Université catholique de Louvain
Faculté des Sciences Appliquées

Exploring Structure and Reformulations in Different Integer Programming Algorithms

Thèse présentée en vue de l'obtention du grade de
Docteur en Sciences Appliquées par

Quentin LOUVEAUX

Promoteur:	L.A. Wolsey
Jury:	V. Blondel (Président)
	K. Aardal
	B. Fortz
	Y. Nesterov
	Y. Pochet
	R. Weismantel

Acknowledgements

There are many people I would like to thank who helped me achieve this long-term work.

First of all, I would like to thank my supervisor Laurence Wolsey. He has always been remarkably open and available, spending time reading my drafts and discussing my research. I am impressed by his intuition and knowledge of the field. His guidance has been invaluable. He always knows the right questions to ask to make your work progress. He is not only an excellent researcher but he also takes care of every aspect of a researcher's life. Indeed he gave me many opportunities to meet people in the domain, at CORE and in various conferences and to be able to spend time abroad.

I am also very grateful to Robert Weismantel for giving me the opportunity to join his group in Magdeburg for three months. His enthusiasm and dynamism pushed me forward. He restored my confidence and encouraged me to complete the thesis. I really enjoyed working with him in Magdeburg.

I spent a large amount of time working with Matthias Köppe in Magdeburg. Our work together was very interesting and entertaining. I am impressed by his programming skill and his dedication to his work. I owe a lot to him and Robert for a large part of the work presented here.

During the first four years, I was fortunate to be financed by the Belgian National Fund for Scientific Research (FNRS) as a research fellow. The freedom they give to researchers is of great aid for the quality of their work. I am also in debt to the department of applied mathematics which allowed me to complete the thesis by financing my last year.

I also wish to thank Leen Stougie and Karen Aardal who welcomed me in the Netherlands for five months and gave me the opportunity to meet a lot of people in a pleasant working atmosphere. I appreciate the time that they devoted to me.

In all the places in which I worked during the past five years I met many people that I wish to thank here for their company. First of all, Francisco Ortega has been like a brother to me. I could always rely on him in any situation.

We have also worked on our physical shape together by playing squash and badminton games. Mathieu Van Vyve was a tough tennis opponent and I also appreciated the long philosophical discussions we had together with him and Pancho. CORE would not be the same without Yves Pochet's permanent good mood and kindness. Utz-Uwe Haus was a remarkable host for me in Magdeburg. I enjoyed his guided tours of Germany. In any particular order, I also would like to thank Andrew, Marko, Renaud, Patricio, Hamish, Bram, Helena, Vincent, Yvan, Mady, Christine, Magali, Sébastien, Jean-Marie, Michel, Ruslan, Peter, Francois and Jean-François at CORE, Tjark, Samuel, René, Judith and Rudi in Eindhoven, Dennis and Bianca in Magdeburg. Their company has been particularly enjoyable.

Finally I would like to thank my family who have supported me throughout my thesis. My brothers and parents always kept an interest in what I was doing. My wife Catherine kept me focused when my motivation was at the lowest. Thank you all for your support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Integer programming

What is Integer Programming? Integer programming is a field of applied mathematics that studies optimization problems in which the decision variables must take integer values. There are many applications in which the indivisibility of some variables is inherent to the problem. For example,

- In the facility location problem, one has to decide on a certain number of facilities to open to serve some clients. This number of facilities must be integer.

- In the cutting stock problem, we look for the best way to cut patterns in a roll of material in order to minimize the quantity of wasted material.

- In a similar vein, in the bin packing problem, one wants to minimize the number of bins needed to carry a certain number of indivisible objects.

Problems with on/off or yes/no decisions also belong to the domain of applications of integer programming. We have for example,

- The traveling salesman problem, i.e. finding the smallest tour to visit a given number of cities.

- The network design problem in which we have to decide on arcs to open in a network to allow a certain flow to pass through the network but minimizing the cost of opening the arcs at the same time.

- Integer programming also appears in airplane or train scheduling for example.

Hardness of Integer Programming However solving optimization problems with integer variables turns out to be a difficult task. For most of the problems described above, it is very difficult to find an algorithm that can be guaranteed to find an optimal solution in a reasonable amount of time. In fact, related to the work of Cook [12] and the complexity theory, it has been shown that integer programming in general belongs to a class of problems for which no one has ever found an efficient algorithm working on all instances. It is strongly believed that it is impossible to find an algorithm that solves all the instances of integer programming efficiently (i.e. in what is called polynomial time).

Main ideas to handle integer programs The last paragraph indicated the difficulty of solving integer programs and in particular the fact that finding one good general algorithm is probably impossible. However integer programs do occur in practice and we need to find ways to tackle them. In today's algorithms, we find three basic ingredients that we now discuss.

Relaxation. We assume that integer problems are hard to solve. It is therefore natural to start by trying to solve easier variants of the problem. The easier variant is called a *relaxation*. From the solution of the relaxation, one can deduce some interesting conclusions for the solution of the initial problem.

Example. This is a common situation in physics for example. When a real problem is too hard to handle, one tries to rely on an easier model to obtain information about the original problem: the perfect gas law comes from assumptions made on the gas that simplify the model and make it computable. On the other hand, we know that the assumptions are too strong and the perfect gas model only gives an idea of how a real gas behaves.

The most common relaxation used in integer programming is the linear relaxation, i.e. solving a continuous problem where the integrality requirements on the variables are dropped. Indeed in this thesis, we only look at linear integer problems. In this case, if we relax the integrality requirements, we obtain a linear program for which "efficient" algorithms exist (like the simplex algorithm which has a good running time in practice, or interior point methods which are guaranteed to run in polynomial time). The fact that the new problem is close enough to the original but computable is of course very important. However it is not the only relaxation used in integer programming. Indeed if all the variables of an equality constrained integer program can take any integer value (not restricted to any interval in $\mathbb{Z}$), the integer problem can be solved in polynomial time. Therefore another interesting relaxation is to keep the integrality but drop the bounds on the variables, such as nonnegativity constraints which often occur in practice. We can do this for all the variables or just a fraction of them.

Enumeration. Assuming that the number of possible values of the integer variables is finite, which is almost always the case in practice, we could imag-

ine enumerating all possible cases to find the best feasible one. However the number of cases to test is typically so huge that a pure enumeration approach is hopeless. But since integer problems are hard to solve, a good idea is to decompose the problem into different subproblems of smaller sizes: "divide and conquer". The most natural way to do it is to fix the value of some variable and consider all the possible subproblems arising with each possible value of the variable. By using the cheap bound information provided by solving a relaxation of each subproblem, it is possible to reduce the number of total cases to study. This is called "intelligent enumeration".

Reformulation. A last major ingredient of today's methods is reformulation. In other words, if the problem is too hard, we can try to take a different viewpoint. There are different ways to reformulate. A first way to reformulate is to describe the problem differently: change the variables and the constraints. A second way to reformulate is to add extra information to the problem even if it may look redundant. Sometimes redundant information added to the original may be of interest for a relaxation. For example adding properly chosen linear inequalities to a problem provides the same set of feasible integer points but changes the linear relaxation. A third way to reformulate is to express the problem with other unknowns, or simply add new variables to a formulation.

Example. The concept of reformulation appears in mathematics in other cases. We can represent a point of the euclidian plane either with traditional cartesian coordinates (x, y) or with polar coordinates (r, θ). For some problems, the choice of coordinates can be crucial to reduce the computational complexity.

The state of the art in integer programming The branch-and-cut algorithm (see Section 1.2.5) has become the most standard algorithm to solve integer programs. It is based on the three main ingredients as explained above: using the bound information provided by the *linear relaxation*, performing intelligent *enumeration* (branch) using the bounds, and adding redundant *linear inequalities* that make the linear relaxation tighter (cut). These redundant linear inequalities are also called cutting planes or valid inequalities. This approach has turned out to be successful on many integer problems and it is used in all the commercial integer programming systems. Nevertheless there still exist some problems for which branch-and-cut does not work. We explain now one such problem.

Suppose that a group of people decide to share between themselves the objects contained in an attic of a house that they own together. There is a fixed number of each type of object and each object has a fixed value. The people want to perform the sharing in a fair way. However they have all put different amounts of money into the house. Therefore they want to share the objects in such a way that every one gets a total value and a number of objects proportional to their investment in the house. Since the objects are indivisible,

it is typically impossible to find a perfect solution. The problem is, therefore, to find a partition of the objects that minimizes the unfairness. This is an integer problem which is hard for the standard methods and we explain briefly why.

The branch-and-cut method relies on the power of enumeration and linear relaxation. In a typical problem, if you fix a large proportion of the variables, the linear relaxation gives you a good idea on how good a solution obtained from the fixing can be. You can sometimes avoid exploring a partial solution further because the linear relaxation tells you that the variables fixed lead to bad solutions. In the case of the sharing of the contents the attic, the situation is different. Suppose that you have fixed the value of 90% of the variables. In most of the cases, you will always manage to complete a perfect sharing by allowing splitting of the remaining objects. In other words, as soon as you allow fractions of some objects, the problem becomes too easy. Therefore the value given by the linear relaxation never gives any interesting information and the standard methods reduce to complete enumeration. In practice, experiments show that, even for small problems, standard systems spend hours without finding an optimal solution. The fifth chapter of this thesis shows that a proper reformulation of the problem using lattice methods (see Section 1.2.8) allows us to solve this and related problems within seconds.

This small example shows the necessity of having good alternatives to the branch-and-cut algorithm. This suggests that research in integer programming has to focus on two ideas.

- Continuing to improve the performance of the branch-and-cut methods. For example, it consists in providing interesting cutting planes, good implementations or polyhedral studies that enhance the efficiency of the method. A result can either be general or problem specific.
- Providing new ideas of algorithms that could be possible alternatives to the branch-and-cut. In this category fall new ideas to solve integer programs in general but also problem specific algorithms that work particularly well when the branch and cut fails.

The main content of the thesis These two lines of research are pursued in this text. Trying to improve the standard branch-and-cut algorithm is the most popular and active field in integer programming. It has led to many results on polyhedral studies and cutting planes for specific algorithms. The quantity of papers written on the subject is of course a rich reservoir of results. On the other hand the basics used in all the papers are often the same and it is sometimes difficult to see that many of these results are in fact particular cases of each other. Therefore possible unifications of all the theories are always welcome. In the first part of the thesis, we explore ways to unify a series of results on polyhedral studies. For example we touch the recurrent question of lifting,

i.e. generate cutting planes from the knowledge of cutting planes for simpler lower-dimensional sets. This procedure is often efficiently used in practice and several papers were written on the subject. Here we try to give a presentation of the lifting procedure. Although our model is not completely general, it allows us to handle most of the cases of the literature and hopefully simplify the presentation of the procedure. We also treat another topic related to the standard branch-and-cut algorithm and more particularly to the generation of efficient cutting planes. A lot of research has been carried out on the so-called single node flow set and its variants. There is again a large quantity of results in the literature and we try to unify them in the sense that we show that they all can be obtained using a procedure combining MIR and lifting, MIR being the most basic way to generate cutting planes for mixed integer problems. The goal of this is first to show that one procedure is sufficient to generate a lot of the known valid inequalities and secondly that one can classify the inequalities with respect to the way they can be generated.

As we indicated earlier, a second line of attack on integer programs is to find alternatives to the standard systems. The problem of sharing the contents of an attic showed that one may find problems for which the branch-and-cut is inefficient. One could expect it from the complexity theory. It is in fact assumed that it is impossible to find an algorithm which solves efficiently all instances of integer programming. In the second part of the thesis, we explore alternatives that have been recently proposed. Our first study is an attempt to combine two non standard approaches to integer programming, namely the group approach proposed by Gomory in 1969 [19] (see Section 1.2.7) and the Integral Basis Method published in 2003 by Haus, Köppe and Weismantel [29] (see Section 1.2.6). There is a natural way to combine the two methods and we explore it in detail. This entails a study of "extended formulations" for the group relaxation. Remark that an extended formulation can also be used on its own as an added feature of a branch-and-cut system. The two viewpoints are investigated. Finally we explore a solution method for the sharing problem mentioned earlier (in a financial form) and for which the traditional approaches are inefficient. In this case, the method is to reformulate the problem before using a branch-and-cut algorithm. The problem being better posed, the reformulation allows us to reduce considerably the computation time. However the computation of the reformulation may be slow for large instances and we show how to decompose the problem in order to speed up the method.

In the next section of the introduction, we come more formally to the mathematics of integer programming. The complete section is dedicated to a survey of all the methods to which we refer later in the text. We present several algorithms to solve general integer programs or mixed integer programs (or both). The first five subsections introduce the basics of integer programming and the most classical methods used today to solve integer programs. The reader used to the integer programming formalism and standard approaches

can skip this part without losing important information. The last three subsections introduce the non standard approaches explored in the thesis, namely an introduction to two primal methods, the group approach as it was introduced by Gomory in 1969 [19] and an introduction to lattices and the way to use them to reformulate integer programs. These three subsections refer to topics that are less known in the integer programming world.

After this long presentation of methods, the last section of this introduction outlines more formally the content and the main results of this text. For each topic, we also present a review of the existing literature.

1.2 Basic algorithms

In this section, we review the basic algorithms used to solve integer and mixed-integer programs. For ease of presentation, we explain all the methods on the same simple example.

Problem 1.1 (The hiker problem) *A hiker prepares an excursion and wonders what to take in his knapsack. Besides the food, he can fill the bag with two types of drinks that he likes: water and soda. One bottle of water costs 0.5 euros while a can of soda costs 0.9 euros. His budget is 3.5 euros. The space left in his bag is 4.5 l and one can suppose that a bottle of water occupies 1.1 l whereas a can needs 0.4 l. Furthermore he slightly prefers soda and therefore assumes that a can of soda is worth 3 "points" while a bottle of water is worth 2 "points". What does he have to buy if he wants to maximize the number of points he takes in his bag, while satisfying the constraints?*

A first step in trying to solve such a problem is to present a *formulation*. A common way of formulating is to define *variables* which represent the important unknown quantities of the problem and *constraints* which are mathematical relations that must be satisfied by the variables to give feasible solutions of the problem. We also define an *objective*. It is a mathematical function of the variables that has to be maximized or minimized. These three definitions provide a *mathematical program* that formulates the problem.

In the case of Problem 1.1, we define two variables. The number of bottles of water to be bought is represented by x_1 while the number of cans of soda is represented by x_2. There are two major constraints in the problem, the budget constraint, and the space constraint. The budget constraint can be formulated by the following condition on the variables

$$0.5x_1 + 0.9x_2 \leq 3.5.$$

Similarly the space constraint can be formulated by

$$1.1x_1 + 0.4x_2 \leq 4.5.$$

Finally we can model the objective by maximizing the "points" given for buying, or

$$\max\ 2x_1 + 3x_2.$$

To complete the model, it is important to state the domain of definition of the variables. In the case of Problem 1.1, both x_1 and x_2 have to take nonnegative integer values. The standard way of writing a mathematical program is to collect the information in a tableau of the form

$$\begin{array}{lrcrcl} \max & 2x_1 & + & 3x_2 & & \\ \text{s.t.} & 5x_1 & + & 9x_2 & \leq & 35 \\ & 11x_1 & + & 4x_2 & \leq & 45 \\ & x_1, x_2 \in \mathbb{Z}_+. & & & & \end{array} \tag{1.1}$$

When all the constraints and the objective are linear, the mathematical program is a *linear program*. When all the variables of a linear program must take integer values, we talk about an *integer program*. When some variables have to be integer and some others have to be real, we handle a *mixed-integer program*.

It is important to notice here that the formulation presented above is one formulation of the problem and there exist several possible mathematical programs that model the same problem. The formulation used is crucial in the choice of the method to solve the problem and in the speed of the different methods. This will become clear from the presentation of the algorithms. We now review some methods that have been proposed to solve integer programs of the type (1.1).

1.2.1 Enumeration

The simplest possible method is the enumeration of the finite set of all possible feasible solutions. Then among them determine the solution that maximizes (or minimizes) the objective.

In the case of Problem 1.1, one can derive bounds on the variables from the linear constraints. For example, the space and the nonnegativity constraints imply that $0 \leq x_1 \leq 4$. The budget and nonnegativity constraints imply that $0 \leq x_2 \leq 3$. This means that we have to check $5 \times 4 = 20$ possible pairs of values and determine the feasible pair that maximizes the objective. For such a small problem, it is of course pretty fast to check all possibilities. However as soon as the number of variables in the problem becomes too large, it is practically intractable to achieve such an enumeration as the following example shows.

Suppose we want to check every possible vector of size n, where every component can take two possible values: 0 or 1. This means that we have to check 2^n vectors. We first suppose that, for every vector, the verification is done in constant time, independently of n. On the other hand, we suppose that this check is done on two computers. Computer 1 would be rather fast

today and would check one billion (10^9) vectors per second. Computer 2, which would be a super computer of the future, is a million times faster and is able to verify 10^{15} vectors per second. Table 1.1 shows the time taken by each machine to do the complete check for given n. In this table, we see that a size for which

n	Computer 1	Computer 2
10	0 sec	0 sec
20	0 sec	0 sec
30	1 sec	0 sec
40	18 min	0 sec
50	13 days	1 sec
60	36 years	19 min
70	37000 y.	13 days
80	38 million years	38 years

Table 1.1: Time for enumerating 2^n vectors by supercomputers

the computers can do the complete check in reasonable time is $n = 40$ for Computer 1 and $n = 60$ for Computer 2. It means that by taking a computer one million times faster, we are able to solve a problem only 1.5 times larger. Moreover, as soon as the problem gets bigger, the number of cases to explore explodes. It becomes impossible to use the algorithm. This suggests that this brutal approach has its limits and cannot be considered as an alternative as soon as the problem has a decent size. In the following, we present other algorithms that try to improve on this very primitive method. Although none of them have a non exponential bound on the computation time, they all tend to have a limit of computability that is larger and thus more suitable to real life problems. And before presenting the algorithms, we give some geometric intuition on integer programs.

1.2.2 Geometric approach

It is very useful to "visualize" the set of feasible solutions of an integer program. It allows one to see where the optimal solution is located. Unfortunately, it is only possible to see two dimensional problems completely. It is therefore hopeless to have a geometric algorithm for arbitrarily large integer programs. Nevertheless it is very interesting to have a geometric insight to what happens in an integer program.

Problem 1.1 is represented in Figure 1.1. In the 2D-space, linear constraints are represented by half-spaces. All the nonnegative integer points that satisfy the two constraints are represented in grey in the picture. The non-feasible integer points are represented in black. The direction depicted on the top right

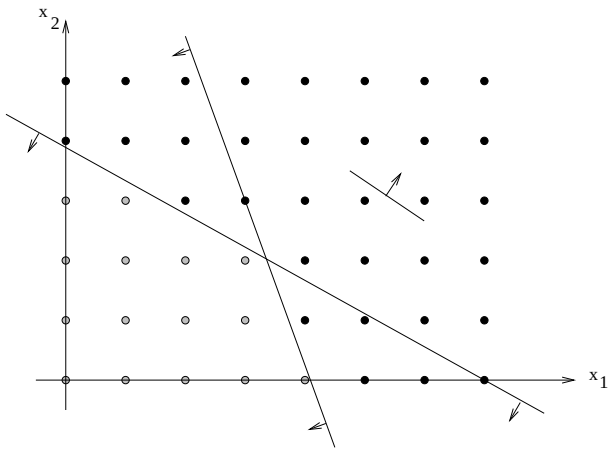

Figure 1.1: Geometric representation of Problem 1.1

corner indicates the direction of the objective function. In this picture, it is pretty easy to see that the feasible point that maximizes the objective function corresponds to $x_1 = 3, x_2 = 2$.

This geometric intuition cannot really be called an algorithm. Nevertheless it provides all the ingredients needed to present the algebraic methods. From the particular case of Figure 1.1, some important general observations can be made. It can be first seen that the set of linear constraints involved in an integer program define a *polyhedron* of the continuous space. A polyhedron is a convex set of the real space and can be defined in two different ways.

Definition 1.1 *A set $P \subseteq \mathbb{R}^n$ is a* polyhedron *if there exist $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ such that $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.*

The following proposition gives an equivalent way of defining polyhedra.

Proposition 1.1 *A set $P \subseteq \mathbb{R}^n$ is a polyhedron if and only if there exist vectors $p^1, \ldots, p^r \in \mathbb{R}^n$ (*extreme points*) and $t^1, \ldots, t^s \in \mathbb{R}^n$ (*extreme rays*) such that*

$$P = \{x \in \mathbb{R}^n : x = \sum_{i=1}^{r} \lambda^i p^i + \sum_{j=1}^{s} \mu^j t^j,\ \sum_{i=1}^{r} \lambda^i = 1,\ \lambda^i, \mu^j \geq 0\}.$$

Figure 1.2 shows the polyhedron P defined by the linear constraints of the formulation (1.1) of Problem 1.1. The extreme points (or vertices) of the polyhedron are represented by circles. This polyhedron has no extreme rays.

There are several different polyhedra that include exactly the same sets of feasible integer points. Of particular interest is the polyhedron defined as the

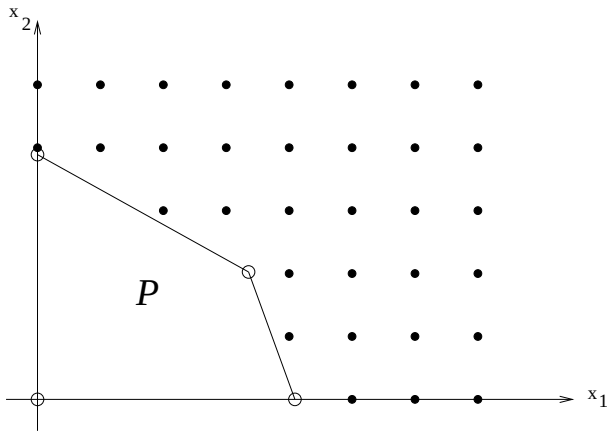

Figure 1.2: The polyhedron P defined by the constraints of Problem 1.1

convex hull of all feasible integer points. In that case, all the vertices of the polyhedron are integral.

