sudoku puzzles with 17 clues [37]. The second group of 10 sudoku grids — instance 30–39 — are randomly picked from the list of sudoku puzzles with a difficulty rating of more than 11 (with the maximum difficulty rating being 12) maintained by the new sudoku players forum[1]. All of the instances in this second group contain more than 20 clues each. Before selecting the instances, we convert the grids to minlex form, an invariant for equivalent

[1]http://forum.enjoysudoku.com/the-hardest-sudokus-new-thread-t6539-600.htmlp277835

		# of generated unavoidable sets with size n													
instance	max size	4	5	6	7	8	9	10	11	12	13	14	15	16	> 16
0	17	5	0	17	0	19	7	62	51	182	124	490	584	1541	1918
1	18	7	0	17	0	14	4	57	25	183	113	404	538	1360	2278
2	18	7	0	18	0	16	6	74	44	166	150	442	486	1279	2312
3	17	4	0	11	0	23	8	72	45	201	138	536	665	1690	1607
4	18	13	0	8	0	20	6	65	35	152	127	404	383	1098	2689
5	18	9	0	15	0	20	5	83	39	156	98	440	499	1197	2439
6	18	8	0	15	0	19	9	68	32	174	117	380	450	1283	2445
7	18	11	0	26	0	13	3	48	32	126	72	264	358	905	3142
8	17	2	0	18	0	12	11	79	48	198	121	523	599	1607	1782
9	18	6	0	18	0	27	4	73	40	186	133	419	487	1319	2288
30	18	7	0	12	0	15	8	66	39	179	129	403	511	1369	2262
31	18	15	0	14	0	26	7	56	21	97	95	306	288	833	3242
32	18	11	0	25	0	21	5	43	18	115	94	257	334	820	3257
33	18	6	0	21	0	23	6	77	41	164	104	465	471	1272	2350
34	18	6	0	18	0	11	12	66	31	182	104	435	535	1458	2142
35	18	10	0	11	0	18	10	75	40	135	108	446	503	1281	2363
36	18	7	0	18	0	16	9	66	33	153	123	464	478	1336	2297
37	18	7	0	15	0	23	6	58	45	170	127	430	497	1321	2301
38	18	13	0	16	0	12	6	59	26	105	86	395	370	979	2933
39	18	18	0	10	0	13	6	51	24	143	68	303	276	790	3298

Table 4.1: Size Of Generated Unavoidable Sets

sudoku, and select ones with different minlex forms to get as diverse of a set as possible. To convert the sudoku grids to minlex form, we use the code by Michael Deverin[2].

Throughout this section, we will use *performance profiles* introduced by Dolan and Moré in [15] to compare the performance of different methods. Given a set of instances $\mathcal{P}$ and a set of methods $\mathcal{S}$, for each instance $p \in \mathcal{P}$ and method $s \in \mathcal{S}$ we define

$$t_{p,s} := \text{time required by method } s \text{ to solve instance } p$$

or $t_{p,s} :=$ time limit if method s cannot solve instance p. Furthermore, we also define the *performance ratio*

$$r_{p,s} := \frac{t_{p,s}}{\min_{\tilde{s} \in \mathcal{S}} t_{p,\tilde{s}}},$$

the ratio between the time required by method s to solve instance p and the minimum time required by any method to solve p. If a method s does not solve an instance p, we set $r_{p,s} = r_M$ where r_M is a sufficiently large constant such that $r_M > r_{p,s}$ for all $p \in \mathcal{P}$ and $s \in \mathcal{S}$. Dolan and Moré elaborated in [15] that the choice of r_M does not affect the explanatory capabilities of the performance profile. Over this chapter, we will be interested in the cumulative distribution function of the performance ratio, that is,

$$\rho_s(\tau) = \frac{1}{|\mathcal{P}|} \big|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}\big|.$$

4.2 Generating Unavoidable Sets

In this subsection, we describe the performance of the procedure `GENERATE`. For each of our 9×9 instances, we generated 5000 minimal unavoidable sets. Table 4.1 shows the size of the generated unavoidable sets along with the size of the largest generated unavoidable set for each instance. In all but three instances, we generated all minimal unavoidable sets of size less than or equal to 17. Note that since we generate unavoidable sets in order of their cardinality, the number shown in the table is equal to the number of minimal unavoidable sets of size n of each instance for all $n \leq 16$. We see that there are no unavoidable sets of sizes 5 and 7. More importantly, we see that the number of minimal unavoidable sets tends to grow exponentially as n increases.

We measured the time it takes to generate the kth unavoidable set by using Gurobi wall-clock time. For a $k \in \mathbb{N}$ the time it takes to generate the kth unavoidable set is the

[2]http://forum.enjoysudoku.com/minlex-form-min-and-max-lists-chaining-t30325-15.html

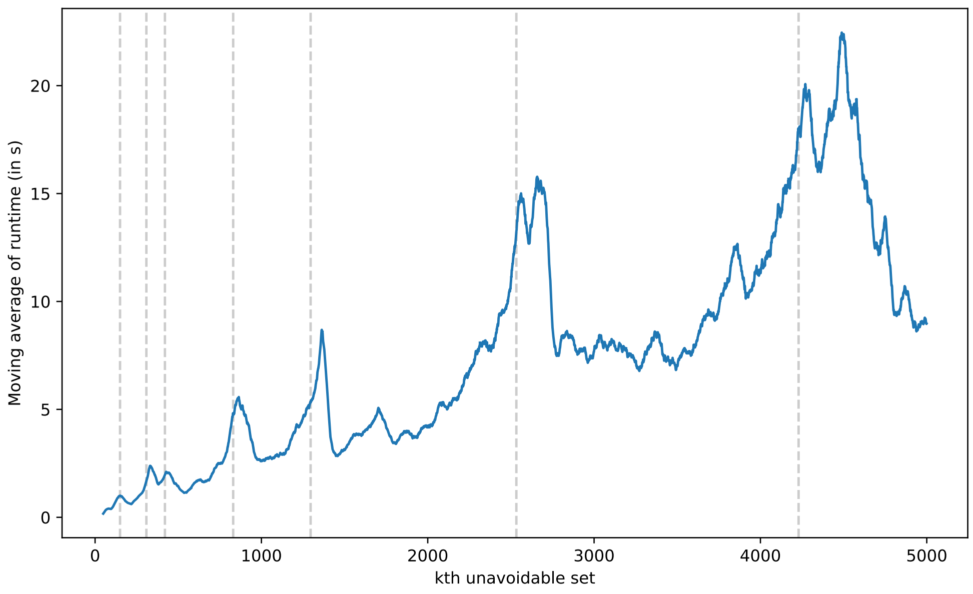


Figure 4.1: Time needed to generate the kth unavoidable in moving average of 50 cut

sum of the wall-clock time of the optimisation which generates the kth unavoidable set and the wall-clock time of any failed optimisation that happens between the optimisation which generates the $(k-1)$th unavoidable set and the optimisation which generates the kth unavoidable set. Rather than doing an analysis of each individual instance, we take the geometric mean of the time needed to generate the kth unavoidable set over all 20 of our instances. To help in identifying the major trends, we plot a moving geometric average over 50 unavoidable sets. The resulting plot is shown in Figure 4.1. We see that generally, the time needed to generate an unavoidable set increases as k gets larger. However, the most interesting part of the plot is the bumps in the moving average. Looking deeper into the result of individual instances, we see that as we try to enumerate all minimal unavoidable sets of size $n \in \mathbb{N}$ the time it needs to generate an unavoidable set increases in each iteration. This is expected as in each iteration, there are fewer and fewer minimal unavoidable sets of size n available, and thus, they become increasingly harder to find. To visualize this effect, we computed the average number of unavoidable set less than or equal to n for $n = 11, \ldots, 17$ (for the instance in which not all unavoidable sets of size 17 have been found, we assume the number of unavoidable set of size less than 17 to be 5000) and drew them as vertical lines in Figure 4.1. One can think of these lines as the average point where instances switch from searching for unavoidable sets of size n to unavoidable sets of size $n+1$. The leftmost line represents $n = 11$. We see that the vertical lines approximately coincide with the bumps in the moving average, showing this effect.

A detail that is not so well illustrated by Figure 4.1 is how extreme these bumps can be. To see this effect, we provide the frequency distribution table of the generation time of unavoidable sets in Table 4.2. Though the majority of the minimally unavoidable sets (93.72%) can be generated in less than 1 minute, some minimal unavoidable set are very hard to find with the longest taking nearly 3 hours to find.

Lastly, it is important to remember that we are not obliged to generate all unavoidable sets. Their only function is to help us reduce the feasible region of our bilevel program and speed up our optimization. To put it slightly informally, we are 'investing' time in exchange for unavoidable sets, which hopefully will help us speed up the optimization process later. For this reason, we find it helpful to plot the average number of cuts generated as a function of time. To do this, for each instance I and each $n = 1, \ldots, 5000$ we calculate the cumulative time our model takes to generate n unavoidable sets of instance I. We then take the geometric mean of the cumulative time for each n over all the instances and plot the result as a function of n. The resulting plot is shown in Figure 4.2. The figure reiterates how generating unavoidable sets is fairly easy at start and shows how it becomes harder as time goes on. It takes less than 20000 seconds to generate the first 2000 unavoidable sets

generation time [s]	# of unavoidable sets
≤ 1	11651
$1 - 10$	54088
$10 - 30$	17250
$30 - 60$	10732
$60 - 300$	5846
$300 - 600$	232
$600 - 1800$	123
$1800 - 3600$	39
$3600 - 7200$	25
≥ 7200	14

Table 4.2: Frequency distribution table of the time needed to generate unavoidable set

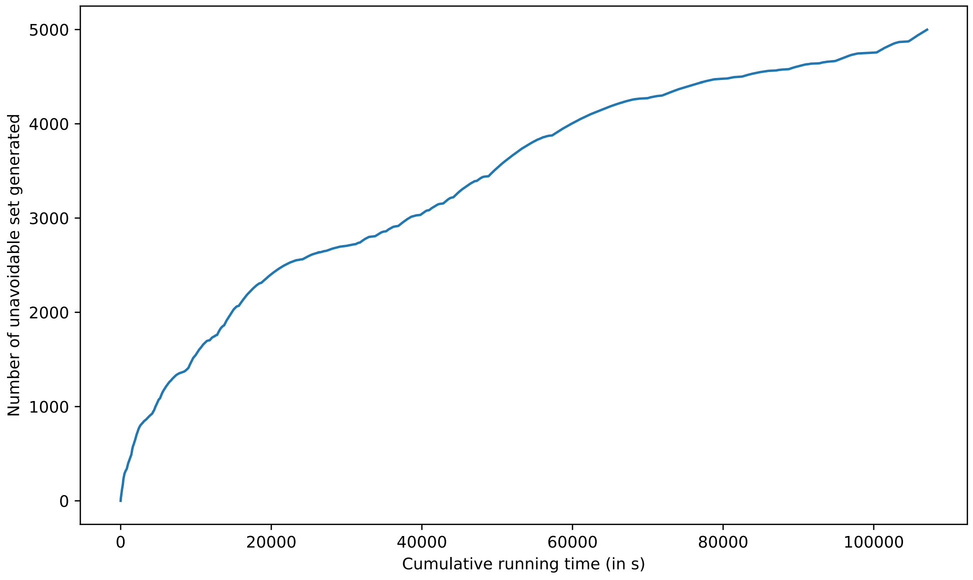


Figure 4.2: The number of unavoidable set generated as a function of time

and nearly 40000 seconds to generate the next 2000.

4.3 Comparing Solving Methods

In this section, we compare the performance of different solver settings on our set of 100 instances of size 6×6. We use the `COUPLING FREE` formulation along with the first 10 unavoidable set cut generated by our unavoidable set generator. Models for the experiments in this section can be found in Appendix B. We consider the following separation methods.

1. Intersection cut with separation using `SEP1` formulation as proposed by Fischetti et al. in [20]. We call this the `SEP1` method.
2. Intersection cut with separation using `SEP2` formulation as proposed by Fischetti et al. in [20]. We call this the `SEP2` method.
3. A combination of intersection cut with separation using `SEP2` formulation and intersection cut with separation using the bilevel free polyhedron proposed by Xu in [44]. This approach was proposed by Fischetti et al. in [19]. We call this the `MIX` method.

We do not consider intersection cuts with separation purely by using the bilevel free polyhedron proposed by Xu in [44] as it fails for to converge on all of the first 10 instances. We use follower upper-bound cuts and activate follower preprocessing as described by Fischetti

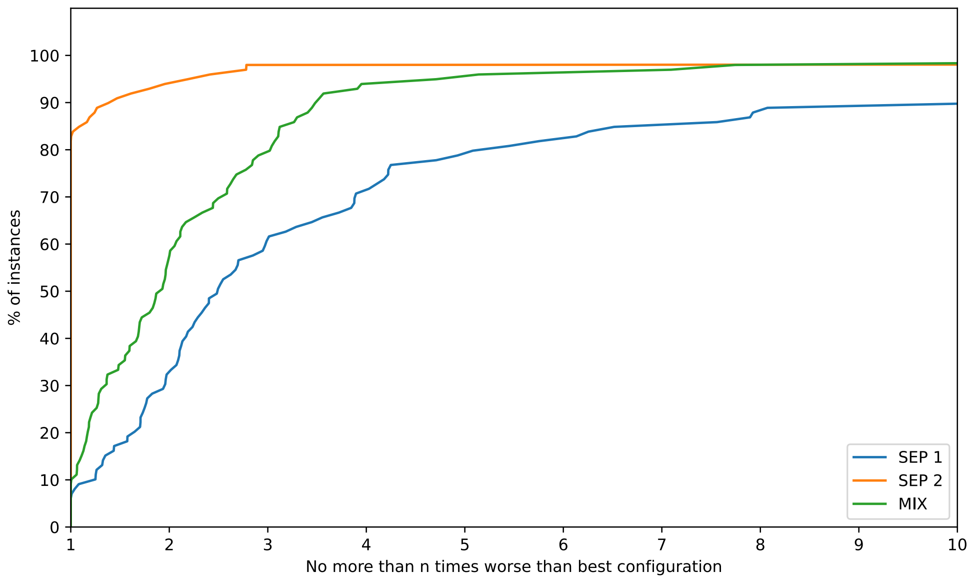


Figure 4.3: Performance profile of different bilevel solving method on 6 × 6 instances

et al. in [19]. The full result of the experiment is given in appendix Table A.1. We plot the performance profiles of the different separation methods. The resulting plot is shown in Figure 4.3. The figure shows the clear superiority of `SEP 2` over the two other methods. On over 80% of the instances, `SEP2` method performed the best, followed by `MIX` with 10%. None of the separation methods give the correct answer on all instances, `SEP1` and `SEP2` return feasible but non-optimal solution on 2 instances and `MIX` returns feasible but non-optimal solution on 1 instance. Although `MIX` solves more instances correctly, it solves them significantly slower than `SEP2`, with only 50% of instances `MIX` solving within less than 2 times worse than the best configuration. In comparison, the `SEP2` setting performed within less than 2 times worse than the best configuration more than 90% of the time. For the rest of the experiment, we thus will use the `SEP2` method for solving our bilevel instances.

4.4 Effect Of Unavoidable Set Cuts

We now see the effect of adding unavoidable set cuts to our model. For this experiment, we use the `STANDARD` bilevel formulation and vary the number of unavoidable sets that are used. For our initial analysis, we assume apriori that the unavoidable sets are already given, that is, we do not take into account the time we need to generate the unavoidable sets. We decide to use for our experiments 500, 1000, 3000, and 5000 unavoidable set cuts respectively, where for an experiment with n unavoidable set cuts, we use the first n cuts generated by our unavoidable set generating algorithm. A summary of the optimization results is shown in Table 4.3. The complete result is provided in Appendix A. 19 out of the 20 instances of size 9 × 9 solved to optimality on at least one solver setting.