Definition 1.2 *Let $S \subseteq \mathbb{R}^n$ with $S \neq \emptyset$, the convex hull of S is defined as*

$$conv(S) = \{x \in \mathbb{R}^n : \text{there exist } k \in \mathbb{Z}_+, y^1, \ldots, y^k \in S \text{ and } \lambda^1, \ldots, \lambda^k \geq 0 \text{ such that } x = \textstyle\sum_{i=1}^k \lambda^i y^i,\ \sum_{i=1}^k \lambda^i = 1 \quad \}.$$

In the case of Problem 1.1, this particular polyhedron is defined by

$$\begin{aligned} P = \{x_1, x_2 \in \mathbb{R}_+ : \\ & x_2 \leq 3 \\ & x_1 + 2x_2 \leq 7 \\ & 2x_1 + x_2 \leq 8 \qquad \}, \end{aligned}$$

and is represented in Figure 1.3. This means that another valid model for Problem 1.1, although it does not really correspond to any "physical" meaning is

$$\begin{array}{lrcrcl} \max & 2x_1 & + & 3x_2 & & \\ \text{s.t.} & & & x_2 & \leq & 3 \\ & x_1 & + & 2x_2 & \leq & 7 \\ & 2x_1 & + & x_2 & \leq & 8 \\ & x_1, x_2 \in \mathbb{Z}_+. & & & & \end{array} \tag{1.2}$$

1.2.3 Cutting plane algorithms

The geometry of integer programs provides an intuition on how to solve such problems. Unfortunately no geometric argument can be used as soon as the

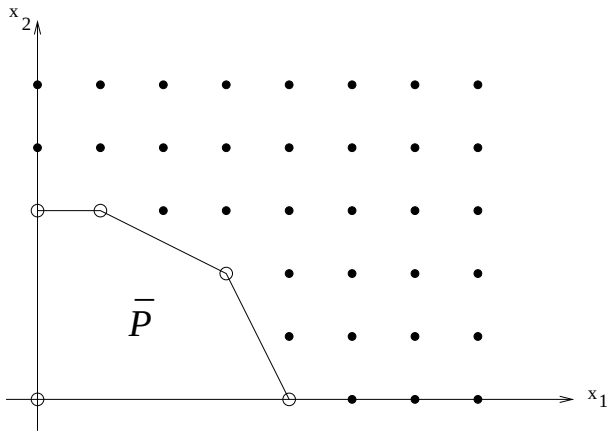

Figure 1.3: The convex hull $\bar{P}$ of all integer solutions of Problem 1.1

dimension of the problem is too large. As said before, it is also inefficient to consider a complete enumeration. Since the problem is then too difficult to solve, we try to solve an easier variant of it. If we can tackle the easier variant, one can hope to extract some interesting information for the harder problem. The easier version of a problem is called a *relaxation*.

Definition 1.3 *Let $X \in \mathbb{R}^n$. A set $Y \in \mathbb{R}^n$ is called a* relaxation *of X if $X \subseteq Y$.*

Solving a relaxation of a problem provides a bound on the optimal value of the original problem.

Proposition 1.2 *Let $x^* = arg\ max\ \{cx : x \in X\}$ and let Y be a relaxation of X. If $y^* = arg\ max\ \{cy : y \in Y\}$, then $cy^* \geq cx^*$.*

Suppose now that we are trying to solve an integer program max $cx,\ Ax \leq b,\ x \in \mathbb{Z}^n_+$, a very natural relaxation is the set of all real points that satisfy the constraints $Ax \leq b$, i.e. ignoring the integrality constraints.

Definition 1.4 *Let $X = \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$. The* linear relaxation *of X is*

$$LP(X) = \{x \in \mathbb{R}^n_+ : Ax \leq b\}.$$

It is well known that optimizing over a polyhedron in the real space can be done efficiently with either the simplex algorithm (which has a good running time in practice) or an interior point method (which is guaranteed to run in polynomial time). The bound given by solving the linear relaxation is called the LP bound. Now if we suppose that some optimal solution y of the linear

relaxation is integral, it implies that y is also an optimal solution of the integer program.

If we go back to Problem 1.1 and solve the linear relaxation of (1.1), we obtain $(x_1, x_2) = (265/79, 160/79)$ which gives a bound of 12.79 on the value of the optimal integer solution. Unfortunately this does not provide an integer solution of the original problem. However if we solve the formulation (1.2), the solution produced is $(x_1, x_2) = (3, 2)$, i.e. the optimal integral solution of value 12. The fact is that among the optimal solutions of a linear program, there is always one located in an extreme point of the corresponding polyhedron. Therefore, when one solves the linear relaxation of an integer program, the optimal integral solution is always found when the polyhedron that models the integer program is the convex hull of all integer feasible points. The principle of the cutting plane method is to iteratively find some new *valid* constraints for the integer program to be as close as possible to the convex hull until some integer solution to the linear relaxation is found. Let us now define the concept of valid inequality.

Definition 1.5 *Let $X \subseteq \mathbb{Z}^n$. An inequality $ax \leq a_0$ is* valid *for X if $ax \leq a_0$ holds for all $x \in X$.*

A crucial question in the cutting plane algorithm is the *separation*.

Definition 1.6 *Let $X \subseteq \mathbb{Z}^n$ and $y \in \mathbb{R}^n$ such that $y \notin conv(X)$. The* separation problem *is to find a valid inequality $ax \leq a_0$ for X that is not satisfied by y, i.e. such that $ay > a_0$.*

Let us assume that we solve the mixed-integer program

$$\begin{array}{ll} \max & c_1x + c_2y \\ \text{s.t.} & A_1x + A_2y = b \\ & x \in \mathbb{R}^{n_1}_+, y \in \mathbb{Z}^{n_2}_+. \end{array} \tag{1.3}$$

We define a set of active constraints $\mathcal{C}$ with respect to the problem. We also introduce the notation $X = \{x \in \mathbb{R}^{n_1}_+, y \in \mathbb{Z}^{n_2}_+ : A_1x + A_2y = b\}$. The cutting plane algorithm can be outlined in the following way.

Cutting Plane Algorithm

(1) **Initialize** $\mathcal{C}_1 = \{(x, y) \in \mathbb{R}^{n_1}_+ \times \mathbb{R}^{n_2}_+ : A_1x + A_2y = b\}$ and $i = 1$.
(2) **Find** $(x^*_i, y^*_i) = \arg\max \{c_1x + c_2y : (x, y) \in \mathcal{C}_i\}$.
(3) **If** $(x^*_i, y^*_i) \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{n_2}_+$ **then**
(4) **Return** (x^*_i, y^*_i) as optimal solution
(5) **Else**
(6) **Find** a valid inequality for X, $a^i_1x + a^i_2y \leq a^i_0$, separating (x^*_i, y^*_i) from $\text{conv}(X)$.
(7) **Set** $\mathcal{C}_{i+1} := \mathcal{C}_i \cap \{(x, y) \in \mathbb{R}^{n_1}_+ \times \mathbb{R}^{n_2}_+ : a^i_1x + a^i_2y \leq a^i_0\}$.
(8) **Set** $i := i + 1$ and **go to** step 2.
(9) **End**

Step 2 can be carried out efficiently by a linear programming algorithm like the simplex algorithm. Step 6 is crucial and non trivial. A lot of research has been done on finding good valid inequalities and on the separation problem. Finally an important remark is that, as such, there is no guarantee that the cutting plane algorithm terminates. We now see an example of the algorithm on Problem 1.1 presented above.

Example: We try to solve the integer problem

$$\max \{2x_1 + 3x_2 : x \in X\},$$

where $X = \{x_1, x_2 \in \mathbb{Z}_+ : 5x_1 + 9x_2 \leq 35,\ 11x_1 + 4x_2 \leq 45\}$.

Iteration 1
We start by considering the linear relaxation

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 5x_1 + 9x_2 \leq 35 \qquad (1.4) \\ & 11x_1 + 4x_2 \leq 45 \qquad (1.5) \\ & x_1, x_2 \in \mathbb{R}_+. \end{aligned}$$

Computing the optimal solution of this linear relaxation yields $x^{1*} = (\frac{265}{79}, \frac{160}{79})$. This point is shown by a circle in Figure 1.4. It is not integral. Therefore we

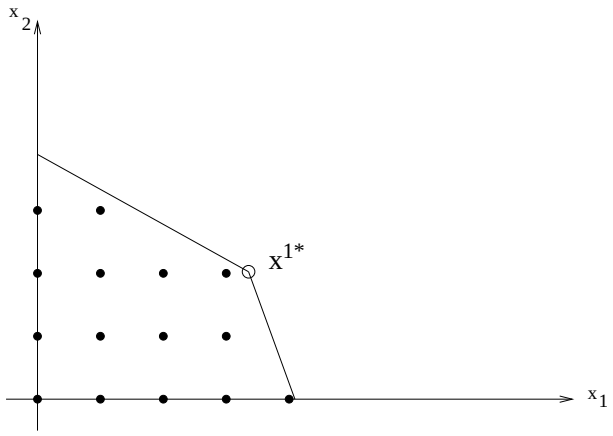

Figure 1.4: Optimal solution of the linear relaxation after Iteration 1

need to find some valid inequality separating x^{1*} from the convex hull of all integer solutions, i.e. conv(X). For the time being, we do not know any systematic way to generate a separating valid inequality. But let us do it

in the following way. If we consider some linear combination of the two main inequalities, we still obtain a valid inequality. For example 2(1.4) + (1.5) yields

$$21x_1 + 22x_2 \leq 115.$$

This can be rewritten as

$$x_1 + \frac{22}{21}x_2 \leq \frac{115}{21}. \tag{1.6}$$

As the variables are nonnegative, a weaker form of (1.6) is

$$x_1 + x_2 \leq \frac{115}{21} \tag{1.7}$$

because the coefficient of x_2 is smaller. But now since the left hand side of (1.7) is always integer for all the feasible integral points, we can round down the right hand side and write that

$$x_1 + x_2 \leq 5 \tag{1.8}$$

is valid for all points in X. We can check that x^{1*} violates the inequality. Indeed $\frac{265}{79} + \frac{160}{79} = 5.38$. Hence we add inequality (1.8) to the model and iterate. Figure 1.5 shows the inequality (1.6) obtained by linear combination. On the same figure, it can also be seen that the inequality (1.8) does include

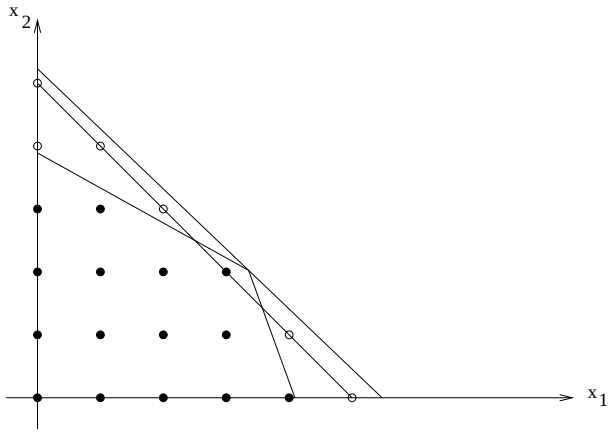

Figure 1.5: An inequality and a separating inequality obtained from it

the same nonnegative integer points as (1.6).

Iteration 2
We now solve

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 5x_1 + 9x_2 \leq 35 && (1.9) \\ & 11x_1 + 4x_2 \leq 45 && (1.10) \\ & x_1 + x_2 \leq 5 && (1.11) \\ & x_1, x_2 \in \mathbb{R}_+. \end{aligned}$$

The optimal solution is $x^{2*} = (\frac{10}{4}, \frac{10}{4})$ and is represented by a circle in Figure 1.6. If we now add the valid inequality $x_2 \leq 4$ to (1.9), divide by 5 and round

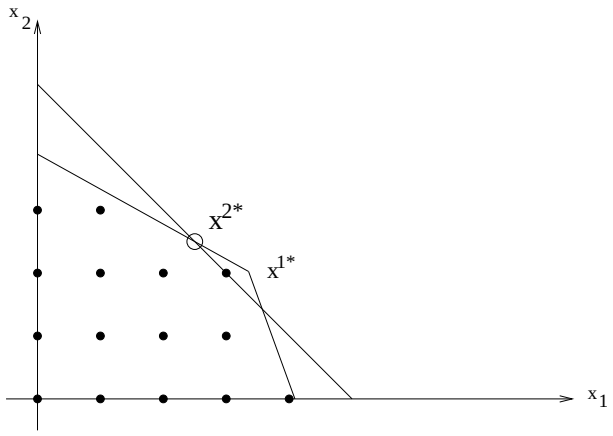

Figure 1.6: The optimal solution of the linear relaxation at Iteration 2

down, with the same argument as before, we obtain a valid inequality

$$x_1 + 2x_2 \leq 7, \tag{1.12}$$

separating $(\frac{10}{4}, \frac{10}{4})$ from $\text{conv}(X)$. Hence, we add (1.12) to the model.

Iteration 3
By reoptimizing, we obtain the solution $x^{3*} = (3, 2)$ as indicated in Figure 1.7. This point is integral and therefore is the optimal solution of the integer program.

As the example shows, the crucial step in the algorithm is to find valid inequalities, also called cutting planes. This has been a very active field of research in the past decades. There is some kind of library of different valid inequalities depending on the problem structure. There are also some general ways of generating valid inequalities. In this introduction, we present the

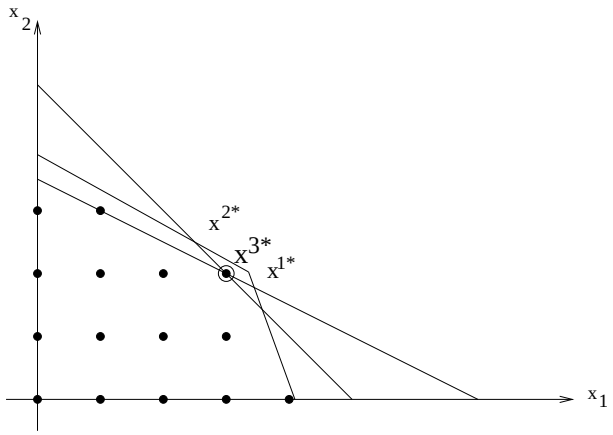

Figure 1.7: The optimal solution of the linear relaxation at Iteration 3

two most general types of valid inequalities: the Chvátal-Gomory cut and the mixed-integer rounding.

Proposition 1.3 (Chvátal-Gomory cut) *Let*

$$X = \{y \in \mathbb{Z}^n_+ : \sum_{i=1}^{n} a_i y_i \leq b\}, \tag{1.13}$$

where $a_i, b \in \mathbb{R}$*, for all* $i = 1, \ldots, n$*. The inequality*

$$\sum_{i=1}^{n} \lfloor a_i \rfloor y_i \leq \lfloor b \rfloor \tag{1.14}$$

is valid for X*.*

Proof: If we round down the left hand side of (1.13), the inequality remains valid beacause it is weaker. Then we can round down the right hand side of (1.13) because the left hand side always takes integer values for points in X.□

The mixed-integer rounding procedure is some sort of generalization of Proposition 1.3 to the case of a mixed-integer program. Nevertheless it is also applicable to pure integer programs and even provides at least as strong inequalities. To introduce it, we first need to define an important function.

Definition 1.7 *Let* $0 < \alpha \leq 1$*. The MIR function* $F_\alpha : \mathbb{R} \to \mathbb{R}$ *can be defined as*

$$F_\alpha(x) = \lfloor x \rfloor + \frac{(\mathcal{F}(x) - \alpha)^+}{1 - \alpha},$$

where $\mathcal{F}(x)$ *denotes the fractional part of* x*, and* $(a)^+ = max\ \{0, a\}$*.*

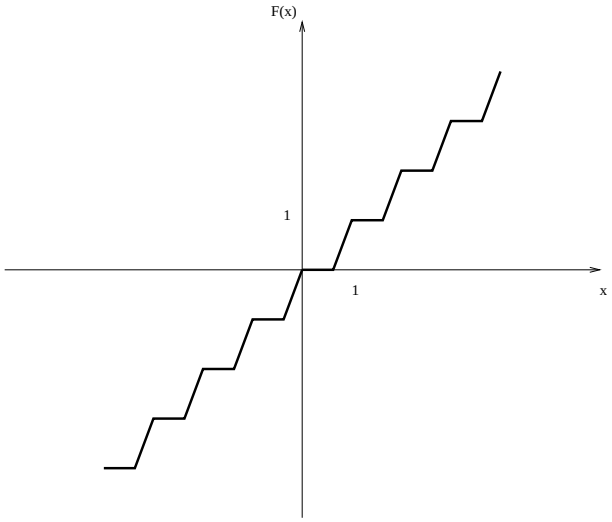

Figure 1.8: The MIR function $F_{3/5}(x)$

The MIR function $F_{3/5}$ is represented in Figure 1.8. The slope of F at the origin defines another useful function.

Definition 1.8 *For $0 < \alpha \leq 1$, the function $\bar{F}_\alpha : \mathbb{R} \to \mathbb{R}$ is defined by*

$$\bar{F}_\alpha(x) = \begin{cases} 0 & \text{if } x \geq 0 \\ \frac{x}{1-\alpha} & \text{if } x < 0. \end{cases}$$

We can now introduce the Mixed Integer Rounding (MIR).

Proposition 1.4 (MIR) *Let*

$$X = \{x \in \mathbb{R}_+^{n_1}, y \in \mathbb{Z}_+^{n_2} : \sum_{i=1}^{n_1} c_i x_i + \sum_{j=1}^{n_2} a_j y_j \leq b\},$$

where $a_j, b, c_i \in \mathbb{R}$ for all i and j. The inequality

$$\sum_{i=1}^{n_1} \bar{F}_\alpha(c_i) x_i + \sum_{j=1}^{n_2} F_\alpha(a_j) y_j \leq F_\alpha(b),$$

where $\alpha = \mathcal{F}(b)$, is valid for X.

There are, of course, other ways to generate valid inequalities and some of them are often problem dependent. Some methods are explored in the next chapters of the thesis.

Propositions 1.3 and 1.4 provide ways to automatically generate valid inequalities for a problem. Nevertheless it is very important to know whether the generated inequalities are weak, strong or maybe as strong as possible. The ideal situation is when the polyhedron that we obtain has integer vertices. In that case, we know that solving a linear program over the polyhedron always yields an integer solution which is therefore optimal with respect to the corresponding integer problem. This is the case, for example, for the formulation (1.2) of Problem 1.1. In this situation, we say that the valid inequalities (1.2) are facet-defining for the convex hull of all integer integer solutions of the hiker problem. We make this more precise in the next definitions.

Definition 1.9 *Consider $a_1, \ldots, a_p \in \mathbb{R}^n$. We say that $a_1, \ldots, a_p$ are* affinely independent *if*

$$(a_2 - a_1, a_3 - a_1, \ldots, a_p - a_1)$$

are linearly independent.

An important concept related to polyhedra is the *dimension*.

Definition 1.10 *Let $P \subseteq \mathbb{R}^n$ be a polyhedron. P is of dimension p if there exist $p+1$ affinely independent vectors $a_i \in P$.*

Definition 1.11 *Let $dx \leq d_0$ be a valid inequality for P, polyhedron of dimension p, $dx \leq d_0$ is a facet-defining inequality for P if there exist p affinely independent vectors $a_i \in P$ such that $da_i = d_0$ for all $1 \leq i \leq p$.*

Geometrically a facet-defining inequality of a polyhedron P is a hyperplane of dimension one less than that of P and which defines the border of P. For example it corresponds to a face of a dice.

This discussion allows us to say that when we look for valid inequalities for an integer set X, the best we can do is to produce facet-defining inequalities for conv(X). Remark that we sometimes use the term facet instead of the long "facet-defining inequality".

1.2.4 Branch-and-Bound

The drawback of the cutting plane algorithm is that there is no guarantee of termination. We have also seen before that pure enumeration is hopeless. The method presented here is an improvement of the enumeration technique. Instead of simply looking at every combination, we enumerate the variables one at a time. Each time we fix some variable(s), we try to compute some bound on the objective after the fixing. This allows us to know that some cases are not worth being explored further and to reduce the enumeration tree. We first explain the method on Problem 1.1 before giving a theoretical presentation of the method.

Example: We again solve the integer problem max $\{2x_1 + 3x_2 : x \in X\}$, with $X = \{x_1, x_2 \in \mathbb{Z}_+ : 5x_1 + 9x_2 \leq 35,\ 11x_1 + 4x_2 \leq 45\}$.

Iteration 1: As in a cutting plane method, we start by considering the linear relaxation of the original formulation. That is what is called the *top node*. As before, we obtain the solution $x^{1*} = (\frac{265}{79}, \frac{160}{79})$ with an objective value of 12.79. The idea of the branch-and-bound is to enumerate different cases but, for each case, to compute a bound on the optimal value in order to draw some conclusions on the different new cases to explore. This bound is usually obtained by the computation of a linear relaxation. Going back to the example, since we are looking for an integral solution, $\frac{265}{79}$ is not an acceptable value. Therefore it is true to say that either $x_1 \geq \lceil \frac{265}{79} \rceil = 4$ or $x_1 \leq 3$. The enumeration tree here splits into two branches: a first case is $x_1 \geq 4$ and the second case is $x_1 \leq 3$.