	# of instances	
# of unavoidable set	optimal	time limit
500	11	9
1000	16	4
3000	15	5
5000	13	7

Table 4.3: Summary of end result for 9 × 9 standard bilevel model with different number of unavoidable set cuts

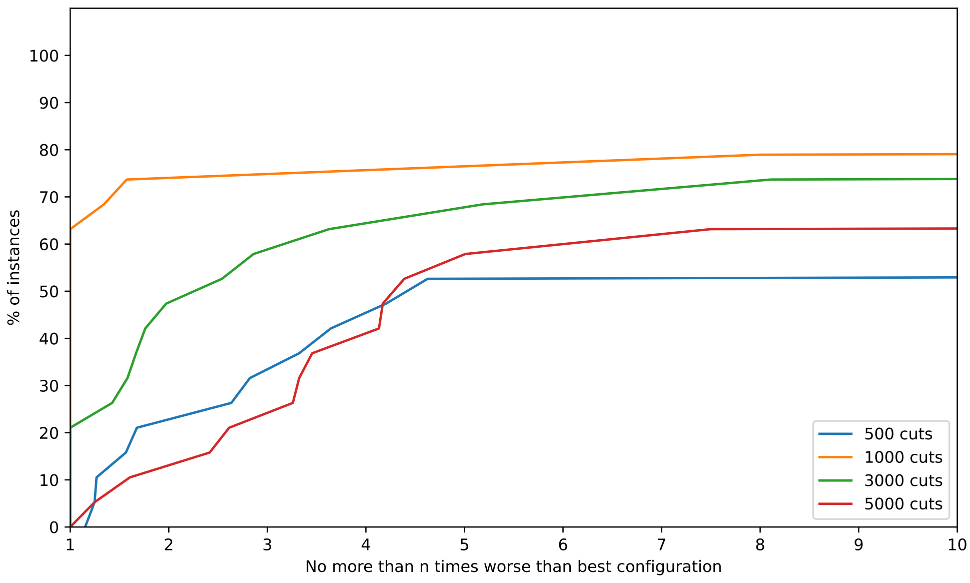


Figure 4.4: Performance profile of different number of unavoidable set on 9×9 instances

As before, we plot the resulting performance profile. The result is shown in Figure 4.4. The most interesting feature of the plot is that adding more unavoidable set cuts does not necessarily result in faster solves. Over 60% of the instances solve fastest on models that use 1000 unavoidable sets, followed by slightly over 20% of instance that solve fastest on models that use 3000 unavoidable sets. To see the reason why the 'extreme' alternatives tend to perform worse, for each number of cuts, we take the geometric average node count over all instances and the geometric average time per node, which we get by dividing the wall-clock time by the number of nodes. The result is shown in Table 4.4

# of unavoidable set	geometric average of node count	geometric average time per node [s]
500	2984487.49	0.038
1000	1933515.13	0.031
3000	1278961.24	0.077
5000	896993.49	0.145

Table 4.4: Average node count and time per node of the different settings

We see that instances with 500 cuts still have too big of a feasible set for the upper-level program and thus resulting in more nodes to check. On the other hand, instances with 5000 cut have a much smaller feasible set for the upper-level program, yet finding a feasible solution is harder because of the larger amount of constraints resulting in a higher average time per node. The trade-offs and both sides lead to the optimal number of cuts being somewhere in the middle. We note that the average time per node of models with 500 is slightly higher 1000, which is counter-intuitive. However, we think this difference is insignificant since the difference is very slight compared to the difference between models with 1000 and 3000 cuts and models with 3000 to 5000 cuts. Moreover, each node is solved very fast, and these slight differences may also be attributed to measurement errors.

Finally, we take into account the time needed to generate the cuts. Note that we only need to compare models with 500 and 1000, as it is clear already that models with 1000 cuts perform better than models with 3000 and 5000 cuts. To compare if it is worth generating the 500 extra unavoidable sets, we compute the augmented time, that is, the time needed to solve the bilevel instance plus the time needed to generate the unavoidable sets, and plot the performance profile for instances with 500 and 1000 cuts. The calculation is done post-hoc using the data from the experiment described in Section 4.2. The resulting plot is shown in 4.5. We see that even when taking the time it takes to generate the unavoidable sets, the

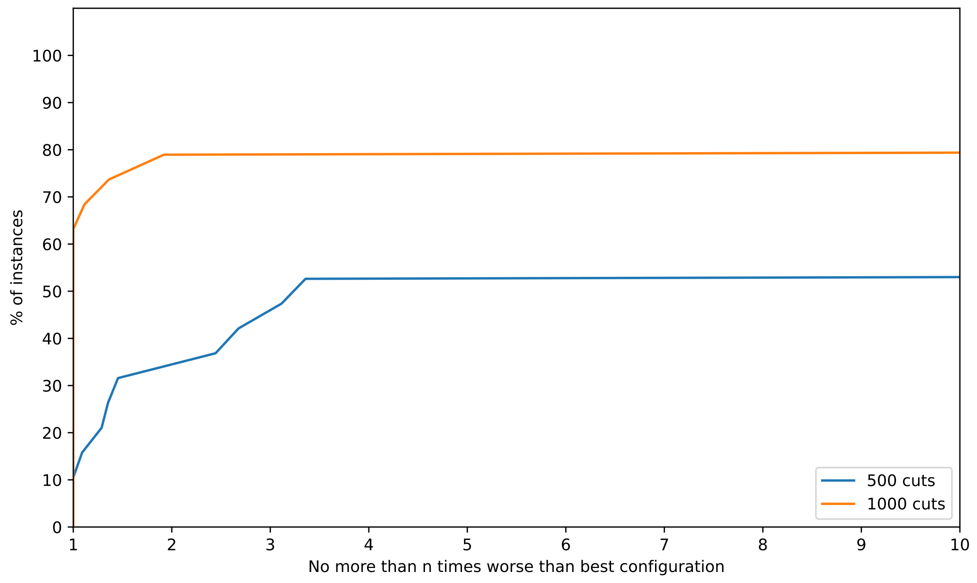


Figure 4.5: Performance profile of different number of unavoidable set on 9×9 instances with augmented time

option of using 1000 cuts is still superior to using 500 cuts.

4.5 Effects Of Coupling Constraints

Finaly, we discuss the effect of removing the coupling constraint in our bilevel through the use of penalty terms. We investigate the case of 1000 and 3000 unavoidable set cuts. Table 4.5 shows a summary of the experimental results comparing the coupling free and standard formulation. The average time is the geometric average of runtime over all instance which solve to optimality and the final gap is the geometric average of final gap over all instances which do not solve to optimality. Since the two formulation have different objective functions, we add the value of the best solution and lower bound of instances which uses the `COUPLING FREE` formulation for solving by M and recalculate the gap. As before, we also plot the performance profile of each method, the resulting plot is shown in Figure 4.6. We see that the `STANDARD` formulation performs slightly faster than the `COUPLING FREE` formulation, which is seen by the lower average time and the performance profile figure. Although slightly slower, instances which are solved using the `COUPLING FREE` formulation are had a slightly lower average final gap.

		# of instances			
# of unavoidable set	formulation	optimal	time limit	average time (in s)	average final gap(%)
1000	standard	16	4	41671.09	10.05
3000	standard	15	5	70496.14	12.30
1000	coupling free	17	3	55347.35	8.56
3000	coupling free	15	5	79161.84	11.46

Table 4.5: Comparison of coupling free formulation and standard formulation

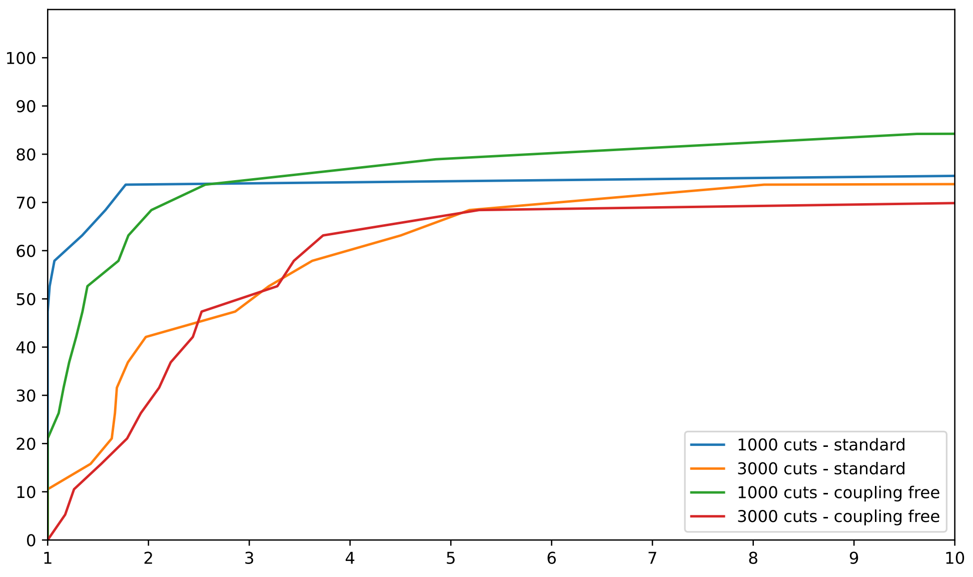


Figure 4.6: Performance profile of coupling free formulation vs standard formulation on 9×9 instances

Conclusion

Throughout this thesis, we have shown that the minimum sudoku clue problem can be solved by formulating it as an integer bilevel linear programming problem. To solve the integer bilevel linear program instances, we introduced a branch and bound framework along with a method to generate intersection cuts to speed up the branch and bound process. Additionally, we also introduced the problem-specific unavoidable set cuts. Finally, we demonstrated the viability of our proposal by providing computational results. In particular, we demonstrated how slight tweaks in the formulation, such as a change in the number of unavoidable set cuts and removal of coupling constraints, affect the performance of the whole model. We saw that using a moderate amount of unavoidable set cuts (1000 and 3000 cuts) performs better than extreme amounts (500 and 5000 cuts), and that the `COUPLING FREE` formulation performed worse than the `STANDARD` formulation in exchange for a slightly better final gap.