Iteration 2 We study the first branch $x_1 \geq 4$. The first step is to compute a bound on the objective function in this branch. To do this, we compute the value of the linear relaxation

$$\begin{array}{ll} \max & 2x_1 + 3x_2 \\ \text{s.t.} & 5x_1 + 9x_2 \leq 35 \\ & 11x_1 + 4x_2 \leq 45 \\ & x_1 \geq 4 \\ & x_1, x_2 \in \mathbb{R}_+. \end{array}$$

The optimal solution is $x^{2*} = (4, \frac{1}{4})$ with a value of 8.75. This can be seen in

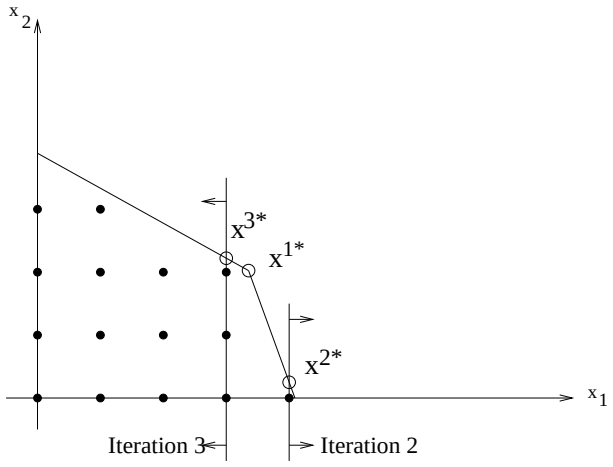

Figure 1.9: Iteration 1,2,3 of the branch-and-bound algorithm

Figure 1.9. Suppose now that a heuristic produced the simple feasible integer

solution $\hat{x} = (0, 3)$ with a value of 9. The conclusion is that no better solution than $\hat{x}$ can be found in the branch $x_1 \geq 4$. Therefore we say that we *prune* the branch by *bound*. In other words we do not explore further this case since we know that all potential integer solutions have a value smaller than 8.75 and therefore smaller than 9. Hence we stop this branch and go to the case $x_1 \leq 3$.

Iteration 3 We study now the case $x_1 \leq 3$. Again it is important to obtain an upper bound on the value of the solutions in the branch. Therefore we solve

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 5x_1 + 9x_2 \leq 35 \\ & 11x_1 + 4x_2 \leq 45 \\ & x_1 \leq 3 \\ & x_1, x_2 \in \mathbb{R}_+. \end{aligned}$$

As we can see in Figure 1.9, the optimal solution is $x^{3*} = (3, \frac{20}{9})$ with an objective value of 12.67. This is still above 9 but not integral. New branches need to be explored. As x_2 is fractional, we enumerate two cases for x_2 namely $x_2 \geq 3$ or $x_2 \leq 2$. Let us first study the case $x_2 \leq 2$.

Iteration 4 We study thus the case $x_1 \leq 3$ and $x_2 \leq 2$. Figure 1.10 shows the case of Iteration 4. We solve the linear relaxation

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 5x_1 + 9x_2 \leq 35 \\ & 11x_1 + 4x_2 \leq 45 \\ & x_1 \leq 3 \\ & x_2 \leq 2 \\ & x_1, x_2 \in \mathbb{R}_+. \end{aligned}$$

The optimal solution is $x^{4*} = (3, 2)$ with a value of 12. It is integral, larger than 9, and gives thus a new lower bound on the value of the optimal integral solution. It is not needed to explore the branch further since no better other integral solution can be found. We say that we *prune* this branch by *optimality*. A node has still to be explored.

Iteration 5 We study the case $x_1 \leq 3$ and $x_2 \geq 3$. Figure 1.10 shows the set studied. The linear relaxation yields $x^{5*} = (\frac{8}{5}, 3)$ with a value of 12.2. x_1^{5*} is fractional and therefore we explore two more cases: $x_1 \leq 1$ or $x_1 \geq 2$.

Iteration 6 For the case $2 \leq x_1 \leq 3$, $x_2 \geq 3$, it appears that the linear relaxation is infeasible. It is clear that the search in the node must be stopped. We say that we *prune* the node by *infeasibility*.

Iteration 7 The case $x_1 \leq 1$ and $x_2 \geq 3$ gives the solution $x^{7*} = (1, 3)$ with a value of 11. This node is prunded both by bound and by optimality.

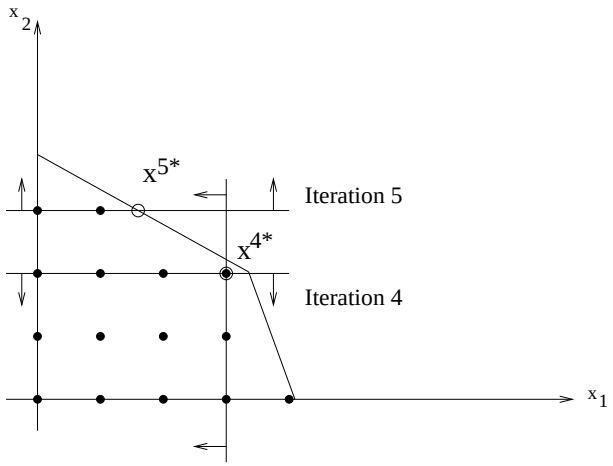

Figure 1.10: Iteration 4,5 of the branch-and-bound algorithm

Now all the nodes have been completely explored. We can deduce that the best solution is $x^{4*} = (3, 2)$, i.e. the best integral solution found during the computation. We show a picture of the branch-and-bound tree in Figure 1.11.

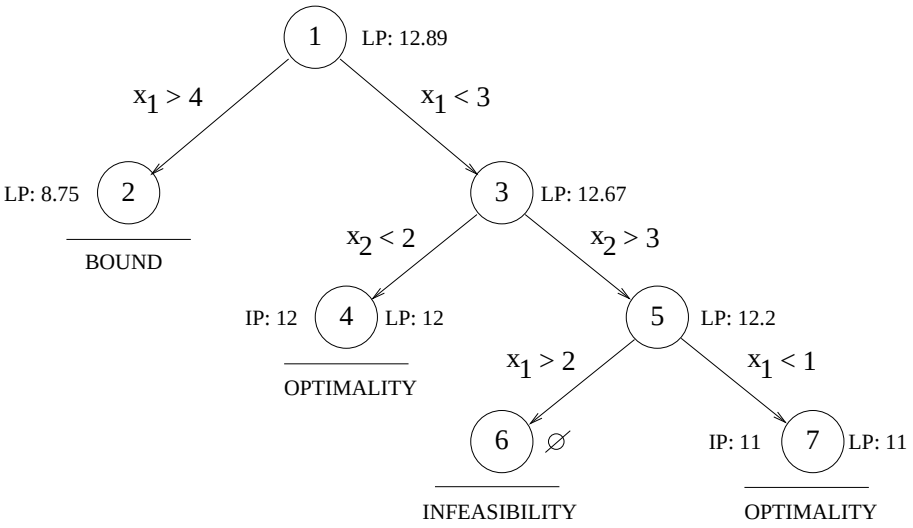

Figure 1.11: The branch-and-bound tree

Based on this example we are now able to come up with an algorithm. For

this purpose, we need to define some sets. The set of all active nodes is denoted by N. Each node i has a certain number of constraints attached to it, denoted by $C(i)$. We present the algorithm for a maximization problem in Table 1.2. In

(1) **Initialize** $N = \{1\}, C(1) = X, UB = +\infty, LB = -\infty, n = 1$
(2) **While** $N \neq \emptyset$ **do**
(3) **Choose** $i \in N$
(4) Solve the LP relaxation $LP(i) = \max\{cx : x \in C(i)\}$ with sol. $x^*(i)$.
(5) **If** $LP(i) < LB$ **or** $C(i) = \emptyset$, **then**
(6) $N = N \setminus \{i\}$ (Prune by bound or infeasibility)
(7) **Else**
(8) **If** $x^*(i) \in \mathbb{Z}^n_+$ **then**
(9) $LB = \max\{LB, LP(i)\}$
(10) $N = N \setminus \{i\}$ (Prune by optimality)
(11) **Else**
(12) **Chose** some variable $x^*_k(i)$
(13) **Create** two branches $C(n+1) = C(i) \cap \{x_k \leq \lfloor x^*_k(i) \rfloor\}$
(14) and $C(n+2) = C(i) \cap \{x_k \geq \lceil x^*_k(i) \rceil\}$
(15) **Set** $n = n + 2$, and $N = N \setminus \{i\}$
(16) **End**
(17) **End**
(18) **End**
(19) **Return** LB as the optimal value.

Table 1.2: LP-based branch-and-bound algorithm

this algorithm, some steps can be replaced by more general ones. At each step of the algorithm, some better lower bounds can be sought in order to cut off some nodes more quickly. In Step (4), we compute an upper bound of the node by a linear relaxation. Every other method that computes an upper bound is of course suited to replace the computation of the linear relaxation. A crucial step in the algorithm is the choice of node i in Step (3). The most common strategies are of the type "depth-first-search" or "breadth-first-search". The depth-first-search allows one to find as quickly as possible a feasible solution in order to have a lower bound. The breadth-first-search allows one to have good upper bounds to know how far from the optimum are the current solutions. The choice in Step (3) is called the "node selection". Finally the choice in Step (12) is also of importance. We usually decide to branch on a fractional variable in the LP solution. The branching can be done either on the most fractional variable or on the closest to an integer. Some algorithms also "look ahead" i.e. try to see which choice of variable produces the best bound before actually chosing. This is called "strong branching". In general, the choice in Step (12) is called the "variable selection" or branching rule.

1.2.5 Branch-and-Cut

The methods of the two last sections both use the linear relaxation to draw some conclusions on the probable solution. It is possible to combine them. The idea of a branch-and-cut algorithm is to use some cutting planes within the branch-and-bound algorithm. This produces tighter bounds and LP solutions closer to actual feasible integer solutions. The cutting phase can be carried out either at the top node by generating globally valid inequalities or during the branching phase. In that case, the cutting planes generated are only valid locally. The branch-and-cut algorithm is implemented in most commercial systems that solve integer and mixed-integer programs (IP and MIP). It has turned out to be a quite successful approach on a series of a problem. Furthermore these systems continue to improve with better cuts and better branching rules. However there remain some problems for which they are pretty inefficient. For these problems, one needs good formulations and sometimes even other algorithms. As a conclusion, it is important to note here that the question of the quality of a formulation remains particularly crucial.

1.2.6 Primal Methods

A very important notion in mathematical programming is that of primal-dual pair of problems. The primal problem is typically the original maximization (minimization resp.) problem that we want to solve. Its dual is a problem with the property that all its feasible solutions have a value of the objective greater (lower resp.) than or equal to the value of the objective of all the feasible solutions of the original problem. A *strong dual* is a dual problem with the property that its optimal solution takes the same value of the objective as the optimal solution of the primal problem, if optimal solutions do exist. In integer programming, we typically do not know beforehand an analytic form of a strong dual problem. However the values of the relaxations of a problem provide bounds and we refer to them as *dual bounds*. The algorithms based on the linear relaxations are dual. It means that instead of solving the original problem, we solve relaxations that provide bounds. We first focus on the bound before searching for feasible solutions. Those feasible solutions are found once the value of the dual has been made close enough to the value of the primal problem. However a very natural way of solving problems is to come up with a solution that we try to improve to another feasible solution. Once it becomes impossible to improve again, we focus on proving the optimality of the solution. We call that approach a *primal algorithm*. The advantage is that it guarantees to keep a feasible solution throughout the computation, feasible solution that could possibly be used in practice, even when the computation is not finished.

In this subsection, we present the Gomory-Young primal all-integer algorithm and a recent primal algorithm proposed by Haus, Köppe and Weismantel based on the Gomory-Young algorithm. We start by presenting the Gomory-

Young algorithm on the example of Problem 1.1.

The principle of the algorithm is to adapt the simplex algorithm to the integer case. As in the simplex algorithm, we start with a feasible solution and, in this case, with an all-integer tableau representation. Concerning Problem 1.1, we need to introduce slack variables on each constraint, $s_1, s_2 \in \mathbb{Z}_+$. The first feasible solution is thus $x^1 = (x_1, x_2, s_1, s_2) = (0, 0, 35, 45)$ and the tableau representation is

$$\begin{array}{rl} \max & 2x_1 + 3x_2 \\ \text{s.t.} & 5x_1 + 9x_2 + s_1 = 35 \\ & 11x_1 + 4x_2 + s_2 = 45 \\ & x_1, \quad x_2, \; s_1, \; s_2 \in \mathbb{Z}_+. \end{array} \tag{1.15}$$

The variable x_1 has a positive reduced cost. It is thus interesting to try to increase its value. By comparing the ratios $\frac{35}{5}, \frac{45}{11}$, we see that (1.15) is the first constraint to become tight when the value of x_1 increases. In a standard simplex algorithm, we would pivot on (1.15) and obtain a value of $\frac{45}{11}$ for x_1. In this algorithm, we want to prevent fractionality and to enforce x_1 to be integer. To do that, we divide (1.15) by the coefficient of x_1 and generate a Chvátal-Gomory cut

$$x_1 + \lfloor \frac{4}{11} \rfloor x_2 \leq \lfloor \frac{45}{11} \rfloor,$$

i.e. $x_1 \leq 4$ or equivalently

$$x_1 + s_3 = 4, \tag{1.16}$$

with $s_3 \in \mathbb{Z}_+$. We add (1.16) to the model and make x_1 basic in the constraint. We obtain the new tableau

$$\begin{array}{rl} \max & 3x_2 - 2s_3 + 8 \\ \text{s.t.} & 9x_2 + s_1 - 5s_3 = 15 \\ & 4x_2 + s_2 - 11s_3 = 1 \\ & x_1 + s_3 = 4 \\ & x_1, \; x_2, \; s_1, \; s_2, \; s_3 \in \mathbb{Z}_+. \end{array} \tag{1.17}$$

The new integer solution is $x^2 = (x_1, x_2, s_1, s_2) = (4, 0, 15, 1)$. The move made in this first iteration is shown in Figure 1.12. The new inequality (1.16) determines this move. In the new tableau, x_2 has still a positive reduced cost and it is therefore interesting to try to augment it. The tighter constraint for x_2 is (1.17). A pivot on this row would make x_2 fractional ($\frac{1}{4}$). We prevent it by generating a Chvátal-Gomory cut on equation (1.17) divided by 4, i.e. the coefficient of x_2. It gives

$$x_2 - 3s_3 + s_4 = 0, \tag{1.18}$$

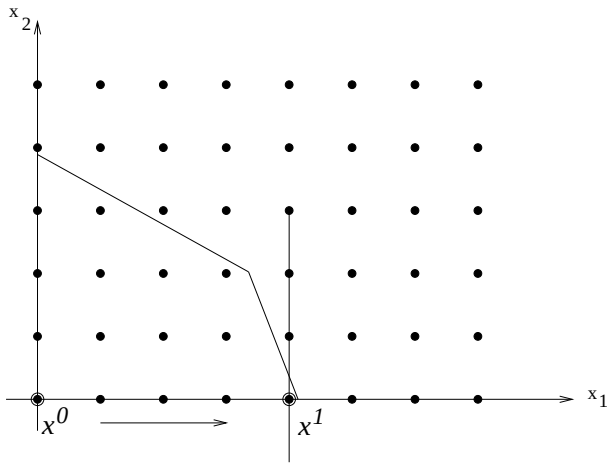

Figure 1.12: The first iteration of the Gomory-Young algorithm

where $s_4 \in \mathbb{Z}_+$ represents a slack variable. We add again (1.18) to the model and make x_2 basic in the tableau to obtain

$$\begin{array}{rrrrrrrr}
\max & & & & 7s_3 - & s_4 + 8 \\
\text{s.t.} & & s_1 & + 22s_3 - & 9s_4 = 15 \\
& & s_2 + & s_3 - & 4s_4 = 1 \\
& x_1 & + & s_3 & = 4 \\
& x_2 & - & 3s_3 + & s_4 = 0 \\
& x_1,\ x_2, & s_1, & s_2, & s_3, & s_4 \in \mathbb{Z}_+.
\end{array}$$

In this case, no new point is obtained because one cannot find an integer point on the way to the complete fractional pivot. This is always the case when the right hand side of the row on which we pivot is smaller than the coefficient of the variable that we want to augment. Therefore $x^2 = (x_1, x_2, s_1, s_2) = (4, 0, 15, 1)$. This iteration is represented in Figure 1.13. Remark that the Chvátal-Gomory cut (1.18) can be represented on the picture via its projection onto the (x_1, x_2)-space. It suffices to substitute (1.16) in (1.18). The cut can thus be written $3x_1 + x_2 \leq 12$.

The next iteration does not provide any new augmentation as the Gomory cut generated

$$s_3 - s_4 + s_5 = 0,$$

with $s_5 \in \mathbb{Z}_+$ being a slack variable, has 0 as right hand side. After this pivot,

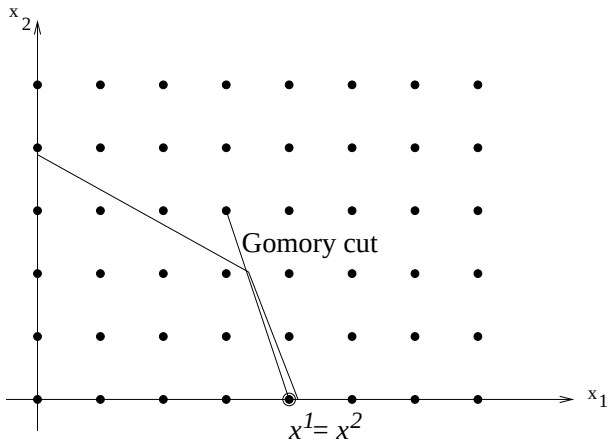

Figure 1.13: The second iteration of the Gomory-Young algorithm

the tableau is still not optimal. A new Gomory cut

$$s_4 - 2s_5 + s_6 = 1 \tag{1.19}$$

is generated. At this stage of the algorithm the simplex tableau is

$$\begin{array}{rrrrrrrrr}
\max & & & & & & s_5 - & 4s_6 + 12 \\
\text{s.t.} & & s_1 & & & + 4s_5 - & 13s_6 = 2 \\
& & & s_2 & & - 7s_5 + & 3s_6 = 4 \\
& x_1 & & & & + \ s_5 - & s_6 = 3 \\
& & x_2 & & & - \ s_5 + & 2s_6 = 2 \\
& & & & s_3 & - \ s_5 + & s_6 = 1 \\
& & & & & s_4 - 2s_5 + & s_6 = 1 \\
& x_1, & x_2, & s_1, & s_2, & s_3, & s_4, \quad s_5, & s_6 \in \mathbb{Z}_+.
\end{array} \tag{1.20}$$

Note that here we have reached the optimal point but we do not have a certificate of optimality because s_5 has still a positive reduced cost. The last iteration produces this certificate. Indeed generating the cut on (1.20) yields

$$s_5 - 4s_6 + s_7 = 0,$$

with $s_7 \in \mathbb{Z}_+$. Making x_5 basic in this constraint yields, for the objective function,

$$0s_6 - s_7,$$

i.e. no variables have a positive reduced cost which means that the present solution cannot be improved. The last iterations are presented in Figure 1.14.

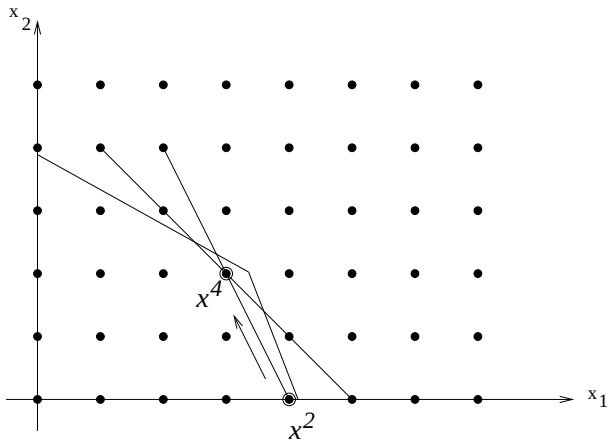

Figure 1.14: The move to the optimal solution by the Gomory-Young algorithm

Based on this example, we are now able to present the Gomory-Young algorithm. It is shown in Table 1.3. We suppose that we are given a feasible integer solution x^0. It is always possible to find an all integer simplex tableau that represents this solution, even if it is sometimes necessary to add some variables to the representation in order to be able to represent x^0. This is the starting point of the algorithm. We also introduce the following notation: $\mathcal{T}$ is a simplex tableau with a basic variable for each constraint, reduced costs for each variable and nonnegative right hand sides for each constraint. We denote by $\bar{a}_{ij}$ the value of the coefficient of variable j in row i, by $\bar{c}_j$ the reduced cost of variable j and by $\bar{b}_i$ the right hand side of row i in the tableau $\mathcal{T}$.

(1) Input: x^0 feasible, $\mathcal{T}^0$ all-integer simplex tableau representing x^0, $i = 1$
(2) **While** $\mathcal{T}^{i-1}$ is not optimal **do**
(3) **Choose** a variable x^k with $\bar{c}(x^k) > 0$
(4) **Set** $t := \arg\min \{\frac{\bar{b}_j}{\bar{a}_{jk}} : \text{for all rows } j \text{ s.t. } \bar{a}_{jk} > 0\}$
(5) **Generate** the Gomory cut $\mathcal{C}_i : \sum_{j \in N} \lfloor \frac{\bar{a}_{tj}}{\bar{a}_{tk}} \rfloor x_j + s_i = \lfloor \frac{b_t}{\bar{a}_{tk}} \rfloor$
(6) **Add** $\mathcal{C}_i$ to $\mathcal{T}^{i-1}$ and make a new tableau $\mathcal{T}^i$ by making x_k basic in $\mathcal{C}_i$
(7) **Set** $i := i + 1$
(8) **End**
(9) **Return** the solution given by $\mathcal{T}^i$

Table 1.3: The Gomory-Young algorithm

This algorithm has some drawbacks. First it adds one new variable and one new constraint at each iteration. At some point, the size of the tableau

becomes very large. Another drawback is that it often needs to add a big number of degenerate rows (i.e. with a 0 right hand side) before being able to move to another feasible point. In other words, the algorithm is often stuck in some feasible point for a big number of iterations. Finally the algorithm lacks flexibility in the sense that imposing the complete integrality in the tableau only allows us to add weak Gomory cuts which also explains that the algorithm progresses slowly.