We note that despite the viability of our model to solve minimum sudoku clue instances, our proposal is not efficient enough to scale. A large bottleneck factor is the inherent difficulty of solving bilevel optimization problems. Thus, the proposal in this thesis will greatly benefit from improvements in the field of bilevel optimization in general. Apart from improvements in the general bilevel solver used, we see three areas that can still be improved: First, we can use faster unavoidable set finding algorithms such as the one proposed by MacGuire et al. [32]. Second, we can make formulations that exploit the symmetric property of sudoku grids. Third, we can change the approach and use unavoidable set cuts to cut non-feasible solutions throughout the branch and bound process instead of applying a large number of cuts at the root node initially.

On a broader outlook, this thesis adds to the huge amount of literature on using generic mathematical solvers to solve puzzles and games. We highlight two larger ideas that this thesis exemplifies: Firstly, how bilevel models can be used to generate puzzles with certain desirable properties, such as puzzles with unique solutions. Secondly, how bilevel optimization can be a *meet in the middle* approach for optimization problems in which a large amount of constraints are given implicitly, such as unavoidable set cuts in the MSC case.

Bibliography

[1] T. Achterberg, *Constraint Integer Programming*, PhD thesis, Technische Universität Berlin, Berlin, 2007.

[2] K. Appel and W. Haken, *Every planar map is four colorable. Part I: Discharging*, Illinois Journal of Mathematics, 21 (1977), pp. 429 – 490.

[3] E. Balas, *Intersection cuts—a new type of cutting planes for integer programming*, Operations Research, 19 (1971), pp. 19–39.

[4] D. Bertsimas, *Introduction to linear optimization*, Optimization and Neural Computation Series, Athena Scientific, Nashua, NH, 1997.

[5] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig, *The SCIP Optimization Suite 8.0*, technical report, Optimization Online, December 2021.

[6] R. G. Bland, *New finite pivoting rules for the simplex method*, Mathematics of operations Research, 2 (1977), pp. 103–107.

[7] C. Carathéodory, *Über den variabilitätsbereich der fourier'schen konstanten von positiven harmonischen funktionen*, Rendiconti del Circolo Matematico di Palermo (1884-1940), 32 (1911), pp. 193–217.

[8] M. Conforti, G. Zambelli, and G. P. Cornuejols, *Integer Programming*, Graduate texts in mathematics, Springer International Publishing, Cham, Switzerland, 2014.

[9] J. Cooper and A. Kirkpatrick, *Critical sets for Sudoku and general graph colorings*, Discrete Mathematics, 315-316 (2014), pp. 112–119.

[10] G. B. Dantzig, *Maximization of a linear function of variables subject to linear inequalities*, Activity analysis of production and allocation, 13 (1951), pp. 339–347.

[11] G. B. Dantzig and M. N. Thapa, *Linear Programming 1: Introduction*, Springer Series in Operations Research and Financial Engineering, Springer New York, New York, NY, 1997, ch. The Linear Programming Problem, pp. 1–25.

[12] J.-P. Delahaye, *The science behind sudoku*, Scientific American, 294 (2006), pp. 80–7.

[13] S. T. DeNegre, *Interdiction and Discrete Bilevel Linear Programming*, PhD thesis, Lehigh University, 2011.

[14] S. T. DeNegre and T. K. Ralphs, *A branch-and-cut algorithm for integer bilevel linear programs*, in Operations Research and Cyber-Infrastructure, J. W. Chinneck, B. Kristjansson, and M. J. Saltzman, eds., Boston, MA, 2009, Springer US, pp. 65–78.

[15] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[16] I. Ekeland and R. Témam, *Convex Analysis and Variational Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[17] G. Farkas, *A fourier-féle mechanikai elv alkalmazásai*, Mathematikai és Természettudományi Ertesito, 12 (1894), pp. 457–472.

[18] B. Felgenhauer and F. Jarvis, *Mathematics of sudoku I*, Mathematical Spectrum, 39 (2006), pp. 15–22.

[19] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl, *A new general-purpose algorithm for mixed-integer bilevel linear programs*, Operations Research, 65 (2017), pp. 1615–1637.

[20] ———, *On the use of intersection cuts for bilevel optimization*, Mathematical Programming, 172 (2017), pp. 77–103.

[21] J. Fourier, *Solution d'une question particullière du calcul des inégalitiés*, Nouveau Bulletin des Sciences par la Socété Philomatique de Paris, (1826), pp. 317–19.

[22] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2022.

[23] J.-B. Hiriart-Urruty and C. Lemarechal, *Fundamentals of convex analysis*, Grundlehren Text Editions, Springer, Berlin, Germany, 1st ed., 2001.

[24] T. Hürlimann, *Puzzles and Games: A Mathematical Modeling Approach*, Department of Informatics, University of Fribourg, Fribourg, Switzerland, 4th ed., 2016.

[25] IBM, *IBM ILOG CPLEX 12.8 User's Manual*, IBM ILOG CPLEX Division, Incline Village, NV, 2017.

[26] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations, (1972), pp. 85–103.

[27] T. Kleinert, M. Labbé, I. Ljubić, and M. Schmidt, *A Survey on Mixed-Integer Programming Techniques in Bilevel Optimization*, EURO Journal on Computational Optimization, 9 (2021), p. 100007.

[28] T. Koch, *Rapid Mathematical Programming or How to Solve Sudoku Puzzles in a Few Seconds*, in Operations Research Proceedings 2005, H.-D. Haasis, H. Kopfer, and J. Schönberger, eds., Berlin, Heidelberg, 2006, Springer Berlin Heidelberg, pp. 21–26.

[29] B. Korte and J. Vygen, *Combinatorial optimization*, Algorithms and Combinatorics, Springer, Berlin, Germany, 5 ed., Jan. 2012.

[30] M. Köppe, M. Queyranne, and C. T. Ryan, *Parametric Integer Programming Algorithm for Bilevel Mixed Integer Programs*, Journal of Optimization Theory and Applications, 146 (2010), pp. 137–150.

[31] S. Lang, *Linear Algebra*, Undergraduate Texts in Mathematics, Springer New York, New York, NY, 3rd ed., 1987.

[32] G. McGuire, B. Tugemann, and G. Civario, *There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem via hitting set enumeration*, Experimental Mathematics, 23 (2014), pp. 190–217.

[33] A. G. Mersha and S. Dempe, *Linear bilevel programming with upper level constraints depending on the lower level solution*, Applied Mathematics and Computation, 180 (2006), pp. 247–254.

[34] J. T. Moore and J. F. Bard, *The mixed integer linear bilevel programming problem*, Operations Research, 38 (1990), pp. 911–921.

[35] P. Norvig, *Solving Every Sudoku Puzzle.* https://norvig.com/sudoku.html.

[36] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization*, Dover Books on Computer Science, Dover Publications, Mineola, NY, 1998.

[37] G. Royle, *Minimum Sudoku.* https://web.archive.org/web/20060220092603/http://www.csse.uwa.edu.au/∼gordon/sudokumin.php.

[38] E. Russell and F. Jarvis, *Mathematics of sudoku ii*, Mathematical Spectrum, 39 (2006), pp. 54–58.

[39] A. Schrijver, *Theory of linear and integer programming*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, Chichester, England, Nov. 1986.

[40] G. Strang, *Linear Algebra and Its Applications*, Brooks/Cole, Belmont, CA, 4th ed., 2005.

[41] R. J. Vanderbei, *Linear programming*, International Series in Operations Research Management Science, Springer, New York, NY, 4th ed., 2013.

[42] L. Vicente, G. Savard, and J. Judice, *Discrete linear bilevel programming problem*, Journal of Optimization Theory and Applications, 89 (1996), pp. 597–614.

[43] H. Von Stackelberg, *Marktform und gleichgewicht*, J. springer, 1934.

[44] P. Xu, *Three essays on bilevel optimization algorithms and applications*, PhD thesis, Iowa State University, Ames, Iowa, 2012.

[45] P. Xu and L. Wang, *An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions*, Computers & Operations Research, 41 (2014), pp. 309–318.

[46] T. Yato and T. Seta, *Complexity and completeness of finding another solution and its application to puzzles*, IEICE TRANSACTIONS on Fundamentals of Electronics, 86 (2003), pp. 1052–1060.