Recently Haus, Köppe and Weismantel have proposed an improvement that keeps the primal character of the algorithm. The idea is to avoid adding weak Gomory cuts and replace the step by the addition of new variables. As before we present the algorithm on the case of Problem 1.1. The first iteration is the same as in the Gomory-Young algorithm because in this case, an augmentation has been found. We obtain thus after Iteration 1, the tableau

$$\begin{array}{rrrrrrr} \max & 3x_2 & & & - & 2s_3 + 8 & \\ \text{s.t.} & 9x_2 & +s_1 & & - & 5s_3 = 15 & \\ & 4x_2 & & +s_2 & - & 11s_3 = 1 & (1.21) \\ & x_1 & & & + & s_3 = 4 & \\ & x_1, & x_2, & s_1, \quad s_2, & & s_3 \in \mathbb{Z}_+. & \end{array}$$

The variable x_2 has a positive reduced cost and it is therefore interesting to try to increase its value. However the tight constraint (1.21) is such that the right hand side is smaller than the coefficient of x_2. A normal Gomory-Young iteration would add a degenerate row. It means that no move would be performed. Instead, Haus, Köppe and Weismantel try to characterize the primitive solutions of (1.21). As $s_2 \geq 0$, it implies that

$$4x_2 - 11s_3 \leq 1. \tag{1.22}$$

Some valid integer solutions of (1.22) are, for example,

$$\begin{pmatrix} x_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 9 \\ 5 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \dots$$

However if we consider the solutions

$$\begin{pmatrix} x_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \tag{1.23}$$

all the integer solutions of (1.22) can be expressed as nonnegative integer combinations of the vectors (1.23). For example,

$$\begin{pmatrix} 9 \\ 5 \end{pmatrix} = 3 \begin{pmatrix} 3 \\ 1 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

We say that the vectors (1.23) form an Hilbert basis of the solutions of (1.22). The vectors (1.23) are also called *irreducibles* because they cannot be written as a nonnegative integer combination of other solutions.

The idea of the algorithm is to associate new variables y_0, y_1, y_2, y_3 with each vector of the Hilbert basis. For example a variable y_2 is associated to $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$. It means that when $y_2 = 1$, x_2 is increased by 2 and s_3 by 1. The coefficients of the variable y_2 in the simplex tableau are therefore equal to twice the coefficients of x_2 added to the coefficients of s_3. The tableau (1.21) can now be written

$$\begin{array}{rrrrrrrrrr} \max & & & -2y_0 & + y_1 & + 4y_2 & + 7y_3 & + 8 & & \\ \text{s.t.} & s_1 & & -5y_0 & +4y_1 & +13y_2 & +22y_3 & = 15 & & (1.24) \\ & & s_2 & -11y_0 & -7y_1 & - 3y_2 & + y_3 & = 1 & & \\ & x_1 & & + y_0 & + y_1 & + y_2 & + y_3 & = 4 & & \\ & x_1, & s_1, & s_2, & y_0, & y_1, & y_2, & y_3 & \in \mathbb{Z}_+, & \end{array}$$

with $x_2 = y_1+2y_2+3y_3$ and $s_3 = y_0+y_1+y_2+y_3$. In this new formulation, three variables have a positive reduced cost: y_1, y_2, y_3. Among these three variables y_1 and y_2 have coefficients less than or equal to the right hand side in every row. We say that these variables are *augmenting directions*. In other words, we are able to find a new better feasible solution by increasing the value of one of these variables. Since y_2 has the larger reduced cost, we try to augment this variable. The tight constraint for y_2 is (1.24) and the Gomory cut generated from the division of (1.24) by the coefficient of y_2 is

$$-y_0 + y_2 + y_3 + s_4 = 1,$$

where $s_4 \in \mathbb{Z}_+$ represents the slack variable. We add this constraint to the tableau and make y_2 basic in order to give it a nonzero value. We now have

$$\begin{array}{rrrrrrrrrrr} \max & & & + 2y_0 & + y_1 & & +3y_3 & - 4s_4 & + 12 & & \\ \text{s.t.} & +s_1 & & + 8y_0 & +4y_1 & & +9y_3 & -13s_4 & = 2 & & (1.25) \\ & & +s_2 & -14y_0 & -7y_1 & & +4y_3 & + 3s_4 & = 4 & & \\ & x_1 & & + 2y_0 & + y_1 & & & - s_4 & = 3 & & \\ & & & - y_0 & & +y_2 & + y_3 & + s_4 & = 1 & & \\ & x_1, & s_1, & s_2, & y_0, & y_1, & y_2, & y_3, & s_4 & \in \mathbb{Z}_+. & \end{array}$$

The solution represented by this tableau is $(x_1, y_2, s_1, s_2) = (3, 1, 2, 4)$. As y_2 represents $2x_2 + s_3$, the solution can be expressed in terms of the original variables by $(x_1, x_2, s_1, s_2, s_3) = (3, 2, 2, 4, 1)$. Let us remark that we have reached the optimal solution. However since there remain some variables with a positive reduced cost, the optimality of the solution is not yet proved. A geometric

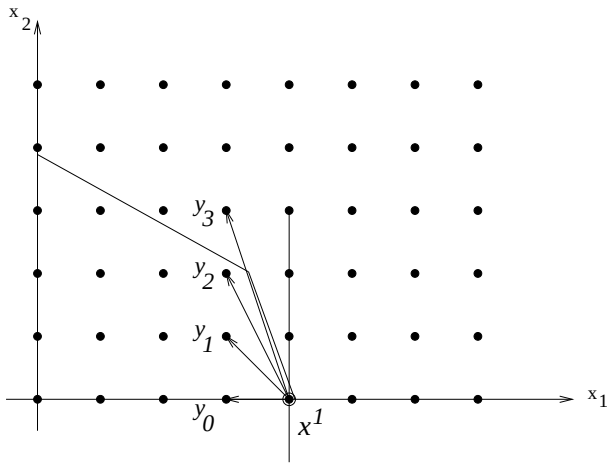

Figure 1.15: Directions computed via irreducible solutions

view of the iteration is represented in Figure 1.15. The four proposed directions y_0, y_1, y_2, y_3 are the integer feasible directions that keep the constraint (1.22) feasible. As we express the variables as nonnegative integer combinations of these directions, in the linear relaxation the original variables must be included in the cone determined by the directions. Therefore, in the space of original variables, we intersect the polyhedron with the cone C indicated in Figure 1.16. This cone C is the set of nonnegative combinations of the directions y_0, y_1, y_2, y_3. However we consider the problem in a higher dimension which

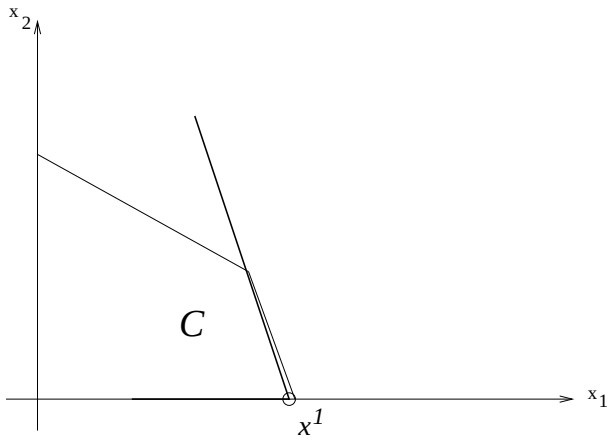

Figure 1.16: In the original space, the polyhedron is intersected with a cone

provides a richer structure that cannot be shown on a picture. For example in this case, it allows us to move directly to another point even if it is not a vertex in the original space.

Let us now go back to the tableau produced by Iteration 2. As said before there remain variables with a positive reduced cost. Before going on with the generation of new variables, an important step is to compactify the tableau, i.e. to get rid of the variables that are useless for the formulation. How can we know that a variable is useless? When, for example, the maximal value that it can take in any real feasible solution is less or equal to 1. The maximal value of y_3 can be computed by considering the linear programming

$$\begin{array}{rlllllllll}
\max & & & & & & & y_3 & & \\
\text{s.t.} & +s_1 & & +\ 8y_0 & +4y_1 & & & +9y_3 & -13s_4 & = 2 \\
 & & +s_2 & -14y_0 & -7y_1 & & & +4y_3 & +\ 3s_4 & = 4 \\
x_1 & & & +\ 2y_0 & +\ y_1 & & & & -\ \ s_4 & = 3 \\
 & & & -\ \ y_0 & & +y_2 & + & y_3 & +\ \ s_4 & = 3 \\
x_1, & s_1, & s_2, & y_0, & y_1, & y_2, & & y_3, & s_4 & \in \mathbb{R}_+.
\end{array}$$

The value of this linear program is 1.3. It means that $y_3 \leq 1$ in every integer solution of Problem 1.1. The same kind of computation can be done for the other variables in order to get bounds as tight as possible. In the case where one obtains a bound u_i for a variable x_i such that $u_i < 1$, the variable x_i can be removed from the formulation. In our case, we find

$$y_0 \leq 4, \quad y_1 \leq 3, \quad y_3 \leq 1, \quad s_4 \leq 5. \tag{1.26}$$

We can now proceed to generate new variables. We consider the tight row (1.25) for y_3 and write

$$8y_0 + 4y_1 + 9y_3 - 13s_4 \leq 2. \tag{1.27}$$

Instead of adding a Gomory cut, we try to find the irreducible solutions of (1.27). We obtain, corresponding to the variables $(y_0\ y_1\ y_3\ s_4)^T$, the following matrix of irreducibles

$$\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 0 & 2 & 0 & 3 & 1 & 3 & 4 & 0 & 2 & 1 & 0 & 5 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 2 & 1 & 0 & 0 & 0 & 3 & 3 & 5 & 0 & 0 & 7 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 3 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 2 & 2 & 3 & 1 & 2 & 2 & 2 & 3 & 2
\end{pmatrix}.$$

Using the bounds (1.26) computed earlier allows us to reduce the number of irreducibles of (1.27). Indeed the vectors

$$\begin{pmatrix}
1 & 1 & 0 & 5 & 0 \\
0 & 5 & 0 & 0 & 7 \\
2 & 0 & 3 & 0 & 0 \\
2 & 2 & 2 & 3 & 2
\end{pmatrix}$$

violate one of the bounds of the nonbasic variables. We can therefore get rid of them in the new formulation. We then consider a new variable for each relevant irreducible solution and reformulate the problem using these new variables. A simple check of the 13 new variables shows that they all have a nonpositive reduced cost. Therefore no variable has an interesting reduced cost, which proves that the current feasible solution is optimal.

This example indicates all the major ingredients of the Integral Basis Method that we now present in a general context. Suppose that we want to solve the integer program $\max\{cx : x \in X\}$ where $X = \{x \in \mathbb{Z}^n_+ : Ax = b\}$. As in the case of the Gomory-Young algorithm, we start with a feasible integer solution $x^0 \in \mathbb{Z}^n_+$ and a simplex tableau $\mathcal{T}^0$ that represents it. The Integral Basis Method algorithm is outlined in Table 1.4. Some steps need to be explained

(1) Input: x^0 feasible, $\mathcal{T}^0$ all-integer simplex tableau representing x^0, $i = 1$
(2) **While** $\mathcal{T}^{i-1}$ is not optimal **do**
(3) **Choose** a variable x^k with $\bar{c}(x^k) > 0$
(4) **Set** $t = \arg\min \{\frac{\bar{b}_j}{\bar{a}_{jk}} : \text{for all rows } j \text{ s.t.} \bar{a}_{jk} > 0\}$
(5) **If** $\bar{b}_t \geq \bar{a}_{tk}$ **then**
(6) **Generate** the Gomory cut $\mathcal{C}_i : \sum_{j \in N} \lfloor \frac{\bar{a}_{tj}}{\bar{a}_{tk}} \rfloor x_j + s_i = \lfloor \frac{b_t}{\bar{a}_{tk}} \rfloor$
(7) **Add** $\mathcal{C}_i$ to $\mathcal{T}^{i-1}$ and make a new tableau $\mathcal{T}^i$ by making x_k basic in $\mathcal{C}_i$
(8) **Else**
(9) **Consider** $R = \{x \in \mathbb{Z}^n_+ : \sum_{j \in S} \bar{a}_{tj} x_j \leq \bar{b}_t\}$, with $S \subseteq N$
(10) **Generate** a Hilbert basis H for the relaxation R
(11) **Create** a variable y_v for each $v \in H$
(12) **Set** $\mathcal{T}^i := \mathcal{T}^{i-1}$
(13) **Add** y_v to $\mathcal{T}^i$ for all $v \in H$, with coefficients $\bar{A}v$
(14) **Remove** the variables $j \in S$ from $\mathcal{T}^i$
(15) Compute the bounds on the variables and compactify $\mathcal{T}^i$
(16) **End**
(17) **Set** $i = i + 1$
(18) **End**

Table 1.4: The Integral Basis Method

in more detail. Steps (1)-(7) are the same as the steps of the Gomory-Young algorithm presented in Table 1.3 except from the fact that the Gomory cut generated in Step (6) is only added to the model when it leads to an improvement to the feasible solution. The relaxation considered in Step (9) often comes from considering one variable with a positive reduced cost along with variables with negative coefficients in the row. For Step (10), we need to define more precisely the concepts of Hilbert bases and of irreducible solutions.

1.2.7 The group approach

We now present another type of method introduced in 1969 by Gomory [19] and based on the modulo arithmetic and its structure of group. The method in itself is not applicable to integer problems of decent size. Nevertheless it is a tool that can be used within either a cutting plane approach or a primal method as introduced in the previous subsections. The topic of Chapter 4 is to show this possible combination of approaches.

As usual we start by solving Problem 1.1. For ease of presentation, we suppose that we start in the first iteration with a formulation on which we added the valid cut $x_1 + x_2 \leq 5$. The first step is to solve the linear relaxation. Therefore we find the optimal solution of

$$\begin{array}{rrrrrrrrr} \max & 2x_1 & +3x_2 & & & & & & \\ \text{s.t.} & 5x_1 & +9x_2 & +s_1 & & & & = & 35 \\ & 11x_1 & +4x_2 & & +s_2 & & & = & 45 \\ & x_1 & + & x_2 & & & +s_3 & = & 5 \\ & x_1, & & x_2, & s_1, & s_2, & s_3 & \in & \mathbb{R}_+. \end{array}$$

The optimal - fractional - tableau that we obtain at this stage is

$$\begin{array}{rrrrrrrrr} \max & & -\frac{1}{4}s_1 & & -\frac{3}{4}s_3 & & & & \\ \text{s.t.} & x_1 & -\frac{1}{4}s_1 & & +\frac{9}{4}s_3 & = & \frac{10}{4} & & (1.28) \\ & x_2 & +\frac{1}{4}s_1 & & -\frac{5}{4}s_3 & = & \frac{10}{4} & & (1.29) \\ & & \frac{7}{4}s_1 & +s_2 & -\frac{79}{4}s_3 & = & \frac{15}{2} & & (1.30) \\ & x_1,x_2, & & s_1, & s_2, & & s_3 & \in & \mathbb{Z}_+. \end{array}$$

The idea of the group relaxation is to relax the nonnegativity constraint on the basic variables x_1, x_2, s_2 but to keep the integrality constraint. In this respect, we can view, for example, (1.28) as

$$4x_1 - s_1 + 9s_3 = 10 \quad \text{with } x_1 \in \mathbb{Z} \text{ and } s_1, s_3 \in \mathbb{Z}_+. \tag{1.31}$$

All the solutions to (1.31) are also the solutions of

$$-s_1 + 9s_3 \equiv 10 \bmod 4,$$

that can be written as

$$3s_1 + s_3 \equiv 2 \bmod 4. \tag{1.32}$$

Similarly (1.29) can be viewed as

$$s_1 + 3s_3 \equiv 2 \bmod 4. \tag{1.33}$$

Remark here that, in the modulo arithmetic, (1.33) is equivalent to (1.32), being (1.32) multiplied by 3 (mod 4). We can verify that (1.30) is also equivalent,

when considering $s_2 \in \mathbb{Z}$, and in the modulo arithmetic to (1.28). Summarizing we now see that considering $x_1, x_2, s_2 \in \mathbb{Z}$ and $s_1, s_3 \in \mathbb{Z}_+$ requires to solve the problem

$$\max\{-\frac{1}{4}s_1 - \frac{3}{4}s_3 : 3s_1 + s_3 \equiv 2 \bmod 4, s_1, s_3 \in \mathbb{Z}_+\}. \tag{1.34}$$

It can be easily proved that the optimal solution of (1.34) is $s_1 = 2, s_3 = 0$. This corresponds, using (1.28),(1.29),(1.30) to the values

$$x_1 = 3, x_2 = 2, s_2 = 4$$

for the basic variables. In this case, all the basic variables have a nonnegative value which shows that we obtained the optimal solution. It is of course possible that the solution of the group problem leads to negative values for the basic variables. In that case, the optimal solution of the group problem provides an upper bound on the optimal integer solution. It is then also possible to add, to the problem, valid inequalities for the group problem. These inequalities are of course also valid for the original problem. Gomory proposes a complete study of the facets of the convex hull of the solutions of a group problem. He calls this convex hull a *corner polyhedron*. In Figure 1.17, one can see the two non-tight constraints determining the vertex $(x_1, x_2, s_2) = (\frac{10}{4}, \frac{10}{4}, \frac{15}{2})$. The corner polyhedron CP is determined by the convex hull of all the integer solutions included in the cone described by these two constraints. It is represented by bold lines on the figure. Remark that the negative points are also valid since we relax the nonnegativity constraint on the basic variables. The name "corner polyhedron" comes from the fact that it represents the facet description around one vertex i.e. one corner of the polyhedron. One can also represent the solutions of the group problem only in the space of the non-basic variables. It is called by Gomory the *t-space*. Such a picture is presented in Figure 1.18. The valid points are shown by circles and the corner polyhedron is represented by straight lines. There is, of course, a one-to-one correspondence between each circle of Figure 1.18 and the integer points of Figure 1.17.

In the complete study of the corner polyhedron, it appears that to obtain the facets of a corner polyhedron related to a group mod d, it suffices to study all the facets of the master polyhedron

$$x_1 + 2x_2 + \cdots + (d-1)x_{d-1} \equiv b \bmod d.$$

Gomory also showed that all the facets of corner polyhedra satisfy nice properties like subadditivity. We come back later in this thesis to these properties.

1.2.8 Lattices

To close this set of approaches to solve integer problems, we finally present a useful reformulation of integer programs using lattice basis reduction. This section cannot be considered as a complete method to solve integer programs

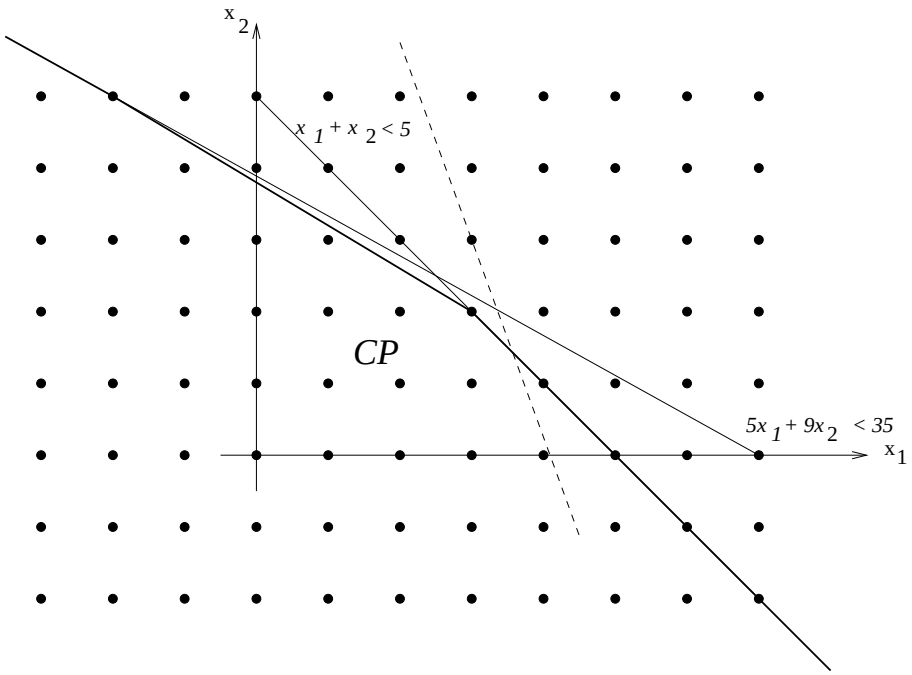

Figure 1.17: The Corner Polyhedron at the vertex $(\frac{10}{4}, \frac{10}{4})$

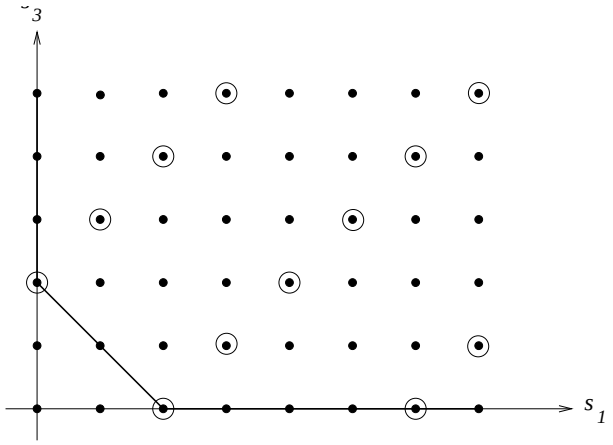

Figure 1.18: The Corner Polyhedron represented in the (s_1, s_3)-space, also called the t-space

but it provides a way to reformulate equality constrained problems. One of the previous methods has then to be used to actually solve the reformulation. However it can be proved that in some cases, the reformulation is a lot easier to solve than the original problem. Let us now define the basic tool of this approach.

Definition 1.12 *Let $A \in \mathbb{Z}^{m\times n}$. A lattice $\mathcal{L}(A)$ is the set of all integer combinations of the columns of A,*

$$\mathcal{L}(A) = \{x \in \mathbb{Z}^m : \exists y \in \mathbb{Z}^n \ s.t. \ x = Ay\}.$$

Several matrices define the same lattice. The relation between two matrices spanning the same lattice is *unimodular*.