Appendix A

Tables of Experimental Results

A.1 6x6 Grid

			SEP1			SEP2			MIX	
instance	opt	%gap	time [s]	nodes	%gap	time [s]	nodes	%gap	time [s]	nodes
0	-27	0.00	36.02	3144	0.00	0.44	2	0.00	0.23	2
1	-28	0.00	14.00	1195	0.00	1.36	19	0.00	10.52	293
2	-28	0.00	13.84	1316	0.00	10.97	631	0.00	27.40	1072
3	-28	0.00	40.51	3954	0.00	18.99	990	0.00	26.07	724
4	-28	0.00	76.41	6750	0.00	29.13	1259	0.00	36.77	1010
5	-28	0.00	73.70	6696	0.00	43.68	1598	0.00	40.35	876
6	-28	0.00	62.40	5373	0.00	31.77	1216	0.00	43.34	937
7	-27	0.00	72.84	6488	0.00	18.70	693	0.00	31.45	750
8	-28	0.00	65.61	5706	0.00	16.92	488	0.00	66.83	1654
9	-27	0.00	2.03	197	0.00	0.31	9	0.00	0.25	3
10	-28	0.00	32.43	2898	0.00	7.68	385	0.00	11.93	315
11	-28	0.00	42.20	4048	0.00	15.79	797	0.00	56.30	1185
12	-28	0.00	32.13	3369	0.00	14.24	485	0.00	40.56	856
13	-28	0.00	74.15	6625	0.00	35.49	1515	0.00	41.30	897
14	-28	0.00	31.04	3204	0.00	18.84	925	0.00	24.13	563
15	-28	0.00	21.45	2049	0.00	8.93	449	0.00	31.07	1012
16	-28	0.00	12.53	1133	0.00	2.66	100	0.00	3.12	85
17	-28	0.00	35.69	3411	0.00	18.43	998	0.00	55.66	1120
18	-28	0.00	34.88	3219	0.00	11.58	592	0.00	13.39	343
19	-28	0.00	42.58	4195	0.00	11.07	424	0.00	24.95	477
20	-27	0.00	13.10	1317	0.00	0.08	0	0.00	0.09	0
21	-28	0.00	15.93	1682	0.00	18.54	1034	0.00	33.65	851
22	-28	0.00	7.05	698	0.00	3.15	158	0.00	7.36	245
23	-28	0.00	37.18	3264	0.00	12.61	550	0.00	41.19	694
24	-28	0.00	29.25	2774	0.00	21.65	861	0.00	27.81	531
25	-28	0.00	82.40	7481	0.00	32.36	1418	0.00	54.67	1034
26	-28	0.00	73.01	6774	0.00	24.58	1062	0.00	49.38	1262
27	-28	0.00	13.68	1171	0.00	6.08	354	0.00	4.80	92
28	-28	0.00	31.92	3046	0.00	15.06	752	0.00	39.45	838
29	-28	0.00	33.91	3478	0.00	15.49	780	0.00	31.81	681
30	-28	0.00	22.05	2327	0.00	21.84	1097	0.00	33.83	820
31	-27	0.00	24.00	2296	0.00	0.31	10	0.00	0.41	7
32	-28	0.00	32.43	3373	0.00	13.06	736	0.00	27.09	732
33	-28	0.00	50.40	4630	0.00	18.70	705	0.00	31.07	706
34	-28	0.00	45.49	4490	0.00	45.66	2241	0.00	48.41	1077
35	-28	0.00	49.12	4849	0.00	24.90	1200	0.00	70.70	1803
36	-28	0.00	7.38	696	0.00	6.83	323	0.00	21.32	700
37	-28	0.00	19.65	1902	0.00	4.89	268	0.00	1.76	29
38	-28	0.00	57.20	5659	0.00	27.22	1204	0.00	50.84	1226
39	-28	0.00	32.40	3113	0.00	25.91	1309	0.00	47.89	1199
40	-28	0.00	51.54	5153	0.00	38.85	1942	0.00	53.00	1345
41	-28	0.00	26.76	2679	0.00	13.66	511	0.00	26.84	787
42	-28	0.00	8.70	804	0.00	4.13	195	0.00	8.73	260
43	-28	0.00	9.77	988	0.00	1.24	39	0.00	2.45	54
44	-28	0.00	45.44	4651	0.00	26.60	1294	0.00	103.98	2595
45	-28	0.00	35.00	3606	0.00	22.25	1214	0.00	78.38	1493
46	-27	0.00	0.09	0	0.00	0.04	0	0.00	0.05	0
47	-28	0.00	16.04	1273	0.00	3.98	218	0.00	13.13	449
48	-28	0.00	75.39	6845	0.00	21.88	1172	0.00	37.64	950
49	-28	0.00	45.46	4522	0.00	34.47	1420	0.00	62.01	1552
50	-28	0.00	7.54	791	0.00	2.03	57	0.00	6.90	182

51	-28	0.00	36.02	3044	0.00	8.78	417	0.00	26.95	672
52	-28	0.00	14.70	1368	0.00	2.40	99	0.00	17.00	607
53	-28	0.00	34.85	2913	0.00	6.86	246	0.00	16.77	367
54	-27	0.00	1.00	77	0.00	0.16	0	0.00	0.17	0
55	-28	0.00	53.90	4757	0.00	31.64	1376	0.00	62.95	1408
56	-28	0.00	35.59	3228	0.00	14.27	631	0.00	49.19	1220
57	-28	0.00	15.16	1266	0.00	23.48	1176	0.00	14.58	377
58	-28	0.00	18.72	1666	0.00	11.88	591	0.00	32.99	1060
59	-27	0.00	43.16	4461	0.00	13.57	947	0.00	39.40	1202
60	-28	0.00	40.18	3744	0.00	32.06	1533	0.00	32.23	726
61	-28	0.00	44.42	3569	0.00	18.88	960	0.00	10.52	300
62	-27	0.00	0.03	0	0.00	0.03	0	0.00	0.04	0
63	-28	0.00	40.22	3625	0.00	17.03	830	0.00	51.77	906
64	-28	0.00	32.59	3228	0.00	2.28	100	0.00	31.90	970
65	-28	0.00	33.83	3657	0.00	23.49	1157	0.00	34.91	606
66	-28	0.00	47.40	4369	0.00	8.69	412	0.00	16.78	406
67	-28	0.00	54.65	4986	0.00	31.58	1377	0.00	58.75	1204
68	-28	0.00	5.61	600	0.00	7.74	287	0.00	10.27	309
69	-28	0.00	38.97	3545	0.00	13.05	558	0.00	31.86	996
70	-28	0.00	38.33	3750	0.00	22.44	980	0.00	44.04	971
71	-28	0.00	19.52	1918	0.00	11.06	479	0.00	11.76	328
72	-28	0.00	8.89	823	0.00	2.57	84	0.00	1.18	20
73	-28	0.00	55.55	4792	0.00	23.13	994	0.00	36.88	726
74	-28	0.00	48.54	4638	0.00	11.61	539	0.00	22.70	564
75	-28	0.00	48.34	4428	0.00	23.96	1038	0.00	64.26	1042
76	-28	0.00	91.03	8390	0.00	36.14	1591	0.00	61.45	1901
77	-28	0.00	8.76	924	0.00	21.14	1140	0.00	22.65	601
78	-28	0.00	47.88	4375	0.00	26.96	1206	0.00	39.92	810
79	-28	0.00	79.11	7432	0.00	24.06	1259	0.00	74.84	1555
80	-28	0.00	25.27	2385	0.00	7.11	341	0.00	8.64	268
81	-28	0.00	21.24	2121	0.00	9.28	504	0.00	24.01	811
82	-28	0.00	64.47	5891	0.00	29.69	1437	0.00	47.44	993
83	-28	0.00	15.24	1576	0.00	5.64	314	0.00	14.94	576
84	-28	0.00	92.19	8138	0.00	23.76	1176	0.00	47.55	1241
85	-28	0.00	27.75	2266	0.00	3.50	123	0.00	18.00	573
86	-28	0.00	10.48	920	0.00	1.61	35	0.00	1.90	25
87	-27	0.00	0.13	3	0.00	0.19	0	0.00	0.16	0
88	-28	0.00	24.30	2370	0.00	4.93	169	0.00	10.51	295
89	-27	0.00	1.84	175	0.00	0.33	7	0.00	0.32	0
90	-28	0.00	39.72	3983	0.00	27.67	1491	0.00	60.00	1379
91	-28	3.70	82.53	6693	3.70	3.99	188	0.00	6.75	147
92	-28	0.00	52.83	4743	0.00	30.20	1363	0.00	58.56	1202
93	-27	0.00	39.82	3308	0.00	0.09	0	0.00	0.10	0
94	-28	0.00	9.76	768	0.00	0.66	7	0.00	3.10	83
95	-28	0.00	85.42	7600	0.00	23.92	775	0.00	20.11	407
96	-28	0.00	26.09	2231	0.00	11.20	581	0.00	13.44	464
97	-27	0.00	0.18	9	0.00	0.50	3	0.00	0.56	4
98	-27	0.00	15.33	1465	0.00	0.15	0	0.00	0.17	0
99	-28	3.70	93.31	6977	0.00	11.63	422	3.70	25.69	525