Proposition 1.5 *Let $A, B \in \mathbb{Z}^{m\times n}$. The two following propositions are equivalent*

(i) $\mathcal{L}(A) = \mathcal{L}(B)$

(ii) There exists $C \in \mathbb{Z}^{n\times n}$ unimodular, i.e. $|det\ C| = 1$, such that $A = BC$.

The lattice is a natural object in the context of integer programming. Indeed if we look for $x \in \mathbb{Z}^n$ such that $Ax = b$, it can be equivalently expressed as checking whether b is in the lattice spanned by the columns of A. This problem is in fact slightly different from the problem we have seen up to now. In integer programming, we look for $x \in \mathbb{Z}^n_+$ such that $Ax = b$. Although it may look really similar, the slight difference, namely the nonnegativity requirements on x makes the problem much harder. In this context we cannot use the lattice concept in its pure form. However we will see that we can take advantage of the well posed relaxation $Ax = b$, with $x \in \mathbb{Z}^n$.

Proposition 1.5 shows that for the same lattice, it is possible to find several bases. It is quite intuitive that some bases are easier to handle than others. This is also the case with continuous subspaces. Within the bases spanning the same linear subspace, some are orthogonal which may be useful in some aspects. In the integer case, it is not always possible to find orthogonal bases of a lattice. However we can try to compute a basis with vectors as orthogonal as possible. Remark also that in the continuous case, the length of the vectors of a basis can always be chosen. Indeed if $v \in \mathbb{R}^n$ belongs to a basis, $\frac{v}{\lambda}$ with $\lambda \in \mathbb{R}\backslash\{0\}$ can replace v spanning the same linear subspace. In the integer case, no vector can be divided by a scalar different from 1 or -1 without changing the lattice. Therefore the lengths of the vectors in a basis are crucial and in general for practical matters, we always prefer to have vectors as short as possible. In this text, we focus on one definition of a good integer basis which is called a *reduced basis* in the Lovász sense [40]. The definition of Lovász leads to reduced bases that have the right properties, namely short vectors nearly orthogonal.

To define the notion of reduced bases of lattices, we first need to recall the Gram-Schmidt procedure that generates an orthogonal basis of a linear subspace.

Proposition 1.6 (Gram-Schmidt orthogonalization) *Let $(b_j)_{j=1,\dots,p} \subset \mathbb{R}^n$ be a basis of a linear subspace of $\mathbb{R}^n$. We define recursively*

$$\hat{b}_1 = b_1$$

$$\mu_{kj} = \frac{\langle b_j, \hat{b}_k \rangle}{\langle \hat{b}_k^T, \hat{b}_k \rangle} \qquad \text{for all } 1 \leq k < j \leq p \tag{1.35}$$

$$\hat{b}_j = b_j - \sum_{k=1}^{j-1} \mu_{kj} \hat{b}_k \qquad \text{for all } j = 2, \dots, p, \tag{1.36}$$

where $\langle x, y \rangle$ denotes the standard inner product of x and y. The set of vectors $(\hat{b}_j)_{j=1,\dots,p}$ spans the same linear subspace as $(b_j)_{j=1,\dots,p}$ and is orthogonal.

As we indicated earlier, this procedure cannot be applied in the integer case. Indeed it may happen that some μ_{kj} defined by (1.35) are not integer which implies that the operation (1.36) is not unimodular. Nevertheless the Gram-Schmidt vectors are a model to aim for. Furthermore they are included in the definition of a reduced basis.

Definition 1.13 *A basis $(b_j)_{j=1,\dots,p}$ of a lattice $\mathcal{L}$ of $\mathbb{Z}^n$ is reduced if*

$$(i) \quad |\mu_{jk}| \leq \frac{1}{2} \qquad \text{for all } 1 \leq j < k \leq p \tag{1.37}$$

$$(ii) \quad \|\hat{b}_j + \mu_{j-1,j} \hat{b}_{j-1}\|^2 \geq \frac{3}{4} \|\hat{b}_{j-1}\|^2 \qquad \text{for all } j > 1, \tag{1.38}$$

where μ_{jk} are the coefficients given by (1.35) *and $\hat{b}_j$ are the Gram-Schmidt vectors given by* (1.36).

We now explain the geometric meaning of these two conditions.

(i) The inequality (1.37) enforces the vectors of the basis to be nearly orthogonal and short. Remark that if $|\mu_{jk}| > \frac{1}{2}$, we can replace b_k by $b_k - \lfloor \mu_{jk} \rceil b_j$ and this would not change the Gram-Schmidt vectors but would make b_k closer to orthogonality with $\hat{b}_j$ i.e. with b_j.

(ii) Remark that if the first Gram-Schmidt vectors of the basis are shorter, then the first condition is stronger. Indeed $\mu_{jk} = \frac{\langle b_k, \hat{b}_j \rangle}{\langle \hat{b}_j^T, \hat{b}_j \rangle}$. If $\|\hat{b}_j\|$ is smaller and we keep $|\mu_{jk}| \leq \frac{1}{2}$, it is clear that b_k will be smaller. Therefore, it is always preferable to have shorter Gram-Schmidt vectors in the first elements of the basis. The second condition tests what is the influence on the length of the Gram-Schmidt vectors when exchanging two vectors of

the basis. The left-hand-side of (1.38) is the length of the new $(j-1)^{th}$ Gram-Schmidt vector if b_{j-1} and b_j are exchanged in the basis. The right-hand-side of (1.38) is the length of the current $(j-1)^{th}$ Gram-Schmidt vector. The second condition checks whether there is no substantial decrease in the length of the $(j-1)^{th}$ Gram-Schmidt vector if we exchange b_{j-1} and b_j in the basis.

Remark that the coefficient $\frac{3}{4}$ can be taken in $]\frac{1}{4}, 1[$, the condition (1.38) being stronger if the coefficient is larger. The best basis one could hope would be to ask that

$$\|\hat{b}_j + \mu_{j-1,j}\hat{b}_{j-1}\|^2 \geq \|\hat{b}_{j-1}\|^2 \qquad \text{for all } j > 1.$$

Unfortunately asking for such a condition would lead to a basis that is too hard to compute. If we keep a coefficient in $]\frac{1}{4}, 1[$, the reduced basis is computable in polynomial time [40]. If we ask for a condition (1.38) with a coefficient of 1, the algorithm to compute a reduced basis becomes exponential, and therefore too time-consuming.

This discussion about the reduced bases allows us to present the reformulation of integer programs that is based on lattice basis reduction. Recall that the major ingredient of this method is to suppose that we have an algorithm that given a lattice basis, computes a reduced basis of the same lattice in polynomial time, i.e. a basis satisfying the two conditions of Definition 1.13.

Suppose that we want to find feasible solutions to

$$\begin{array}{l} Ax = b \\ x \in \mathbb{Z}^n_+, \end{array} \tag{1.39}$$

with $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$. We now explain a method to reformulate (1.39) using lattice basis reduction. We have explained that finding feasible solutions to

$$\begin{array}{l} Ax = b \\ x \in \mathbb{Z}^n \end{array} \tag{1.40}$$

is polynomially solvable. The goal is to construct a basis of the integer null space

$$\begin{array}{l} Ax = 0 \\ x \in \mathbb{Z}^n, \end{array} \tag{1.41}$$

and a particular solution of (1.40). Indeed any solution x of (1.40) can be written in the form $x = q + \sum_i \lambda^i p^i$, with $\lambda^i \in \mathbb{Z}$ if q is a particular solution of (1.40) and P is an integer basis of the null space (1.41), p^i being its columns. We make use of a lattice basis reduction to do this. The idea of the method of Aardal, Hurkens and Lenstra [2] is to construct the lattice of integer points

$$(x_1, \ldots, x_n, N_1 t, N_2(Ax - bt)), \tag{1.42}$$

with N_1, N_2 large integers. This is a lattice generated by the columns of

$$\begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & N_1 \\ N_2 A & & & -N_2 b \end{pmatrix}. \tag{1.43}$$

A vector y of the form (1.42) with $y_{n+1} = 0$ and $y_{n+2} = 0$ is such that $y_{1:n}$ is a homogeneous solution of (1.40), i.e. a solution of (1.41). A vector y of the form (1.42) with $y_{n+1} = N_1$ and $y_{n+2} = 0$ is such that $y_{1:n}$ is a solution of (1.40). We reduce the basis (1.43). Since a reduced basis tends to have short vectors, it can be shown that if N_1 and N_2 are sufficiently large, we obtain a reduced basis of the form

$$\begin{pmatrix} P & q & \times \\ 0 & N_1 & \times \\ 0 & 0 & \times \end{pmatrix}. \tag{1.44}$$

In other words, reducing the basis yields a basis P of homogeneous solutions and a vector q particular solution of (1.40). Furthermore by the reduced basis properties, it can be shown that P and q are "short". Given P and q, we are now able to reformulate (1.39) as finding $\lambda \in \mathbb{Z}^{n-m}$ such that

$$0 \leq q + \sum_{i=1}^{n-m} \lambda^i p^i. \tag{1.45}$$

In some cases, the formulation (1.45) is easier to solve with a traditional method like branch-and-bound than the original formulation (1.39). We show an example of this in Chapter 5 and discuss an improvement of the method on a structured problem.

1.3 Outline of the thesis

We now come to an outline of the thesis. For each subject, we also review the main references in the literature and present more formally what are the main questions that are raised. First, let us mention that the four chapters of this thesis are based on three papers. Chapters 2 and 3 are based on [45]. Chapter 4 presents the result of the report [37]. Finally Chapter 5 is based on the article [44].

In Chapter 2 we discuss an important problem when trying to generate strong valid inequalities for the branch-and-cut algorithm: the lifting problem. The lifting procedure consists of the generation of valid inequalities for a set X from valid inequalities that have been generated in a lower dimensional set

$Y \subset X$. The basic model that we choose is the following. We study the set Z defined as the set of solutions of

$$\sum_{k=1}^{\tau} A^k z^k \leq b + s$$
$$z^k \in Z^k, s \in \mathbb{R}^m_+,\ k = 1, \ldots, \tau.$$

The goal is to find valid inequalities for Z, but as Z may be too complicated, it is useful to consider a restriction of Z by setting the variables of each block $Z^k, k = 2, \ldots, \tau$ to a fixed value. This simplifies the model and it is therefore easier to derive a valid inequality for

$$\begin{aligned} A^1 z^1 &\leq b + s \\ z^1 &\in Z^1, s \in \mathbb{R}^m_+. \end{aligned} \tag{1.46}$$

Let $\pi^1 z^1 \leq \mu + \nu s$ be such a valid inequality. This inequality is not necessarily valid for Z. The next step is thus to "lift" in the variables in the block Z^2, then $Z^3, \ldots$ By lifting we mean finding coefficients π^2 such that $\pi^1 z^1 + \pi^2 z^2 \leq \mu + \nu s$ is valid for

$$A^1 z^1 + A^2 z^2 \leq b + s$$
$$z^1 \in Z^1, z^2 \in Z^2, s \in \mathbb{R}^m_+.$$

Lifting was first used by Gomory [19] in the group framework. Lifting in itself has been defined by Padberg [54] in the early 1970's and extended in [63]. In [64] the crucial link between lifting and superadditivity was raised. In [26] the authors propose a method to lift more than one variable at a time, in the case of mixed-integer programming. Lifting has also been used to derive interesting theoretical results like [7, 46].

In Chapter 2, we review the basic concepts of lifting and try to give a simplified presentation of the method. Two main simplifications are made in the procedure to make it easy to present. First we assume that all the variables fixed during the procedure are fixed to zero. This does not look completely general but can be done up to a substitution of variables. The other simplification is to keep the continuous variables s in the model (1.46). This restricts the generality of the method. However it allows us to make convenient assumptions of existence and continuity of the crucial lifting function. The procedure described can still be applied without continuous variables s but in that case, we no longer have the guarantee that the lifting can be carried out in general. In the chapter, we also raise the question of lifting several blocks of variables sequentially. The computation may be greatly simplified when the lifting function is *superadditive*. We introduce the notion and show how it is used in lifting. This last topic is related to the results of [64] and [26].

In Chapter 3 we deal with an important set in mixed integer programming, the single node flow set, denoted $X^N(n_1, n_2, b, a, u) =$

$$\begin{gathered}\sum_{j\in N_1} x_j - \sum_{j\in N_2} x_j - s \leq b\\ x_j \leq a_j y_j \text{ for } j \in N_1 \cup N_2\\ y_j \leq u_j \text{ for } j \in N_1 \cup N_2\\ x \in \mathbb{R}_+^{n_1+n_2}, y \in \mathbb{Z}_+^{n_1+n_2}, s \in \mathbb{R}_+^1,\end{gathered}$$

where $n_1 = |N^1|, n_2 = |N^2|, n = n_1 + n_2, b \in \mathbb{Z}^1, a \in \mathbb{R}^n, u \in \mathbb{Z}_+^n$. This set arises as a subproblem of a fixed charge network flow problem. Furthermore it can be shown that a general mixed integer row can be represented in the form X^N. The first polyhedral results for X^N were derived by Padberg, Van Roy and Wolsey [55] in 1985 introducing the so called flow cover inequalities. Van Roy and Wolsey [58] later showed how to use the flow cover inequalities computationally. Stallaert [57] extended the flow cover inequalities to a generalization obtained by reversing the arcs of the network and obtained the reverse flow cover inequalities. Other results on related sets were obtained in [5, 17, 48, 66, 33]. In [27], Gu, Nemhauser and Savelsbergh showed how to lift pairs of variables (x_k, y_k) in flow cover inequalities. The viewpoint taken in this text in Chapter 3 is closed to theirs, to that of Marchand and Wolsey [46] and of Atamtürk [6]. The idea is to show how to derive valid inequalities from a general procedure. This procedure works as follows. We first partition the variables into different sets and for each set we either fix its variables to a certain value or leave them in the model. We obtain then a simpler model to which we apply the basic MIR inequality introduced in Proposition 1.4. Then we lift the remaining variables and obtain a valid inequality for the set X^N. The family of valid inequalities that we obtain is closely related to the flow cover inequalities. Our model seems, therefore, to show that an MIR operation combined with lifting are the simplest operations needed to obtain strong valid inequalities for the single node flow set. We apply our procedure to both the 0-1 and the general integer single node flow sets. We also show how superadditive lifting can be used in order to strengthen the obtained inequalities.

From Chapter 4 on, we study non standard approaches to integer programming. In Chapter 4, we show how the group relaxation approach can be used in a reformulation framework. The group relaxation has been first introduced by Gomory in 1969 [19]. Let us recall briefly the context, already presented in Section 1.2.7. We start with an integer linear program

$$\begin{array}{ll}\max & cx\\ \text{s.t.} & Ax = b\\ & x \in \mathbb{Z}_+^n.\end{array} \tag{1.47}$$

Choosing a subset B of the variables as a basis, we can write (1.47) at a vertex

of the corresponding linear relaxation polyhedron. It gives

$$\begin{array}{ll} \max & \bar{c}_N x_N \\ \text{s.t.} & x_B + \bar{A}_N x_N = \bar{b} \\ & x \in \mathbb{Z}^n_+. \end{array} \tag{1.48}$$

Now we relax the nonnegativity constraints on x_B and this produces the relaxation for (1.47)

$$Y(\bar{b}) = \{\bar{A}_N x_N \equiv \bar{b} \,(\text{mod } 1) | x_N \in \mathbb{Z}_+^{|N|}\}. \tag{1.49}$$

The set (1.49) is called a group problem. From this relaxation, one can try to derive valid inequalities for (1.47). In [19] and [20] for the mixed case, they investigate the structure of the facets of $\text{conv}(Y(\bar{b}))$. They provide a complete implicit description of it. Although the theory of the group approach was rather complete, little computational work was reported using this approach. The main study was that of Gorry, Northup and Shapiro [24]. They developed a branch-and-bound code in which the bounds were obtained by solving relaxed group problems by dynamic programming. The major obstacle encountered was the huge size of the groups arising in practice (10^8 to 10^{10}) while the group problems that could be solved were of the order of 10^4. The dynamic programming methods for the group problems were reported by Wolsey in [61]. More recently theoretical and empirical results on the facets of corner polyhedra were published [4, 16, 23].

The main objective of Chapter 4 is to study extended formulations of a group relaxation instead of the description of the convex hull by linear inequalities. This opens up the possibility to use a group reformulation in any framework using reformulations by variables. In the chapter we first study four possible ways of reformulating the group relaxation. Three among the four relaxations use the concept of irreducible solutions of a system of equations. The difference between them lies in the possibility to aggregate variables in the original problem in order to reduce the number of new variables in the reformulation. A fourth way to reformulate is presented in which the path representation of the group problem is used. This is related to [61]. In a next step we study the strength of these different reformulations. It is shown that two formulations using irreducible solutions and the path reformulation are as tight as possible in the sense that the projection of these formulations onto the space of original variables give the convex hull of the original problem. This property is not true for the formulation using the most complicated aggregation of the variables. Nevertheless we can strengthen it by proposing a series of linear inequalities coming from the polyhedral study of another corner polyhedron. In the chapter we also show how the irreducible solutions of a group problem can be computed. Finally some computational results are reported. Two algorithms are tested. A first algorithm simply computes a sequence of LP optimal points and reformulates at each step in order to come closer to the

integral optimal. The second algorithm is based on the primal method of Haus, Köppe and Weismantel [29] presented in Section 1.2.6. The idea is to use the group problem as a tool to generate new variables. We report on some limited computational results related to that.

In Chapter 5 we propose a study of the banker's problem, i.e. the financial form of the problem of sharing an attic. As we told earlier, this problem cannot be solved by the standard methods like branch-and-bound whereas a reformulation using lattice basis reduction makes the problem easy to solve. The banker's problem has an interesting structure and the reduced basis can become heavy to compute for large sizes of the input. The main focus of the chapter is to show that one can decompose the problem in order to compute the reduced basis. Furthermore we show that the fact to decompose the computation does not affect the reduced character of the final basis obtained. We close the section with some computational results showing the benefit of the reformulation and the decomposition.

The sixth chapter of the thesis is dedicated to a conclusion summarizing the main results and proposing directions for future research in the areas that were studied in the text.

Chapter 2

Lifting of valid inequalities

2.1 Introduction

In this chapter we study the question of lifting valid inequalities. The central line of this problem is to see how to generate valid inequalities for a set X from valid inequalities for a lower-dimensional restriction of X. The structure of the chapter is the following. In the first section, we introduce the notion of lifting. We try to capture the intuition of the operation on a simple two dimensional example. The next section presents our procedure of lifting in detail. We present all the assumptions that are needed to apply our procedure and present the theoretical results related to lifting. Section 4 is related to the difference of lifting several blocks of variables sequentially or simultaneously. However in some particular case, the two approaches coincide. That is the case when the lifting function is superadditive. Finally we conclude by discussing some variations of our hypotheses. Remark that this chapter is inspired by Sections 1 to 3 of [45].

2.2 The Context of Lifting: Generating Valid Inequalities from Sub or Supersets

Throughout we consider mixed integer sets described by integer (or rational) coefficients of the form

$$X = \{z \in \mathbb{R}_+^{n^1} \times \mathbb{Z}_+^{n^2} : Az \leq b\}$$

with $n = n^1 + n^2$. Here we also consider a second lower-dimensional subset

$$Y = X \cap \{z : Cz = e\}.$$

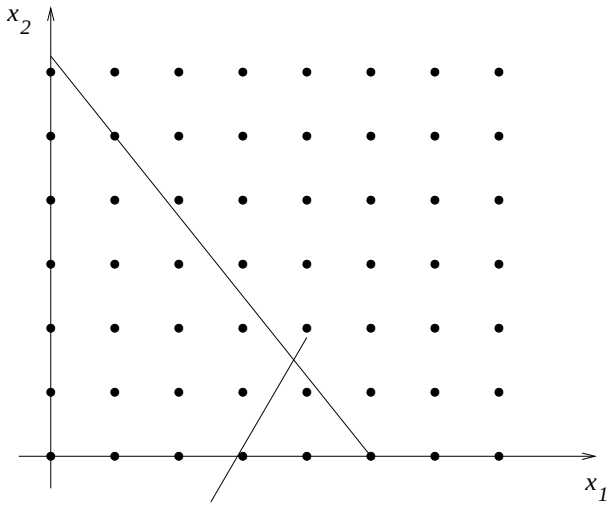

Figure 2.1: The set X

2.2.1 Valid Inequalities from Supersets

Observation 2.1 *If $Cz \leq e$ for all $z \in X$, then* $\text{conv}(Y)$ *is a face of* $\text{conv}(X)$*, and*

$$\text{conv}(Y) = \text{conv}(X) \cap \{z : Cz = e\}.$$

Thus every facet-defining inequality of $\text{conv}(Y)$ *corresponds to a facet-defining inequality of* $\text{conv}(X)$*.*

Example: Consider the simple set

$$X = \{x \in \mathbb{Z}^2_+ : 12x_1 - 7x_2 \leq 35,\ 5x_1 + 4x_2 \leq 25\}.$$

The set is represented in Figure 2.1. The convex hull of the set is shown in Figure 2.2. We have

$$\text{conv}(X) = \{x \in \mathbb{R}^2_+ : x_1 + x_2 \leq 6, 3x_1 + 2x_2 \leq 13, x_1 \leq 3, x_1 - x_2 \leq 2\}.$$

Now consider the set $Y = X \cap \{x_1 = 3\}$. First observe that $Y = \{(3,1),(3,2)\}$. Then we see that $x_1 \leq 3$ holds for every valid point of X. Therefore, $\text{conv}(Y)$ is a face of $\text{conv}(X)$ and the conditions of Observation 2.1 are satisfied. Therefore the facet-defining inequalities of $\text{conv}(Y)$ are given by replacing x_1 by 3 in every facet-defining inequality of $\text{conv}(X)$. Doing this we obtain that

$$\text{conv}(Y) = \{x \in \mathbb{R}^2_+ : x_2 \leq 3,\ x_2 \leq 2,\ x_2 \geq 1\} \cap \{x_1 = 3\}.$$

We see in Figure 2.2 that the inequalities $x_2 \leq 2$ and $x_2 \geq 1$ give the complete description of $\text{conv}(Y)$ besides the trivial equality $x_1 = 3$.