Table A.1: Results for solving 6 × 6 instances using coupling free formulation, 10 unavoidable cut and different seperation tehniques, optimal value (opt) is the minimum value over all the optimal solutions returned by the three seperation tehniques, nonzero gaps means the solver returns an non optimal solution as an optimal solution. Zero nodes means the model is solved using only prepossessing. The minimum number of clue for each instance is opt plus 36

A.2 9x9 Grid Standard Formulation

Instance	BestSol	LB	%gap	time [s]	nodes
0	17.00	17.00	0.00	184881.75	4922697
1	19.00	16.62	12.55	270000.05	5746923
2	17.00	17.00	0.00	69378.94	1927561
3	18.00	16.31	9.40	270000.03	8104016
4	17.00	17.00	0.00	9871.85	301718
5	17.00	17.00	0.00	22947.72	548237
6	17.00	17.00	0.00	138559.56	3519192
7	17.00	17.00	0.00	9859.71	222863
8	18.00	16.27	9.63	270000.15	7349302
9	17.00	17.00	0.00	25233.01	1230915
30	19.00	16.40	13.67	270000.07	6049168
31	19.00	19.00	0.00	41344.89	1568468
32	19.00	19.00	0.00	195397.29	4043637
33	19.00	17.09	10.07	270000.40	6955426
34	19.00	16.65	12.35	270000.05	7480449
35	19.00	17.08	10.11	270000.07	7268023
36	19.00	16.91	11.00	270000.04	6226133
37	19.00	16.92	10.97	270000.10	6291091
38	19.00	19.00	0.00	252680.48	5945957
39	19.00	19.00	0.00	61478.32	1793686

Table A.2: Results for solving 9×9 instances using standard formulation with 500 unavoidable set cuts

Instance	BestSol	LB	%gap	time [s]	nodes
0	17.00	17.00	0.00	55644.06	1619160
1	17.00	17.00	0.00	123869.51	3725900
2	17.00	17.00	0.00	131983.50	3906065
3	17.00	17.00	0.00	223561.98	7397955
4	17.00	17.00	0.00	8574.81	235709
5	17.00	17.00	0.00	28496.09	869549
6	17.00	17.00	0.00	38024.60	1241642
7	17.00	17.00	0.00	2131.18	52748
8	17.00	17.00	0.00	76565.86	2622712
9	17.00	17.00	0.00	20233.32	690817
30	19.00	16.77	11.73	270000.18	7641399
31	19.00	19.00	0.00	26421.42	1108176
32	19.00	19.00	0.00	74166.50	2434659
33	18.00	18.00	0.00	48643.59	1852269
34	19.00	17.00	10.53	270000.05	7648869
35	19.00	17.34	8.74	270000.04	8891077
36	19.00	17.20	9.45	270000.11	8407526
37	18.00	18.00	0.00	42342.48	1485254
38	19.00	19.00	0.00	89523.14	3183401
39	19.00	19.00	0.00	36674.09	1218309

Table A.3: Results for solving 9×9 instances using standard formulation with 1000 unavoidable set cuts

Instance	BestSol	LB	%gap	time [s]	nodes
0	17.00	17.00	0.00	79326.35	885596
1	17.00	17.00	0.00	92296.77	1084257
2	17.00	17.00	0.00	16501.87	183823
3	19.00	16.16	14.97	270000.04	3816173
4	17.00	17.00	0.00	24527.62	266418
5	17.00	17.00	0.00	18110.93	205722
6	17.00	17.00	0.00	96547.52	1114468
7	17.00	17.00	0.00	17280.69	197104
8	19.00	15.96	16.00	270000.03	3092500
9	17.00	17.00	0.00	104903.85	1279667
30	18.00	18.00	0.00	252184.60	3293895
31	19.00	19.00	0.00	52145.46	916912
32	19.00	19.00	0.00	123675.45	1860318
33	18.00	18.00	0.00	176334.69	2250158
34	18.00	18.00	0.00	214335.39	4345425
35	19.00	16.99	10.56	270000.04	3234488
36	19.00	17.00	10.53	270000.05	3360901
37	19.00	16.99	10.58	270000.04	3698168
38	19.00	19.00	0.00	157674.36	2314770
39	19.00	19.00	0.00	57975.19	835819

Table A.4: Results for solving 9×9 instances using standard formulation with 3000 unavoidable set cuts

Instance	BestSol	LB	%gap	time [s]	nodes
0	17.00	17.00	0.00	89334.78	587431
1	17.00	17.00	0.00	115114.04	745294
2	17.00	17.00	0.00	123691.05	853041
3	19.00	15.72	17.24	270000.03	1717607
4	17.00	17.00	0.00	20704.22	125524
5	17.00	17.00	0.00	74846.62	528577
6	17.00	17.00	0.00	126351.61	954631
7	17.00	17.00	0.00	6944.75	49012
8	18.00	15.78	12.33	270000.03	1753736
9	17.00	17.00	0.00	84395.78	550776
30	19.00	16.11	15.21	270000.03	1732943
31	19.00	19.00	0.00	132341.15	1024163
32	19.00	19.00	0.00	256229.63	1897398
33	18.00	18.00	0.00	213500.30	1454807
34	19.00	16.21	14.66	270000.05	1683150
35	18.00	18.00	0.00	180456.29	1253670
36	19.00	16.59	12.67	270000.04	1784160
37	19.00	16.59	12.69	270000.08	1816081
38	19.00	17.78	6.44	270000.03	2261143
39	19.00	19.00	0.00	95809.29	780504

Table A.5: Results for solving 9×9 instances using standard formulation with 5000 unavoidable set cuts

A.3 9x9 Grid Coupling Free Formulation

instance	BestSol	LB	%gap	time [s]	nodes
0	-64.00	-64.00	0.00	71369.06	2199784
1	-64.00	-64.00	0.00	157111.89	4404113
2	-64.00	-64.00	0.00	5165.66	125424
3	-63.00	-64.32	7.34	270000.04	7786728
4	-64.00	-64.00	0.00	17398.84	494154
5	-64.00	-64.00	0.00	20089.21	451566
6	-64.00	-64.00	0.00	21438.74	519194
7	-64.00	-64.00	0.00	20510.11	453029
8	-63.00	-64.71	9.48	270000.09	7459432
9	-64.00	-64.00	0.00	51863.18	1407757
30	-63.00	-63.00	0.00	154165.49	4702752
31	-62.00	-62.00	0.00	35542.35	1095851
32	-62.00	-62.00	0.00	89967.39	2495785
33	-63.00	-63.00	0.00	87594.07	2743598
34	-63.00	-63.00	0.00	248156.52	6840725
35	-63.00	-63.00	0.00	139945.56	4260294
36	-62.00	-63.71	9.02	270000.17	9129557
37	-63.00	-63.00	0.00	205374.55	6436955
38	-62.00	-62.00	0.00	87798.11	3108216
39	-62.00	-62.00	0.00	34378.14	1060724

Table A.6: Results for solving 9×9 instances using coupling free formulation with 1000 unavoidable set cuts

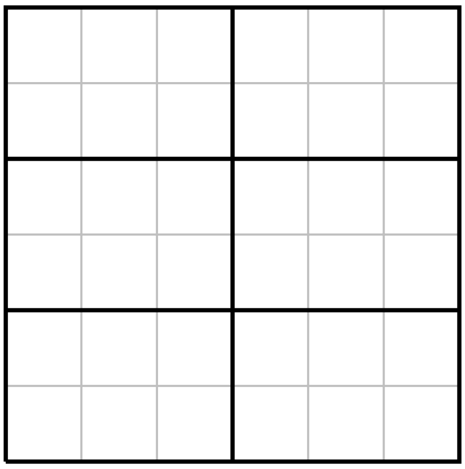
instance	BestSol	LB	%gap	time [s]	nodes
0	-64.00	-64.00	0.00	70183.83	665977
1	-64.00	-64.00	0.00	108317.65	1132657
2	-64.00	-64.00	0.00	116215.44	1369098
3	-62.00	-65.00	15.79	270000.04	2906380
4	-64.00	-64.00	0.00	19041.28	172717
5	-64.00	-64.00	0.00	44195.89	494296
6	-64.00	-64.00	0.00	80015.52	848456
7	-64.00	-64.00	0.00	11253.13	103710
8	-64.00	-64.00	0.00	251040.73	2567400
9	-64.00	-64.00	0.00	69656.04	730164
30	-63.00	-63.00	0.00	236066.31	3019671
31	-62.00	-62.00	0.00	66780.09	967468
32	-62.00	-62.00	0.00	142718.95	1826723
33	-62.00	-63.73	9.13	270000.03	4998743
34	-62.00	-64.34	12.30	270000.03	3581811
35	-63.00	-63.00	0.00	100375.62	1248234
36	-62.00	-64.00	10.53	270000.03	3354429
37	-62.00	-64.01	10.59	270000.03	3438391
38	-62.00	-62.00	0.00	184864.09	2524939
39	-62.00	-62.00	0.00	61470.40	878497

Table A.7: Results for solving 9×9 instances using coupling free formulation with 3000 unavoidable set cuts

Appendix B

Models For 6 by 6 Instances

We present in this section, the models used in our 6×6 experiment. A 6×6 sudoku board has the following form.



B.1 Model GENERATE

$$
\begin{array}{lrll}
\min\limits_{x} & 0^T x & & \\
\text{subject to} & \sum_{k=1}^{6} x_{ijk} = 1, & \text{for all } i, j \in \{1, \dots, 6\} & (G0) \\
 & \sum_{j=1}^{6} x_{ijk} = 1, & \text{for all } i, k \in \{1, \dots, 6\} & (G1) \\
 & \sum_{i=1}^{6} x_{ijk} = 1, & \text{for all } j, k \in \{1, \dots, 6\} & (G2) \\
 & \sum_{i=2p-2+1}^{2p} \sum_{j=3q-3+1}^{3q} x_{ijk} = 1, & \text{for all } p \in \{1, \dots, 3\},\ q \in \{1, 2\} \text{ and } k \in \{1, \dots, 6\} & (G3) \\
 & \sum_{i=1}^{6} \sum_{j=1}^{6} x_{ijG_{ij}} = n^2 - m & & (D1) \\
 & x_{ijk} \in \{0, 1\}, & \text{for all } i, j, k \in \{1, \dots, n\}, &
\end{array}
$$

B.2 Model COUPLING FREE

$$\begin{aligned}
\min_{x,y,z} \quad & \sum_{i=1}^{6}\sum_{j=1}^{6} y_{ij} - Mz \\
\text{subject to} \quad & \sum_{(i,j)\in U} y_{ij} \geq 1, && \text{for all } U \in \mathcal{U} \\
& y_{ij} \in \{0,1\}, && \text{for all } i,j \in \{1,\dots,6\} \\
& (x,z) \in S(y).
\end{aligned}$$

where $S(y)$ is the set of optimum solutions to the y-parameterized follower problem

$$\min_{x,z} \quad z$$

$$\text{subject to} \quad \sum_{k=1}^{6} x_{ijk} + z \geq 1, \quad \text{for all } i,j \in \{1,\dots,6\} \qquad (G'0)$$

$$\sum_{j=1}^{6} x_{ijk} \leq 1, \quad \text{for all } i,k \in \{1,\dots,6\} \qquad (G'1)$$

$$\sum_{i=1}^{6} x_{ijk} \leq 1, \quad \text{for all } j,k \in \{1,\dots,6\} \qquad (G'2)$$

$$\sum_{i=2p-2+1}^{2p} \sum_{j=3q-3+1}^{3q} x_{ijk} \leq 1, \quad \text{for all } p \in \{1,\dots,3\},\ q \in \{1,2\} \text{ and } k \in \{1,\dots,6\} \qquad (G'3)$$

$$x_{ijG_{ij}} \geq y_{ij}, \quad \text{for all } i,j \in \{1,\dots,n\} \qquad (F'1)$$

$$\sum_{i=1}^{6}\sum_{j=1}^{6} x_{ijG_{ij}} \leq 35 \qquad (N1)$$

$$x_{ijk}, z \in \{0,1\}, \quad \text{for all } i,j,k \in \{1,\dots,6\}.$$

Appendix C

Convex Sets

In this section, we briefly introduce key results from convex analysis. For a more extensive discussion on the results covered in this section, we mention Ekeland and Témam [16] and Hiriart-Urruty and Lemaréchal [23], both of which heavily inspired the presentation in this section.

Let u and v be two points in $\mathbb{R}^n$. We define the *line-segment* $[u, v] \subseteq \mathbb{R}^n$ to be

$$[u, v] := \{\lambda u + (1 - \lambda)v \mid 0 \leq \lambda \leq 1\} = \{v + \lambda(u - v) \mid 0 \leq \lambda \leq 1\}.$$

The points u and v are called *endpoints* of the line-segment $[u, v]$. We call a set $C \subseteq \mathbb{R}^n$ *convex*, if for every pair of elements (u, v) of C the line segment connecting the two elements $[u, v]$ is contained in C. An illustration of convex sets is given in Figure C.1. The set A is convex whilist the set B is not.

Given a set of vectors $x_1, \ldots, x_n \in \mathbb{R}^n$, a vector $x^* \in \mathbb{R}^n$ is a *convex combination* of $x_1, \ldots, x_n$ if there exist $\lambda_1, \ldots, \lambda_n \geq 0$ such that

$$x^* = \sum_{i=1}^{n} \lambda_i x_i \quad \text{and} \quad \sum_{i=1}^{n} \lambda_i = 1.$$

Proposition C.1. *A set $C \subseteq \mathbb{R}^n$ is convex, if and only if for every finite subset $\{x_1, \ldots, x_k\}$ of C, the set of all convex combinations of $x_1, \ldots, x_k$ is a subset of C*

Proof. To show sufficiency, note that for every pair of elements $u, v \in C$, the line-segment $[u, v]$ is the set of all convex combinations of u and v. To show the necessity, we use induction over the cardinality of the subsets. For subsets of size two, the results follows directly from the definition of convex sets. Suppose that the theorem is true for all subsets of size k. We consider a subset $\{x_1, \ldots, x_k, x_{k+1}\} \subseteq C$ of size $k + 1$. Let x^* be a convex combination of $x_1, \ldots, x_k, x_{k+1}$ with

$$x^* = \sum_{i=1}^{k+1} \lambda_i x_i, \quad \sum_{i=1}^{k+1} \lambda_i = 1$$

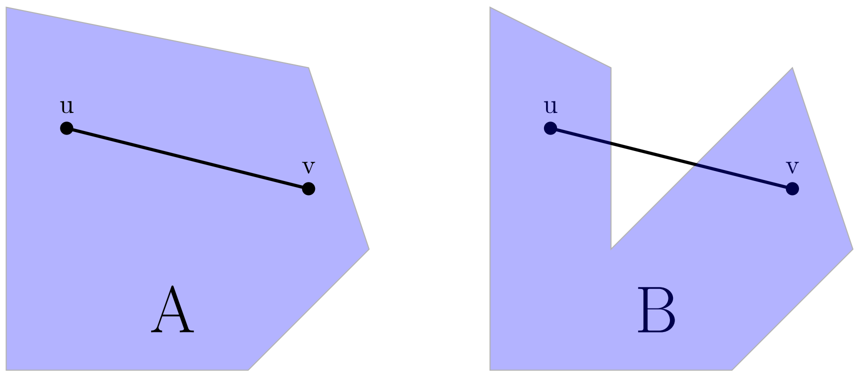


Figure C.1: A Convex Set A and a Non Convex Set B

and $\lambda_1, \ldots, \lambda_k, \lambda_{k+1} \geq 0$. If $\lambda_{k+1} = 0$, we are done because x^* is then a convex combination of $x_1, \ldots, x_k$. Otherwise, we let $z := \sum_{i=1}^{k} \lambda_i = 1 - \lambda_{k+1}$ and

$$\tilde{x} := \sum_{i=1}^{k} \frac{\lambda_i}{z} x_i.$$

By construction is $\tilde{x}$ a convex combination of $x_1, \ldots, x_k$. Thus, $\tilde{x} \in C$ by the induction assumption. Furthermore, we have

$$x^* = \lambda_{k+1} x_{k+1} + z\tilde{x} = \lambda_{k+1} x_{k+1} + (1 - \lambda_{k+1})\tilde{x}.$$

Using the convexity of C it follows $x^* \in C$, which shows the desired result. □

Lemma C.2. *Let $(C_i)_{i \in I}$ be a collection of convex subsets of $\mathbb{R}^n$. The set $\bigcap_{i \in I} C_i$ is convex.*

Proof. Let u and v be two elements of $\bigcap_{i \in I} C_i$. For all $i \in I$, it applies that the line segment $[u, v]$ is in C_i because C_i is convex. Thus, it applies,

$$[u, v] \in \bigcap_{i \in I} C_i$$

which completes the proof. □

For a given set $S \subseteq \mathbb{R}^n$, we define the *convex hull* of S to be the inclusion wise minimal convex set that contains S, that is, the set

$$conv(S) := \bigcap \{C \subseteq \mathbb{R}^n \mid C \text{ is a convex set and } S \subseteq C\}.$$

The following proposition give a characterisation for elements in the convex hull.