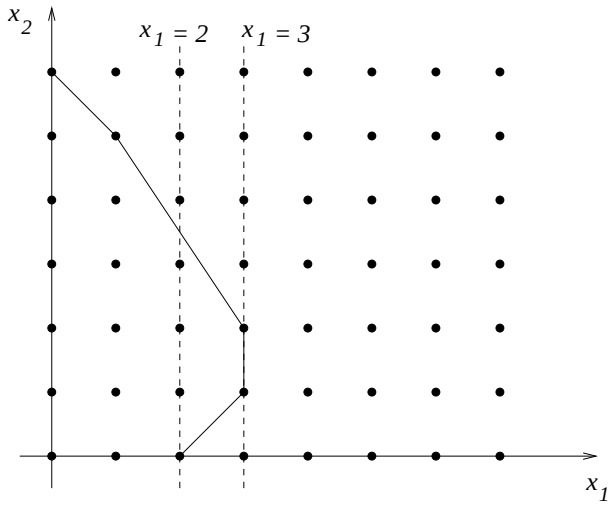

Figure 2.2: conv(X) and its two restrictions

On the other hand, we do not always obtain the convex hull if we consider a restriction of X that is cut in the middle of conv(X). Consider, for example, $Z = X \cap \{x_1 = 2\}$. Checking in Figure 2.2, we see that $Z = \{(2,0),(2,1),(2,2),(2,3)\}$. If we replace x_1 by 2 in the inequalities describing conv(X), we obtain

$$x_2 \leq 4,\ x_2 \leq \frac{7}{2},\ x_2 \geq 0.$$

This set of inequalities does not provide the complete description of conv(Z) although all three inequalities are valid. □

This example is really basic but this observation can be applied to more complicated sets. For example, if we consider the so called single node flow set (which we discuss in Chapter 3 in detail)

$$\begin{array}{rl} X^N(n_1,n_2,b,a) = \{ & (x,y,s) \in \mathbb{R}_+^{n_1+n_2} \times \{0,1\}^{n_1+n_2} \times \mathbb{R}_+ : \\ & \sum_{j\in N_1} x_j - \sum_{j\in N_2} x_j \leq b+s \\ & x_j \leq a_j y_j \text{ for } j \in N_1 \cup N_2 \quad \} \end{array}$$

and the continuous knapsack

$$Y^{CK} = \{(y,s) \in \{0,1\}^{n_1+n_2} \times \mathbb{R}_+ : \sum_{j\in N_1} a_j y_j - \sum_{j\in N_2} a_j y_j \leq b+s\},$$

we remark that Y^{CK} is obtained from X^N by setting $x_i = a_i y_i$. Obviously by the definition of X^N, $x_i \leq a_i y_i$ hold for all i for the valid points in X^N.

Therefore all the facets of $\text{conv}(Y^{CK})$ can be obtained by setting $x_i = a_i y_i$ in all the facets of $\text{conv}(X^N)$.

Example: The facet

$$x_1 - y_1 + x_2 - y_2 + x_3 - y_3 - y_4 - x_5 \leq 3 + s \tag{2.1}$$

of the convex hull of the single node flow set

$$\begin{aligned} X^N(3,2,4,(3,4,5,2,3)) = \{ \quad & (x,y,s) \in \mathbb{R}^5_+ \times \{0,1\}^5 \times \mathbb{R}_+ : \\ & x_1 + x_2 + x_3 - x_4 - x_5 \leq 4 + s \\ & x_1 \leq 3y_1,\ x_2 \leq 4y_2,\ x_3 \leq 5y_3,\ x_4 \leq 2y_4,\ x_5 \leq 3y_5\} \end{aligned}$$

leads to the valid inequality

$$2y_1 + 3y_2 + 4y_3 - y_4 - 3y_5 \leq 3 + s, \tag{2.2}$$

for the set

$$Y^{CK} = \{(y,s) \in \{0,1\}^5 \times \mathbb{R}_+ : 3y_1 + 4y_2 + 5y_3 - 2y_4 - 3y_5 \leq 4 + s\},$$

when one replaces x_i by $a_i y_i$ for all i in the facet (2.1). The inequality (2.2) is indeed facet-defining for $\text{conv}(Y^{CK})$. □

Another consequence of Observation 2.1 concerns the separation problem arising when one wishes to find a valid inequality for X cutting off a point z^*.

Observation 2.2 *If $Cz \leq e$ for all $z \in X$ and z^* satisfies $Cz^* = e$, then $z^* \notin \text{conv}(X)$ if and only if $z^* \notin \text{conv}(Y)$.*

Thus to solve the separation problem over X, it suffices to solve the separation problem over Y, and then convert the violated valid inequality for Y into a violated inequality for X. The advantage is that the set Y has a simpler structure since it is in a lower dimension and one can hope that the separation problem is easier to solve. The counterpart is that one needs to convert the violated inequality for Y to an inequality for X and this needs some computation. The conversion from the Y space to the X space is precisely the lifting problem that we now consider.

2.2.2 Valid Inequalities from Subsets

In the previous subsection, we were interested in generating valid inequalities for a small set from the knowledge of valid inequalities of supersets, i.e. more complicated sets. However usually the convex hull description of the larger set is not known and harder to derive.

The question of deriving valid inequalities for X from valid inequalities for $Y = X \cap \{z : Cz = e\}$ is called the "*lifting problem*". Again the fact that $X \subseteq \{z : Cz \leq e\}$ and thus $\text{conv}(Y)$ is a face of $\text{conv}(X)$ is crucial.

Observation 2.3 *If* $\text{conv}(Y)$ *is a face of* $\text{conv}(X)$ *and* $\pi^1 z \leq \pi_0$ *is a valid inequality for* $\text{conv}(Y)$*, there exists a vector* π^2 *such that*

$$\pi^1 z + \pi^2(e - Cz) \leq \pi_0$$

is valid for $\text{conv}(X)$*.*

It is easily verified that the inequality is valid if one takes $\pi^2_j = -M$ for all $j \in N$ with M sufficiently large.

Example: Let us go back to the previous example with

$$X = \{x \in \mathbb{Z}^2_+ : 12x_1 - 7x_2 \leq 35,\ 5x_1 + 4x_2 \leq 25\} \text{ and } Y = X \cap \{x_1 = 3\}.$$

In the last subsection, we used the facet description of $\text{conv}(X)$ to derive the facet-description of $\text{conv}(Y)$. Observation 2.3 states that it is possible, at least theoretically, to go the other way, i.e. to use the facet-defining inequalities of $\text{conv}(Y)$ to derive valid inequalities of $\text{conv}(X)$. Let us see what happens geometrically. The sets $\text{conv}(X)$ and $\text{conv}(Y)$ are represented in Figure 2.3. The dotted area is $\text{conv}(X)$ while the vertical bold line is $\text{conv}(Y)$. It is shown

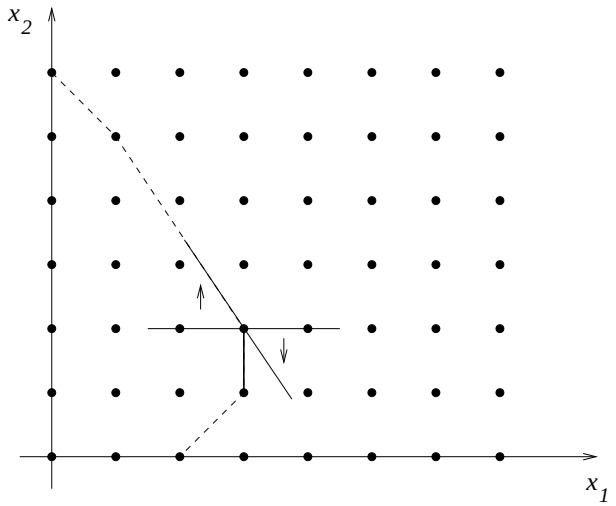

Figure 2.3: A facet of $\text{conv}(Y)$ can be *lifted* to a valid inequality of X

in the figure that the facet-defining inequality $x_2 \leq 2$ for $\text{conv}(Y)$ can be lifted to a valid inequality

$$\pi(3 - x_1) + x_2 \leq 2$$

for X, by finding the right coefficient π. Geometrically the problem is to rotate the line $x_2 \leq 2$ around the point $(3, 2)$ until a valid inequality is found. The picture shows that in this case, it is clearly possible. □

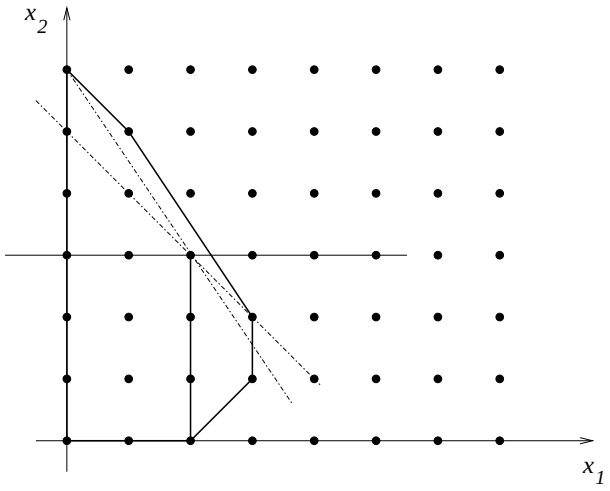

Figure 2.4: When Y_2 is not a face of X, the lifting problem may be impossible

On the other hand, when $Cz = e$ cuts through the interior of conv(X), it is known that the required multipliers may not exist. This is demonstrated by the following simple example.

Example: We consider again

$$X = \{x \in \mathbb{Z}^2_+ : 12x_1 - 7x_2 \leq 35,\ 5x_1 + 4x_2 \leq 25\}$$

and $Z = X \cap \{x_1 = 2\}$. A facet-defining inequality for conv(Z) is $x_2 \leq 3$. Now if $\pi(2 - x_1) + x_2 \leq 3$ is a valid inequality for X, the point $(3, 2) \in X$ implies $\pi \geq -1$ whereas the point $(0, 6) \in X$ implies $\pi \leq -\frac{3}{2}$. Both conditions are obviously disjoint. Therefore the lifting is impossible. Geometrically the problem is to find a rotation of the line $x_2 \leq 3$ around the point $(2, 3)$ which leads to a valid inequality for X. This is impossible as Figure 2.4 shows. □

In the next section we consider how to find the multipliers π^2 or show that there are none.

2.3 Lifting Valid Inequalities

2.3.1 Lifting: Basic Method

We consider the mixed integer sets, denoted $Z^\tau(b)$, of the form

$$\begin{gathered} \textstyle\sum_{k=1}^{\tau} A^k z^k \leq b + s \\ z^k \in X^k \text{ for } k = 1, \ldots, \tau, s \in \mathbb{R}^m_+, \end{gathered} \tag{2.3}$$

where $A^k \in \mathbb{R}^{m \times n_k}$ for $k = 1, \ldots, K, b \in \mathbb{R}^m$, $X^k = \{z^k \in \mathbb{R}^{n_k^1} \times \mathbb{Z}^{n_k^2} : C^k z^k \leq c^k\}$ with $n_k = n_k^1 + n_k^2$ is a mixed integer set in $\mathbb{R}^{n_k}$ for all k and $0 \in X^k$ for $k = 2, \ldots, K$. Here we study how to find valid inequalities for $Z^K(b)$, starting from valid inequalities for $Z^1(b)$. The lifting approach consists of the following:

1. Fix $z^k = 0$ for $k = 2, \ldots, K$.
2. Find a tight valid inequality $\pi^1 z^1 \leq \pi_0 + \nu s$ for $Z^1(b)$.
3. Iterations $\tau = 2, \ldots, K$. Given a tight valid inequality $\sum_{k=1}^{\tau-1} \pi^k z^k \leq \pi_0 + \nu s$ for $Z^{\tau-1}(b)$, lift the variables z^τ and derive coefficients $\pi^\tau \in \mathbb{R}^{n_k}$ such that

$$\sum_{k=1}^{\tau-1} \pi^k z^k + \pi^\tau z^\tau \leq \pi_0 + \nu s \tag{2.4}$$

   is valid for $Z^\tau(b)$, or determine that no such π^τ exists.

Each of these steps needs some comments. Let us take them one by one.

1. As it appears in the introduction of this chapter, we do not only want to restrict our procedure to fixing variables to 0 because it is often useful to consider setting inequality $Cx \leq e$ to equality $Cx = e$ as we proposed earlier. Nevertheless, by a simple substitution, it is always possible to equivalently fix some other variables to 0. Indeed if we consider the slack variable $\hat{x} = Cx - e$, fixing $\hat{x} = 0$ is equivalent to fixing $Cx = e$. We show some more examples of this fact later in the chapter.
   Relative to the description in Section 2.3.1, $X = Z^K(b)$ and $Y = Z^K(b) \cap \{(z^1, \ldots, z^K) : z^2 = \cdots = z^K = 0\} = Z^1(b)$.
2. In this chapter we do not focus on the way to find a valid inequality for $Z^1(b)$. But this relies on the fact that since the set $Z^1(b)$ is smaller, it is easier to find valid inequalities for $Z^1(b)$ than for the complete set. The next chapter focuses on generating valid inequalities for single node flow sets by lifting MIR inequalities.
3. The core of this chapter is to handle this step. We develop this topic in the next section.

2.3.2 Determining the lifting coefficient

When computing the lifting coefficient, a crucial concept is that of lifting function. It computes the slack remaining in the valid inequality when the variables fixed earlier at 0 take non zero values.

Definition 2.1 *The lifting function* $\phi^k : \mathbb{R}^m \to \mathbb{R}^1$ *is*

$$\phi^k(u) = \min\{\pi_0 + \nu s - \sum_{\tau=1}^{k} \pi^\tau z^\tau : (z^1, \ldots, z^k, s) \in Z^k(b - u)\}.$$

The following example illustrates the intuition of such a definition.

Example: As earlier we start with the set

$$X = \{x \in \mathbb{Z}^2_+ : 12x_1 - 7x_2 \leq 35,\ 5x_1 + 4x_2 \leq 25\},$$

represented in Figure 2.1, and $Y = X \cap \{x_1 = 3\}$. As we pointed before, $x_2 \leq 2$ is facet-defining for $\text{conv}(Y)$ and therefore we look for a coefficient π such that

$$\pi(3 - x_1) + x_2 \leq 2 \tag{2.5}$$

is valid for X.

Consider $\hat{x}_1 = 3 - x_1$. When $\hat{x}_1 = 0$, the set X is restricted to Y and (2.5) is always valid for X. The role of the lifting function is to study all the possible values of $\hat{x}_1$ and for each case, show which values of π are acceptable in order to obtain a valid inequality for X.

When $\hat{x}_1 = 1$, i.e. $x_1 = 2$, the two main constraints become

$$-7x_2 \leq 11 \quad \text{and} \quad 4x_2 \leq 15,$$

which basically means that $x_2 \leq 3$. Therefore when $\hat{x}_1 = 1$, the lifting function

$$\begin{aligned} \phi(1) = &\min\ 2 - x_2 \\ &\text{s.t.} \quad -7x_2 \leq 11 \\ &\qquad\ \ 4x_2 \leq 15 \\ &\qquad\ \ x_2 \in \mathbb{Z}_+ \end{aligned}$$

takes the value -1. This means that the slack in the inequality (2.5) can go down to -1 when $\hat{x}_1 = 1$. In other words, the inequality $\pi\hat{x}_1 \leq 2 - x_2$ has -1 as smallest right hand side when $\hat{x}_1 = 1$. Therefore a necessary condition on π is that $1\pi \leq -1$.

When $\hat{x}_1 = 2$, the two main constraints are

$$-7x_2 \leq 23 \quad \text{and} \quad 4x_2 \leq 20,$$

i.e. $x_2 \leq 5$. Therefore, $\phi(2) = \min\{2 - x_2 : 0 \leq x_2 \leq 5\} = -3$. This means that when $\hat{x}_1 = 2$, the slack in (2.5) can go down to -3. For this case, this implies the necessary condition on π : $2\pi \leq -3$.

Finally, when $\hat{x}_1 = 3$, we have $x_2 \leq 6$ and $\phi(3) = -4$. Therefore we also need that $3\pi \leq -4$. These four cases show that we have to take $\pi \leq -\frac{3}{2}$ in (2.5) in order to obtain a valid inequality for X. This can be seen geometrically in Figure 2.5. For each value of $\hat{x}_1$ (i.e. x_1), the lifting function computes the minimum slack obtained in the inequality $x_2 \leq 2$, which thus has to be rotated.

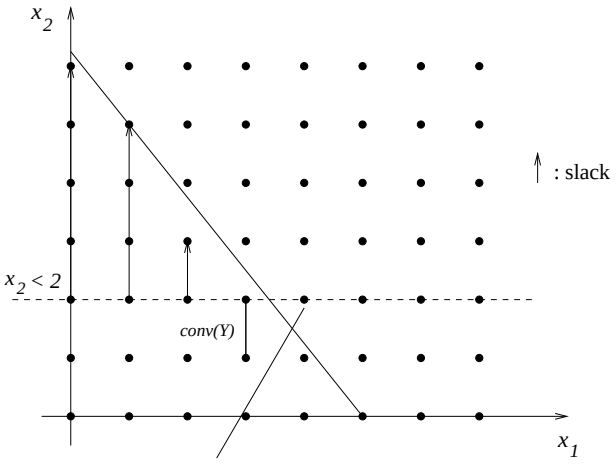

Figure 2.5: An illustration of what the lifting function computes

Each slack is shown in the figure and determines a minimum angle of rotation. If we now rotate the inequality using the coefficient $\pi = -\frac{3}{2}$, we obtain

$$3x_1 + 2x_2 \leq 13$$

which is one of the facet-defining inequalities of $\text{conv}(X)$. □

We now come back to the theory, based on Definition 2.1. Note that for any $u \in \mathbb{R}^m$ and any $(z^1, \ldots, z^k) \in X^1 \times \ldots \times X^k$, there exists $s \in \mathbb{R}^m_+$ such that $(z^1, \ldots, z^k, s) \in Z^k(b-u)$, so $\phi^k(u)$ is finite for all $u \in \mathbb{R}^m$. Also from the definition, it follows that

$$\phi^1 \geq \ldots \geq \phi^K.$$

We also introduce the set

$$\Pi^k = \{\pi \in \mathbb{R}^{n_k} : \pi t \leq \phi^{k-1}(A^k t) \text{ for all } t \in X^k\}$$

of lifting coefficients. For most of the results, it will suffice to consider the case where $K = 2$.

Proposition 2.1 *If $\pi^1 z^1 \leq \pi_0 + \nu s$ is a tight valid inequality for $Z^1(b)$, then*

$$\pi^1 z^1 + \pi^2 z^2 \leq \pi_0 + \nu s$$

is valid for $Z^2(b)$ if and only if $\pi^2 \in \Pi^2$.

Proof: Suppose $\pi^2 \in \Pi^2$ and consider a point $(\bar{z}^1, t, \bar{s}) \in Z^2(b)$. Then

$$\begin{aligned} \pi^2 t &\leq \phi^1(A^2 t) \text{ as } t \in X^2 \\ &\leq \pi^0 + \nu\bar{s} - \pi^1\bar{z}^1 \text{ as } (\bar{z}^1, \bar{s}) \in Z^1(b - A^2 t) \end{aligned}$$

and so the inequality is valid.

Conversely if $\pi^2 t > \phi^1(A^2 t)$ for some $t \in X^2$, take a point $(z^1, s) \in Z^1(b - A^2 t)$ with $\pi^1 z^1 = \pi_0 + \nu s - \phi^1(A^2 t)$. Now $(z^1, t, s) \in Z^2(b)$, but $\pi^1 z^1 + \pi^2 t > \pi_0 + \nu s$, and the inequality is not valid. □

We now consider briefly the structure of the functions ϕ^k and the sets Π^k, and whether the required calculations can be carried out.

Observation 2.4 *$\phi^k(u)$ is the value function of a mixed integer program. Thus there exists a finite set of polyhedra $P^q = \{u \in \mathbb{R}^m : D^q u \leq d^q\}$ whose union is $\mathbb{R}^m$ and vectors $(\alpha^q, \beta^q) \in \mathbb{R}^m \times \mathbb{R}^1$ such that for all q*

$$\phi^k(u) = \alpha^q u + \beta^q \text{ for } u \in P^q.$$

Proposition 2.2 *If each set $X^k = \{(x, y) \in \mathbb{R}_+^{n_k^1} \times \mathbb{Z}_+^{n_k^2} : C_1^k x + C_2^k y \leq c^k\}$ is a bounded mixed integer set, then Π^k is a polyhedron.*

Proof: We consider Π^2. Now for fixed $y \in \text{proj}_y(X^2)$ and fixed region q, with $\pi = (\lambda, \mu) \in \Pi^2$ and $t = (x, y) \in X^2$, $\pi \in \Pi^2$ if and only if

$$\pi t = \lambda x + \mu y \leq \phi^2(A^2 t) \leq \alpha^q(A_1^2 x + A_2^2 y) + \beta^q$$

for all $u = A^2 t = A_1^2 x + A_2^2 y$ satisfying $D^q u \leq d^q$, with $A^2 = (A_1^2, A_2^2)$ and $C^2 = (C_1^2, C_2^2)$. In other words $\pi \in \Pi^2$ if and only if $(\lambda - \alpha^q A_1^2)x + (\mu - \alpha^q A_2^2)y \leq \beta^q$ for all x such that $D^q(A_1^2 x + A_2^2 y) \leq d^q, C_1^2 x \leq c - C_2^2 y, x \geq 0$. For fixed y and q, the latter set is a bounded polyhedron in x with a finite number of extreme points $\{x^t\}_{t=1}^T$. Thus enumerating over the finite set of feasible integer vectors y, the finite number of regions q and the finite set of extreme points, we obtain an explicit description:

$$\Pi^2 = \{\pi = (\lambda, \mu) : \lambda x^t + \mu y \leq \alpha^q(A_1^2 x^t + A_2^2 y) + \beta^q \ \forall q, y, t\}.$$

□

Proposition 2.2 states that the lifting coefficients can be determined as feasible solutions of a finite set of linear inequalities. This finite set of linear inequalities comes from the inequality

$$\pi t \leq \phi^{k-1}(A^k t) \tag{2.6}$$

expressed for every t such that there exists $z^k = t$ feasible for Z^k, and all pieces of ϕ^{k-1}, since we know that ϕ is piecewise affine (by Observation 2.4). This implies that it suffices to express the inequalities (2.6) on every extreme point of the polyhedra $P^q = \{u \in \mathbb{R}^m : D^q u \leq d^q\}$ i.e. for every t such that $\phi^{k-1}(A^k t)$ is a break point of the lifting function.