Proposition C.3. *Let $S \subseteq \mathbb{R}^n$. It applies*

$$conv(S) = \{x \in \mathbb{R}^n \mid x \textit{ is a convex combiantion of finitely many elements of } S\}.$$

Proof. Note that $conv(S)$ is convex since it is an intersection of convex sets. Let x be a convex combination of $x_1, \ldots, x_k \in S$, then per definition $x_1, \ldots, x_k \in conv(S)$. Thus $x \in conv(S)$ by Proposition C.1. To show the converse direction, we denote

$$T := \{x \in \mathbb{R}^n \mid x \text{ is a convex combiantion of finitely many elements of } S\}$$

It applies $S \subseteq T$ and $T \subseteq C$ for any convex set C with $S \subseteq C$ by Proposition C.1. It remains to show that T is convex. Consider a pair of elements (u, v) of T. By definition of T there exist elements $x_1, \ldots, x_k, y_1, \ldots, y_l$ of S such that

$$u = \sum_{i=1}^{k} \alpha_i x_i \text{ and } v = \sum_{j=1}^{l} \beta_j y_j$$

with $\sum_{i=1}^{k} \alpha_i = \sum_{j=1}^{l} \beta = 1$ and $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_l \geq 0$. For an arbitrary but fixed $0 \leq \lambda \leq 1$, it applies

$$\lambda u + (1 - \lambda) v = \sum_{i=1}^{k} \lambda \alpha_i x_i + \sum_{j=1}^{l} (1 - \lambda) \beta_j y_j.$$

Note that all the coefficients $\lambda \alpha_i$ and $(1 - \lambda)\beta_j$ are non-negative. Furthermore, it applies

$$\sum_{i=1}^{k} \lambda \alpha_i + \sum_{j=1}^{l} (1 - \lambda) \beta_i = \lambda \sum_{i=1}^{k} \alpha_i + (1 - \lambda) \sum_{j=1}^{l} \beta_i = \lambda + (1 - \lambda) = 1.$$

Therefore, $\lambda u + (1 - \lambda) v$ is a convex combination of $x_1, \ldots, x_k, y_1, \ldots, y_l$ and is an element of T. □

Carathéodory [7] showed the sharpened result that any point in the convex hull of a set $S \subseteq \mathbb{R}^n$ can be written as the convex combination of at most $n+1$ elements of the S. Before presenting this result, we first introduce *conic combinations*.

For a set of vectors $x_1, \ldots, x_k \in \mathbb{R}^n$, we call a vector $x^* \in \mathbb{R}^n$ a *conic combination* of $x_1, \ldots, x_k$ if there exist scalars $\lambda_1, \ldots, \lambda_k \geq 0$ such that

$$x^* = \sum_{i=1}^{k} \lambda_i x_i.$$

For a set $S \subseteq \mathbb{R}^n$, we define the *conic hull* of S

$$cone(S) := \{x \in \mathbb{R}^n \mid x \text{ is a conic combination of finitely many elements of } S\}$$

Theorem C.4 (Carathéodory's Theorem [7]). *Let $S \subseteq \mathbb{R}^n$ be a non-empty set.*

(i) For all non-zero vectors $x^* \in cone(S)$, there exists a set of linearly independent vectors $x_1, \ldots, x_d$ of S such that x^* is a conic combination of $x_1, \ldots, x_d$.

(ii) For all $x^* \in conv(S)$, there exists a set of at most $n+1$ elements $x_1, \ldots, x_d$ of S such that x^* is a convex combination of $x_1, \ldots, x_d$.

Proof. (i) Let $x^* \in cone(S)$ be arbitrary but fixed with $x^* = \sum_{i=1}^{k} \lambda_i x_i$, $x_i \in S$ and $\lambda_i \geq 0$ for $i = 1, \ldots k$. If the vectors x_i are linearly independent, then we are done. Otherwise there exist scalars $\beta_1, \ldots, \beta_k$, not all zero, such that

$$\sum_{i=1}^{k} \beta_i x_i = 0.$$

and at least one β_i is strictly positive. We let

$$t^* := \max\{t \geq 0 \mid \lambda_i - t\beta_i \geq 0 \text{ for all } i = 1, \ldots, k\} = \min_{\beta_j > 0} \frac{\lambda_j}{\beta_j}$$

and $\tilde{\lambda}_i := \lambda_i - t^* \beta_i$ for all $i = 1, \ldots, k$. By construction, it applies $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k \geq 0$ and $\tilde{\lambda}_j = 0$ for some $1 \leq j \leq k$. We have

$$\sum_{i=1}^{k} \tilde{\lambda}_i x_i = \sum_{i=1}^{k} (\lambda_i - t^* \beta_i) x_i = \sum_{i=1}^{k} \lambda_i x_i - t^* \sum_{j=1}^{k} \beta_j x_j = \sum_{i=1}^{k} \lambda_i x_i = x^*.$$

which shows that x^* can be written as a conic combination of $k-1$ elements. Because k is finite, we can iterate this process until all vectors x_i are linearly independent.

(ii) We denote with $\bar{S} := \{(s,1) \mid s \in S\} \subseteq \mathbb{R}^{n+1}$. Let $x^* \in conv(S)$ be arbitrary. By Proposition C.3 there exists a set of vectors $x_1, \ldots, x_k \in S$ such that x^* is the convex combination of $x_1, \ldots, x_k$, that is, there exist scalars $\lambda_1, \ldots, \lambda_k \geq 0$ with

$$x^* = \sum_{i=1}^{n} \lambda_i x_i \quad \text{and} \quad \sum_{j=1}^{n} \lambda_j = 1.$$

This is equivalent to

$$\begin{bmatrix} x^* \\ 1 \end{bmatrix} = \lambda_1 \begin{bmatrix} x_1 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} x_2 \\ 1 \end{bmatrix} + \cdots + \lambda_n \begin{bmatrix} x_n \\ 1. \end{bmatrix}$$

which implies that $(x^*, 1) \in cone(\bar{S})$. We get using (i) that $(x^*, 1)$ can be written as a conic combination of linearly independent vectors of $\bar{S}$, that is, there exist a set of linearly independent vectors $(\bar{x}_1, 1), \ldots, (\bar{x}_d, 1) \in \bar{S}$ and scalars $\alpha_1, \ldots, \alpha_d \geq 0$ such that

$$\begin{bmatrix} x^* \\ 1 \end{bmatrix} = \alpha_1 \begin{bmatrix} \bar{x}_1 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} \bar{x}_2 \\ 1 \end{bmatrix} + \cdots + \alpha_d \begin{bmatrix} \bar{x}_d \\ 1 \end{bmatrix}.$$

Note that $d \leq n+1$ because of $(\bar{x}_1, 1), \ldots, (\bar{x}_d, 1)$ are linearly independent and $\bar{x}_i \in S$ for all $i \in \{1, \ldots, d\}$ by definition of $\bar{S}$. Considering the equality row wise yields:

$$x^* = \sum_{i=1}^{d} \alpha_i \bar{x}_i \quad \text{and} \quad \sum_{j=1}^{d} \alpha_j = 1.$$

which shows that x^* is a convex combination of $\bar{x}_1, \ldots, \bar{x}_d$. □

A geometric interpretation of proof of the second statement is that one can think of conic hulls as the intersection between a cone and the plane $z = 1$ as illustrated in Figure C.2

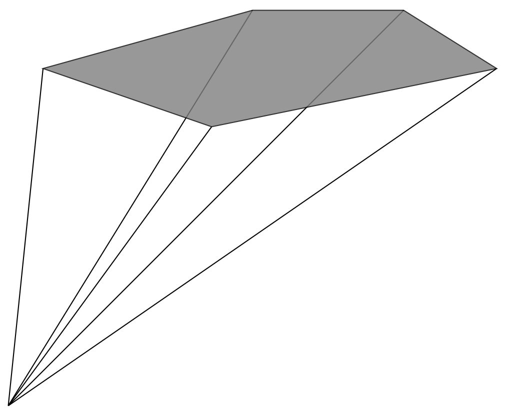


Figure C.2: Illustration For The Proof To Theorem C.4