Next if $\Pi^2 \neq \emptyset$ and lifting coefficients $\pi^2 \in \Pi^2$ have been selected, one needs to calculate the new lifting function ϕ^2. Rather than calculating it from scratch, it can also be obtained by updating.

Proposition 2.3

$$\phi^2(u) = \min_{t \in X^2} [\phi^1(u + A^2 t) - \pi^2 t].$$

Proof: By definition $\phi^2(u)$

$$\begin{aligned}
= & \min_{z^1, z^2, s} \{\pi_0 + \nu s - \pi^1 z^1 - \pi^2 z^2 : \\
& \qquad A^1 z^1 + A^2 z^2 \leq b + s - u, z^i \in X^i \text{ for } i = 1, 2, s \geq 0\} \\
= & \min_{t \in X^2} \{\min_{z^1, s} \{\pi_0 + \nu s - \pi^1 z^1 : A^1 z^1 \leq b + s - u - A^2 t, z^1 \in X^1, s \geq 0\} - \pi^2 t\} \\
= & \min_{t \in X^2} \{\phi^1(u + A^2 t) - \pi^2 t\}.
\end{aligned}$$

□

Example: This example is divided in four parts. We show that the four cases presented by Gu, Nemhauser and Savelsbergh [26] can be treated in a unified way in our framework and that two distinct lifting functions are not needed. In each of the four examples, we start from a different single node flow set X_1, X_2, X_3, X_4 and by projecting one variable, we obtain the same set

$$\begin{aligned}
Y = \{ & x \in \mathbb{R}^5_+, y \in \{0,1\}^5 : x_1 + x_2 + x_3 - x_4 - x_5 \leq 4 + s, \\
& 0 \leq x_1 \leq 3y_1,\ 0 \leq x_2 \leq 4y_2,\ 0 \leq x_3 \leq 5y_3,\ 0 \leq x_4 \leq 2y_4,\ 0 \leq x_5 \leq 3y_5\ \}.
\end{aligned}$$

A valid inequality for this set is

$$x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 3 + s. \tag{2.7}$$

In each of the four examples, we lift this inequality. The four cases from Gu et al. come from two binary choices that have to be made. The first choice is on the variable to fix. It comes either from N^+, i.e. its coefficient in the flow constraint is 1, or from N^-, i.e. coefficient equal to -1. The second choice is the value at which we fix the variable. It is either at 0 (lower bound) or at the upper bound of the variable.

Case 1 We fix the variable $x_0 \in N^+$ at 0 and $y_0 = 0$.
The starting set is

$$\begin{aligned}
X_1 = \{ & x \in \mathbb{R}^6_+, y \in \{0,1\}^6 : x_0 + x_1 + x_2 + x_3 - x_4 - x_5 \leq 4 + s, \\
& 0 \leq x_0 \leq 6y_0,\ 0 \leq x_1 \leq 3y_1,\ 0 \leq x_2 \leq 4y_2, \dots \qquad \}.
\end{aligned}$$

If we fix $x_0 = 0$, we obtain the set Y and the valid inequality (2.7). The lifting function can be computed for $u \in [0, 6]$ as

$$\begin{aligned}
\phi^1(u) = \min \quad & 3 - x_1 + 2y_1 - x_2 + y_2 - x_3 + 2y_3 + x_5 + s \\
\text{s.t.} \quad & x_1 + x_2 + x_3 - x_4 - x_5 \leq 4 + s - u \\
& 0 \leq x_1 \leq 3y_1,\ 0 \leq x_2 \leq 4y_2,\ 0 \leq x_3 \leq 5y_3, \dots \\
& x \in \mathbb{R}^5_+,\ y \in \{0,1\}^5.
\end{aligned}$$

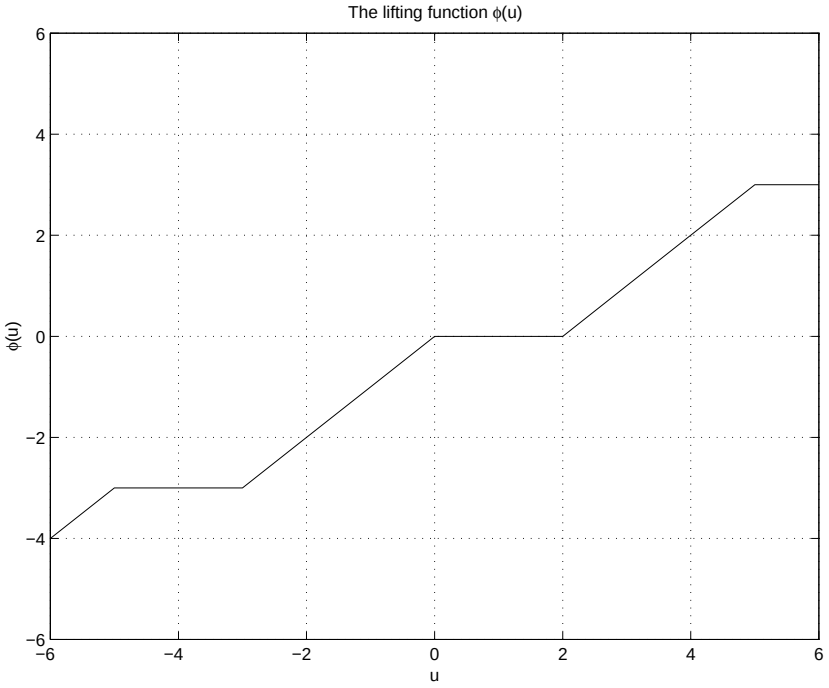

Figure 2.6: The lifting function used for the four cases

The value of $\phi^1(u)$ is shown in Figure 2.6. For this case, the admissible values of u are the admissible values of x_0, i.e. from 0 to 6. If we now try to lift the variables (x_0, y_0) with the coefficients (λ_0^1, μ_0^1), we have to satisfy the following conditions coming from the singular points of both the domain of the variables and the function $\phi^1(u)$, namely

$$\begin{array}{ll} u = 0: & \mu_0 \leq 0 \\ u = 2: & 2\lambda_0^1 + \mu_0^1 \leq 0 \\ u = 5: & 5\lambda_0^1 + \mu_0^1 \leq 3 \\ u = 6: & 6\lambda_0^1 + \mu_0^1 \leq 3 \end{array}$$

Two extreme solutions are $(\lambda_0^1, \mu_0^1) = (0, 0)$ and $(\lambda_0^1, \mu_0^1) = (\frac{3}{4}, -\frac{3}{2})$. These coefficients are computed by finding an affine support to the lifting function on the valid domain of u. One of the affine supports is shown in Figure 2.7. The other is provided by the x axis and leads to the lifting coefficients $(0, 0)$. The

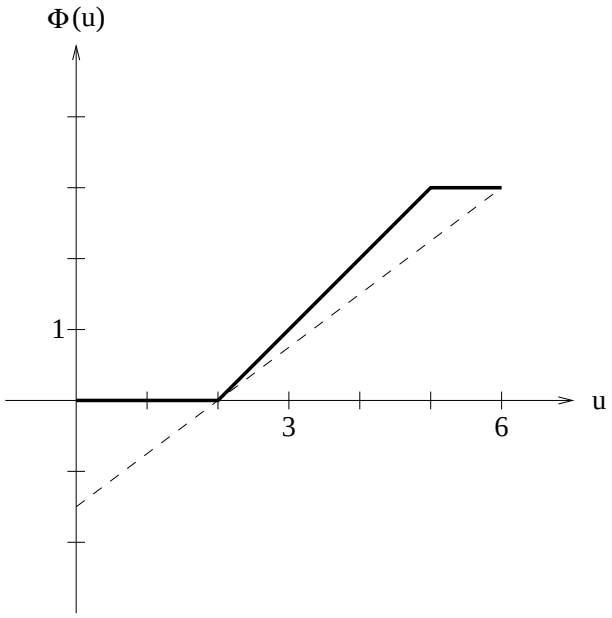

Figure 2.7: An affine support to the lifting function on $[0, 6]$

case $(\lambda_0^1, \mu_0^1) = (\frac{3}{4}, -\frac{3}{2})$ yields the valid inequality

$$\frac{3}{4}x_0 - \frac{3}{2}y_0 + x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 3 + s.$$

for X_1.

Case 2 We fix the variable $y_0 = 1$ and $x_0 \in N^+$ at its upper bound.
In this case, the starting set is

$$X_2 = \{x \in \mathbb{R}^6_+, y \in \{0,1\}^6 : x_0 + x_1 + x_2 + x_3 - x_4 - x_5 \leq 10 + s, \qquad (2.8)$$
$$0 \leq x_0 \leq 6y_0,\ 0 \leq x_1 \leq 3y_1,\ 0 \leq x_2 \leq 4y_2, \dots \quad \}.$$

Since we want to fix $x_0 = 6$, we have to complement the variables in order to be able to fix to 0. Therefore we define $\hat{x}_0 = 6 - x_0$ and $\hat{y}_0 = 1 - y_0$. By substituting in (2.8), we obtain

$$x_1 + x_2 + x_3 - \hat{x}_0 - x_4 - x_5 \leq 4 + s$$

for the flow constraint. The corresponding variable upper bound is now $\hat{x}_0 \leq 6$ and $\hat{x}_0 \geq 6\hat{y}_0$. If we fix $\hat{x}_0 = 0$, we obtain the set Y and the valid inequality (2.7). The lifting function is exactly the same as in Case 1 and is depicted in Figure 2.6. But in this case, the interesting values of u are those corresponding to the feasible values of $-x_0$, namely the interval $[-6, 0]$. The lifting coefficients (λ_0^2, μ_0^2) of $\hat{x}_0$ and $\hat{y}_0$ are determined by

$$\begin{array}{ll} u = -3: & 3\lambda_0^2 \leq -3 \\ u = -5: & 5\lambda_0^2 \leq -3 \\ u = -6: & \mu_0^2 + 6\lambda_0^2 \leq -4 \end{array}$$

An extreme solution is $(\lambda_0^2, \mu_0^2) = (-1, 2)$. Therefore the inequality

$$-\hat{x}_0 + 2\hat{y}_0 + x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 3 + s,$$

which can be rewritten as

$$x_0 - 2y_0 + x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 7 + s$$

is a valid inequality for X_2.

Case 3 We fix the variable $y_0 = 0$ and $x_0 \in N^-$ to 0.
In this case, the starting set is

$$X_3 = \{x \in \mathbb{R}^6_+, y \in \{0,1\}^6 : x_1 + x_2 + x_3 - x_0 - x_4 - x_5 \leq 4 + s,$$
$$0 \leq x_0 \leq 6y_0,\ 0 \leq x_1 \leq 3y_1,\ 0 \leq x_2 \leq 4y_2, \dots \quad \}.$$

By fixing again $x_0 = 0$, we obtain the set Y and for example the valid inequality (2.7). The lifting function can be computed and is the same as for the two previous cases. It is shown in Figure 2.6. The interesting values of u are exactly the feasible values of $-x_0$ i.e. the interval $[-6, 0]$ again. To obtain the valid lifting coefficients (λ_0^3, μ_0^3) corresponding to the variables (x_0, y_0), we

must find values of (λ_0^3, μ_0^3) satisfying

$$\begin{aligned} u = 0: &\quad \mu_0^3 \leq 0 \\ u = -3: &\quad 3\lambda_0^3 + \mu_0^3 \leq -3 \\ u = -5: &\quad 5\lambda_0^3 + \mu_0^3 \leq -3 \\ u = -6: &\quad 6\lambda_0^3 + \mu_0^3 \leq -4 \end{aligned}$$

Two extreme solutions to this system are $(\lambda_0^3, \mu_0^3) = (-1, 0)$ and $(\lambda_0^3, \mu_0^3) = (-\frac{1}{3}, -2)$. Therefore, the best valid inequalities for X_3 that can be obtained from this lifting are

$$-x_0 + x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 3 + s \quad \text{and}$$
$$-\frac{1}{3}x_0 - 2y_0 + x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 3 + s.$$

Case 4 We fix the variable $x_0 \in N^-$ at its upper bound.
The initial problem is therefore

$$\begin{aligned} X_4 = \{x \in \mathbb{R}^6_+, y \in \{0,1\}^6 : x_1 + x_2 + x_3 - x_0 - x_4 - x_5 \leq -2 + s, \\ 0 \leq x_0 \leq 6y_0,\ 0 \leq x_1 \leq 3y_1,\ 0 \leq x_2 \leq 4y_2, \dots \quad \}. \end{aligned} \tag{2.9}$$

We want to fix $x_0 = 6$, therefore we again need to change the variables in order to be able to apply our framework. Therefore we introduce $\hat{x}_0 = 6 - x_0$ and $\hat{y}_0 = 1 - y_0$. If we substitute in (2.9), we obtain

$$\hat{x}_0 + x_1 + x_2 + x_3 - x_4 - x_5 \leq 4 + s,$$

and by fixing $\hat{x}_0 = \hat{y}_0 = 0$, we again obtain the set Y. If we compute the lifting function corresponding to the valid inequality (2.7), it yields the same lifting function as in the three previous cases (see Figure 2.6). The domain of interesting values of u corresponds to the values of $\hat{x}_0$, i.e. the interval $[0, 6]$. The conditions to be satisfied by the lifting coefficients are

$$\begin{aligned} u = 2: &\quad 2\lambda_0^4 \leq 0 \\ u = 5: &\quad 5\lambda_0^4 \leq 3 \\ u = 6: &\quad 6\lambda_0^4 + \mu_0^4 \leq 3. \end{aligned}$$

An extreme solution for this system is $(\lambda_0^4, \mu_0^4) = (0, 3)$ which leads to the valid inequality for X_4

$$3\hat{y}_0 + x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 3 + s$$

which can be rewritten as

$$-3y_0 + x_1 - 2y_1 + x_2 - y_2 + x_3 - 2y_3 - x_5 \leq 0 + s.$$

It is interesting to remark here that the four cases have a graphic illustration. For Case 1 and 3, the lifting problem consists in finding an affine support to the lifting function on the domain of x. The slope of the support gives the coefficient of x while the y-intersect gives the coefficient of y. For Cases 2 and 4, the situation is slightly different. Indeed the link between the two variables to lift is not like a standard variable upper bound constraint due to the variable substitution. Therefore, the lifting problem consists now in finding a linear support to the lifting function on the domain of x. The slope of the function gives the coefficient of $\hat{x}$, while the slack at the end of the domain provides the coefficient of $\hat{y}$. These affine supports always include the point $(0, 0)$. This is illustrated in Figure 2.8. For each of the four cases, the place where the coefficient μ is computed is indicated. The coefficient λ is always the slope of the support. □

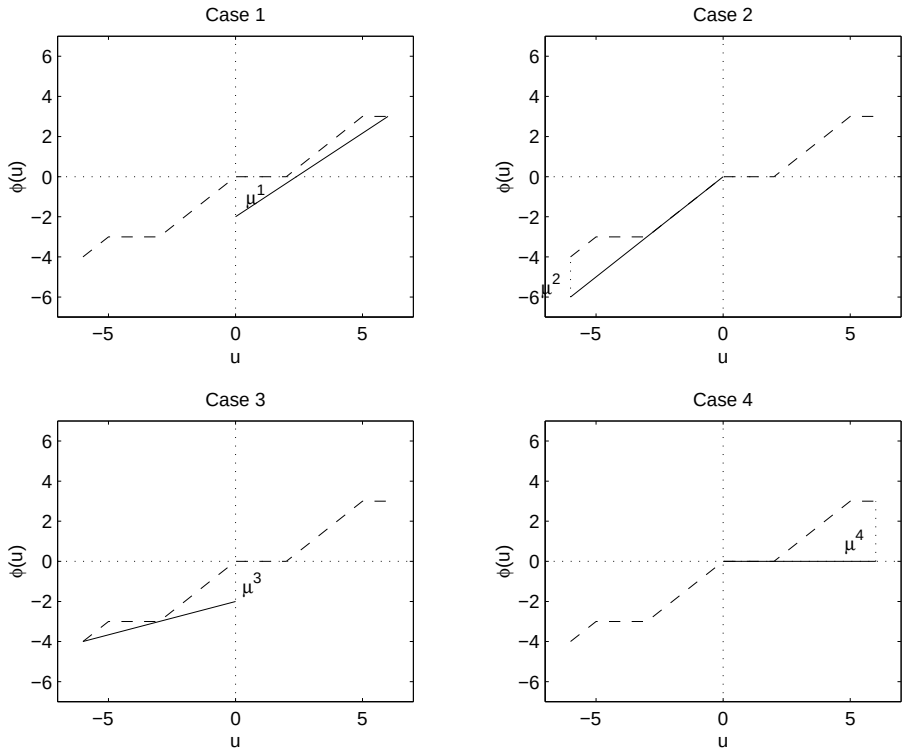

Figure 2.8: Lifting as finding affine supports to the lifting function

A second example shows a variant of the four cases presented above. Here

there is both a variable upper and lower bound. Furthermore the discrete variables y_k take three different values instead of two.

Example: (The Lifting Function and the Set of Lifting Coefficients).
Consider the set

$$5y_1 + 5y_2 + 5y_3 + x_4 + 2y_4 \leq 12 + s$$
$$1y_4 \leq x_4 \leq 3y_4$$
$$y_i \in \{0,1\} \text{ for } i = 1,\dots 3,\ y_4 \in \{0,1,2\}, x_4 \in \mathbb{R}^1_+.$$

This can be modelled in the form (2.3) with $A^1 = (5,5,5), A^2 = (1,2), b = 12, X^1 = \{0,1\}^3, X^2 = \{(x_4,y_4) \in \mathbb{R}^1_+ \times \mathbb{Z}^1_+ : 1y_4 \leq x_4 \leq 3y_4,\ y_4 \leq 2\}$.
As valid inequality for $Z^1(b)$, we take

$$3y_1 + 3y_2 + 3y_3 \leq 6 + s. \tag{2.10}$$

The lifting function ϕ^1 is given, see Figure 2.9, by

$$\phi^1(u) = \begin{cases} -3 & if \quad u \leq -3 \\ u & if \quad -3 \leq u < 0 \\ 0 & if \quad 0 \leq u < 2 \\ u-2 & if \quad 2 \leq u < 5 \\ 3 & if \quad 5 \leq u < 7 \\ u-4 & if \quad 7 \leq u < 10 \\ 6 & if \quad 10 \leq u < 12 \\ u-6 & if \quad 12 \leq u. \end{cases}$$

Now we wish to find lifting coefficients $\pi^2 = (\lambda, \mu) \in \Pi^2$. We note that $3 \leq x_4 + 2y_4 \leq 10$ and $y_4 \in \{1,2\}$ for $(x,y) \in X^2 \setminus \{(0,0)\}$.
For $y_4 = 1$, $1 \leq x_4 \leq 3$ and thus $3 \leq u = x_4 + 2y_4 \leq 5$. This just intersects the region/segment $u \in [2,5]$, and the extreme points are $x_4 = 1$ and $x_4 = 3$.

For $y_4 = 2$, $6 \leq u = x_4 + 2y_4 \leq 10$, and two segments $[5,7]$ and $[7,10]$ are intersected leading to the extreme points $x_4 = 2, x_4 = 3$ and $x_4 = 6$. So Π^2 is described by the inequalities

$$\begin{array}{rcrcrcl} 1\lambda & + & 1\mu & \leq & \phi^1(3) & = & 1 \\ 3\lambda & + & 1\mu & \leq & \phi^1(5) & = & 3 \\ 2\lambda & + & 2\mu & \leq & \phi^1(6) & = & 3 \\ 3\lambda & + & 2\mu & \leq & \phi^1(7) & = & 3 \\ 6\lambda & + & 2\mu & \leq & \phi^1(10) & = & 6 \end{array}$$

with extreme points $\pi = (-1,2)$ and $\pi = (1,0)$ giving the valid inequalities

$$3y_1 + 3y_2 + 3y_3 - z_4 + 2y_4 \leq 12 + s \text{ and}$$

$$3y_1 + 3y_2 + 3y_3 + z_4 \leq 12 + s.$$

□

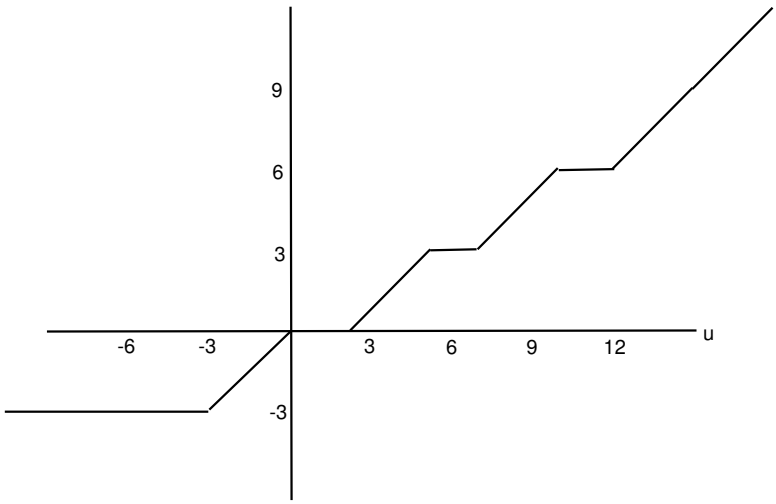

Figure 2.9: Lifting function

2.3.3 Superadditive Lifting

The question of updating the lifting function is crucial because it is usually computationally intractable to lift too many variables at a time. That is why one has to include variables into blocks lifted sequentially. Between each computation of coefficients, the lifting function has to be updated. In the general case, the order in which we consider the blocks to lift is relevant. A different ordering leads to different coefficients. However when the lifting function has more structure, the lifting can be *sequence independent*.

Definition 2.2 *A function* $F : D \to \mathbb{R}$ *is* superadditive *on* $D \subseteq \mathbb{R}^m$ *if*

$$F(u) + F(v) \leq F(u + v)$$

for all u, v *for which* $u, v, u + v \in D$.

Throughout we will assume that D is a cone, so that $u, v \in D$ implies $u+v \in D$. We also limit our attention to superadditive functions that are continuous, with the property that $\bar{F}(d) = \lim_{t \to 0} \frac{F(td)}{t}$ exists for all $d \in D$, and with $F(\underline{0}) = 0$. Two classes of functions will be very useful later.

Definition 2.3 *For* $0 < \alpha < 1$, *the* mixed integer rounding function $F_\alpha : \mathbb{R}^1 \to \mathbb{R}^1$ *is defined by*

$$F_\alpha(d) = \lfloor d \rfloor + \frac{(f_d - \alpha)^+}{1 - \alpha},$$

where $f_d = d - \lfloor d \rfloor$.

This function is superadditive on $\mathbb{R}^1$ and is shown in Figure 2.10. Note that $\bar{F}_\alpha$ exists, and $\bar{F}_\alpha(d) = \min[0, \frac{d}{1-\alpha}]$.

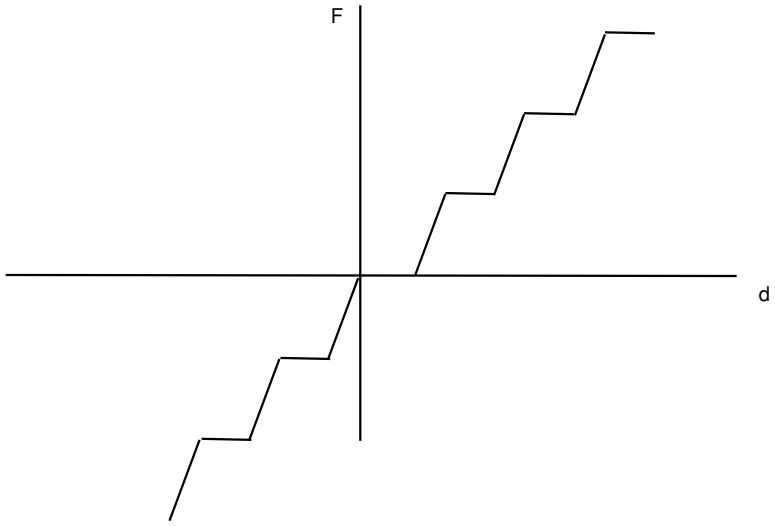

Figure 2.10: Superadditive MIR Function

Definition 2.4 *Suppose that* $a \in \mathbb{R}^n_+$ *with* $a_{i_1} \geq a_{i_2} \geq \ldots \geq a_{i_r} > \lambda \geq a_{i_{r+1}} \ldots a_{i_n} > 0$, *and let* $A_t = \sum_{j=1}^t a_{i_j}$ *for* $t \leq r$ *with* $A_0 = 0$ *and* $A_{r+1} = \infty$. *Define* $G_{a,\lambda} : \mathbb{R}^1_+ \to \mathbb{R}^1_+$ *by*

$$G_{a,\lambda}(u) = \begin{cases} (j-1)\lambda \text{ if } A_{j-1} \leq u \leq A_j - \lambda \ j = 1, \ldots, r \\ (j-1)\lambda + [u - (A_j - \lambda)] \text{ if } A_j - \lambda \leq u \leq A_j \ j = 1, \ldots, r-1 \\ (r-1)\lambda + [u - (A_r - \lambda)] \text{ if } A_r - \lambda \leq u. \end{cases}$$

This function is superadditive on $\mathbb{R}^1_+$ and an instance with $a = (10, 7, 4, 2)$ and $\lambda = 3$ is shown in Figure 2.11. Though we will not use it directly here, superadditive functions are basic to mixed integer programming as the next Proposition indicates.

Proposition 2.4 *[31, 30] If* $X^{MIP} = \{(x, y) \in \mathbb{R}^{n_1^1}_+ \times \mathbb{Z}^{n_1^2}_+ : A_1 x + A_2 y \leq b\}$, $F : \mathbb{R}^m \to \mathbb{R}^1$ *is superadditive and nondecreasing, and* $\bar{F}$ *exists, then*

$$\sum_{j=1}^{n_1^1} \bar{F}(a_{1j}) x_j + \sum_{j=1}^{n_1^2} F(a_{2j}) y_j \leq F(b)$$

is a valid inequality for X^{MIP}, *where* a_{1j} *and* a_{2j} *are the columns of* A_1 *and* A_2 *respectively.*

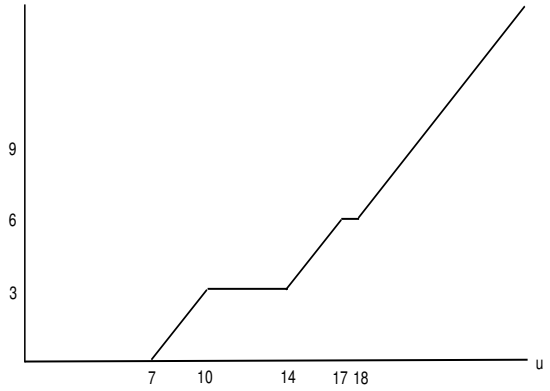

Figure 2.11: Superadditive Function $G_{a,\lambda}$ on $\mathbb{R}^1_+$

Now we consider what happens when the lifting function ϕ^1 is superadditive on some cone D.

Proposition 2.5 *If ϕ^1 is superadditive on D, and $A^2t \in D$ for all $t \in X^2$, $\phi^2 = \phi^1$ on D.*

Proof: For $t \in X^2$ and $u \in D$,

$$\begin{aligned} \phi^1(u + A^2t) - \pi^2 t &\geq \phi^1(u + A^2t) - \phi^1(A^2t) \quad \text{as } \pi^2 t \leq \phi^1(A^2t) \\ &\geq \phi^1(u) \quad \text{by superadditivity.} \end{aligned}$$

On the other hand as $0 \in X^2$, $\min_{t \in X^2}[\phi^1(u + A^2t) - \pi^2 t] \leq \phi^1(u)$. Therefore for $u \in D$, $\phi^2(u) = \min_{t \in X^2}[\phi^1(u + A^2t) - \pi^2 t] = \phi^1(u)$. □

This proposition is important for practical issues. Whenever a lifting function is superadditive, the ordering in which we lift the different blocks of variables is irrelevant. Furthermore the lifting function need not to be computed again between the different computations of coefficients.

When the lifting function is not superadditive, it may change after each computation of a coefficient. However one can avoid the updating of the lifting function by using a weaker version and obtain the same behaviour. Any function $\hat{\phi} \leq \phi^1$ is called a *valid lifting fuction* for X^2 because $\pi^2 t \leq \hat{\phi}(A^2t)$ for all $t \in X^2$ implies that $\pi^1 z^1 + \pi^2 z^2 \leq \pi_0 + \nu s$ is valid for $Z^2(b)$. We say that $\hat{\phi}$ *is used for lifting* if π^2 satisfies $\pi^2 t \leq \hat{\phi}(A^2t)$ for all $t \in X^2$. When in addition $\hat{\phi}$ is superadditive, we keep the property of sequence independent lifting.

Proposition 2.6 *Suppose that $\hat{\phi} \leq \phi^1$ on D and $\hat{\phi}$ is superadditive on D. If $A^2t \in D$ for all $t \in X^2$ and $\hat{\phi}$ is used for lifting, then $\phi^1 \geq \phi^2 \geq \hat{\phi}$ on D.*

Proof: As $\hat{\phi}$ is used for lifting, $\pi^2 t \leq \hat{\phi}(A^2t)$ for $t \in X^2$. Now for $t \in X^2$ and $u \in D$, $\phi^1(u + A^2t) - \pi^2 t \geq \phi^1(u + A^2t) - \hat{\phi}(A^2t) \geq \hat{\phi}(u + A^2t) - \hat{\phi}(A^2t) \geq \hat{\phi}(u)$

where the last two inequalities follow from $\hat{\phi} \leq \phi^1$ and the superadditivity of $\hat{\phi}$ on D respectively. Therefore $\phi^2(u) = \min_{t \in X^2}[\phi^1(u + A^2 t) - \pi^2 t] \geq \hat{\phi}(u)$ for $u \in D$. □

So $\hat{\phi}$ remains a valid lifting function for $\phi^2, \ldots, \phi^K$. Thus if such a superadditive function $\hat{\phi}$ is used for lifting, the ordering of the sets $X^2, \ldots, X^K$ and of the calculations is irrelevant, as shown by the following result.

Corollary 2.7 *If $\hat{\phi} \leq \phi^1$ and $\hat{\phi}$ is superadditive on D, $\hat{\pi}^k t \leq \hat{\phi}(A^k t)$ and $A^k t \in D$ for all $t \in X^k$ and $k = 2, \ldots, K$, then*

$$\pi^1 z^1 + \sum_{k=2}^{K} \hat{\pi}^k z^k \leq \pi_0 + \nu s$$

is valid for $Z^K(b)$.

Proof: If $(z^1, \ldots, z^K, s) \in Z^K(b)$,

$$\begin{aligned}
\pi^1 z^1 + \sum_{k=2}^{K} \hat{\pi}^k z^k &\leq \pi_0 + \nu s - \phi^1(\sum_{k=2}^{K} A^k z^k) + \sum_{k=2}^{K} \hat{\phi}(A^k z^k) \\
&\leq \pi_0 + \nu s - \phi^1(\sum_{k=2}^{K} A^k z^k) + \hat{\phi}(\sum_{k=2}^{K} A^k z^k) \quad \text{by superadditivity} \\
&\leq \pi_0 + \nu s \qquad \text{as } \hat{\phi} \leq \phi^1.
\end{aligned}$$

□

It is natural to ask whether functions such as $\hat{\phi}$ always exist.

Proposition 2.8 *$\phi^*(u) = \min_{v \in D}[\phi^1(u + v) - \phi^1(v)]$ is superadditive on D, and $\phi^* \leq \phi^1$ on D.*

Proof: For $u, v \in D$, $u + v \in D$ and so, for some $w \in D$,
$\phi^*(u+v) = \phi^1(u+v+w) - \phi^1(w) = \phi^1(u+v+w) - \phi^1(v+w) + \phi^1(v+w) - \phi^1(w)$
$\geq \phi^*(u) + \phi^*(v)$. Also $\phi^*(u) \leq \phi^1(u + \underline{0}) - \phi(\underline{0}) = \phi^1(u)$ for all $u \in D$. □

If a superadditive function has been used to generate the initial valid inequality $\pi^1 z^1 \leq \pi_0 + \nu s$ for $Z^1(b)$, there is a natural candidate to be used as a valid lifting function.

Proposition 2.9 *Suppose that the initial valid inequality for $Z^1(b)$ is of the form*

$$\sum_{j=1}^{n_1^1} \bar{F}(a_{1j}) x_j^1 + \sum_{j=1}^{n_1^2} F(a_{2j}) y_j^1 \leq F(b)$$

with F superadditive and nondecreasing on $\mathbb{R}^m$. Then $\hat{F}(u) \equiv F(b) - F(b - u)$ is a valid lifting function with $\hat{F}(u) \leq \phi^1(u)$ for all $u \in \mathbb{R}^m$.

Proof: $\phi^1(u) = F(b) - \max\{\sum_{j=1}^{n_1^1} \bar{F}(a_{1j})x_j^1 + \sum_{j=1}^{n_1^2} F(a_{2j})y_j^1 : A_1^1x^1 + A_2^1y^1 \leq b-u, (x^1, y^1) \in \mathbb{R}^{n_1^1} \times \mathbb{Z}^{n_1^2}\} \geq F(b) - F(b-u)$ as $\sum_{j=1}^{n_1^1} \bar{F}(a_{1j})x_j^1 + \sum_{j=1}^{n_1^2} F(a_{2j})y_j^1 \leq F(b-u)$ is a valid inequality for $Z^1(b-u)$. □

Now suppose that the MIR function F_α is used to generate the first inequality.

Observation 2.5 *If $f_b = \alpha$, $\hat{F}(u) = F_\alpha(b) - F_\alpha(b-u) = F_\alpha(u)$ for all $u \in \mathbb{R}^m$.*

Thus F_α is itself a valid superadditive lifting function.

Example: (Simultaneous Lifting)
Consider the initial set X

$$\begin{gathered} 5y_1 + 5y_2 + 5y_3 + x_4 - x_5 \leq 4 + s \\ 0 \leq x_4 \leq 6y_4, 0 \leq x_5 \leq 8y_5 \\ y \in \{0,1\}^5, s \in \mathbb{R}^1_+. \end{gathered}$$

Setting $x_4 = 0, y_4 = 0, x_5 = 8, y_5 = 1$, we obtain the set Y

$$\begin{gathered} 5y_1 + 5y_2 + 5y_3 \leq 12 + s \\ y \in \{0,1\}^3, s \in \mathbb{R}^1_+. \end{gathered}$$

We now rewrite the set X in the form (2.3).

$$\begin{array}{ccccc} 5y_1 + 5y_2 + 5y_3 & & +x_4 & & +\bar{x}_5 - s \\ (y_1, \ldots, y_3) \in X^1, & & (x_4, y_4) \in X^2, & & (\bar{x}_5, \bar{y}_5) \in X^3, s \in \mathbb{R}^1_+, \end{array} \quad \leq 12$$

where $\bar{x}_5 = 8 - x_5$, $\bar{y}_5 = 1 - y_5$,
$X^1 = \{0,1\}^3$, $A^1 = (5,5,5)$
$X^2 = \{(x_4, y_4) \in \mathbb{R}^1_+ \times \{0,1\} : x_4 \leq 6y_4\}$, $A^2 = (1)$
$X^3 = \{(\bar{x}_5, \bar{y}_5) \in \mathbb{R}^1 \times \{0,1\} : 8 \geq \bar{x}_5 \geq 8\bar{y}_5\}$, $A^3 = (1)$ and $b = 12$.

Taking the same valid inequality (2.10), it is easily checked that its lifting function ϕ^1 is superaditive on $\mathbb{R}^1_+$. As $A^2t \in \mathbb{R}^1_+$ for all $t \in X^2$ and $A^3t \in \mathbb{R}^1_+$ for all $t \in X^3$, Corollary 2.7 of Proposition 2.6 is applicable.

For the set X^2, we have

$$\Pi^2 = \{(\lambda, \mu) : \lambda x_4 + \mu y_4 \leq \phi^1(x_4) \text{ for } (x_4, y_4) \in X^2\},$$

and, as shown in [26], we obtain valid lifting coefficients by taking a support to the lifting function ϕ^1 over the range [0,6]. There are two extreme solutions $(\lambda, \mu) = (0,0)$ and $(\lambda, \mu) = (\frac{3}{4}, -\frac{3}{2})$.

For the set X^3,

$$\Pi^3 = \{(\lambda, \mu) : \lambda \bar{x}_5 + \mu \bar{y}_5 \leq \phi(\bar{x}_5) \text{ for } (\bar{x}_5, \bar{y}_5) \in X^3\}$$

with unique extreme point $(\lambda, \mu) = (0, \phi(8)) = (0, 4)$.

So simultaneously lifting on the sets X^2 and X^3 gives the valid inequalities

$$3y_1 + 3y_2 + 3y_3 + 4(1 - y_5) \leq 6 + s \text{ and}$$

$$3y_1 + 3y_2 + 3y_3 + (\frac{3}{4}x_4 - \frac{3}{2}y_4) + 4(1 - y_5) \leq 6 + s.$$

□

2.4 Other Remarks

2.4.1 The Role of the Continuous Variables s

The inclusion of the continuous variables s in the description (2.3) of $Z^k(b)$ for all $k = 1, \ldots, K$ simplifies the presentation, but clearly restricts the inequalities that can be obtained by lifting. When the variables s are kept, it guarantees that the lifting function is defined for all u and is continuous everywhere on the domain. These statements are expressed in the two following lemmas. Recall that we handle a set $Z^\tau(b)$ of the form

$$\begin{array}{c} \sum_{k=1}^{\tau} A^k z^k \leq b + s \\ z^k \in X^k \text{ for } k = 1, \ldots, \tau, s \in \mathbb{R}^m_+, \end{array} \tag{2.11}$$

where $A^k \in \mathbb{R}^{m \times n_k}$ for $k = 1, \ldots, K, b \in \mathbb{R}^m$, $X^k = \{z^k \in \mathbb{R}^{n_k^1} \times \mathbb{Z}^{n_k^2} : C^k z^k \leq c^k\}$ with $n_k = n_k^1 + n_k^2$ is a mixed integer set in $\mathbb{R}^{n_k}$ for all k and $0 \in X^k$ for $k = 2, \ldots, K$. Suppose that we lift a valid inequality

$$\sum_{k=1}^{\tau-1} \pi^k z^k \leq \pi_0 + \sum_{i=1}^{m} \nu_i s_i \tag{2.12}$$

with $\nu_i \geq 0$ for all $i = 1, \ldots, m$. The lifting function is defined by

$$\phi^\tau(u) = \min \left\{ \pi_0 + \sum_{i=1}^{m} \nu_i s_i - \sum_{k=1}^{\tau-1} \pi^k z^k : (z^1, \ldots, z^{\tau-1}, s) \in Z^{\tau-1}(b - u) \right\}. \tag{2.13}$$

Lemma 2.10 *If the continuous variables s are included in the model and not fixed to zero, the lifting function is defined everywhere and is such that for all $\epsilon \in \mathbb{R}^m_+$ and all $u \in \mathbb{R}^m$,*

$$\phi^\tau(u + \epsilon) \leq \phi^\tau(u) + \sum_{i=1}^{m} \nu_i \epsilon_i.$$

Proof: Let us fix $\tau \geq 2$. The lifting function ϕ^τ is defined everywhere since we assume that $0 \in X^{\tau-1}$ and there always exist some $s \in \mathbb{R}^m_+$ such that $b + s - u \in \mathbb{R}^m_+$.

We now prove the second property. We fix $u \in \mathbb{R}^m$. Since ϕ^τ is defined everywhere, there exists $(z^*, s^*) \in Z^{\tau-1}(b-u)$ for which $\phi^\tau(u)$ reaches its value. We now look at the value taken by the point $(z^*, s^* + \epsilon)$ in the integer program (2.13) defining $\phi^\tau(u + \epsilon)$. The point is feasible. Indeed $(z^*, s^*) \in Z^{\tau-1}(b - u)$ implies $(z^*, s^* + \epsilon) \in Z^{\tau-1}(b - u + \epsilon)$. The objective function for this point is

$$\pi_0 + \sum_{i=1}^{m} \nu_i(s_i + \epsilon_i) - \sum_{k=1}^{\tau-1} \pi^k z^{*k} = \phi^\tau + \sum_{i=1}^{m} \nu_i \epsilon_i.$$

The optimal value for (2.13) defining $\phi^\tau(u + \epsilon)$ cannot be greater. Therefore we have $\phi^\tau(u + \epsilon) \leq \phi^\tau(u) + \sum_{i=1}^{m} \nu_i \epsilon_i$. □

Corollary 2.11 *When continuous variables s are included in* (2.11)*, the lifting function $\phi^\tau(u)$ is continuous.*

Proof: This follows immediately from Lemma 2.10 and the fact that the function ϕ is nondecreasing. □

If the variables s are set to zero and lifted later, we no longer have that $Z^k(b - u) \neq \emptyset$ for all u, with the result that ϕ^k can be discontinuous, and is not defined everywhere. Calculating ϕ^k and new coefficients π^{k+1}, and finding a valid lifting function that is superadditive remain difficult problems. The resulting lifting functions ϕ^k are potentially stronger, but the final inequality may not be valid until the variables s are lifted in. Examples of such functions can be found in [26] among others.

2.4.2 Facet-defining Inequalities

We have not discussed at all the question when the lifted inequalities are facet-defining. The brief answer is that if the set $Z^1(b)$ is full-dimensional, the initial inequality is facet-defining, the exact lifting function is used to define Π^k and π^k is an extreme point of Π^k for all k, then the final inequality (2.4) is facet-defining for $Z^K(b)$. When the sets are not full-dimensional, more conditions are needed. See [53] for a detailed study of this question.

2.4.3 The complexity of the lifting procedure

We have seen that the subproblems arising in the computation of the lifting function are new (mixed) integer problems. For an arbitrary valid inequality, they are NP-hard problems. Hence, the lifting procedure may turn out to be an expensive task. However there are some cases in which the computation

simplifies and for which the lifting approach becomes computationally interesting. For example in the next chapter, we study an approach to generate valid inequalities for the single node flow set. In this approach, the lifting functions that we obtain during the process always have the same structure. Furthermore it is always possible to find an analytic form of such functions. In this case the lifting procedure is easy and cheap to apply. Another example of tractable lifting includes the case of knapsacks. In that case the subproblems arising to compute the lifting function are other knapsacks but with smaller coefficients. These problems are therefore suitable for pseudo-polynomial methods like dynamic programming.

Chapter 3

Single Node Flow Sets

3.1 Introduction

As indicated in the previous chapter, single node flow sets X^N are a natural generalization of knapsack sets. As we also suggested in the previous chapter, we can take two approaches to tackle them. The first: studying $\text{conv}(X^N)$ in depth and thereby obtaining complete information about the special cases Y (for example the knapsack set with a continuous variable X^{CK}) is unfortunately still an important challenge. The second, using knowledge about special cases to obtain important, but partial knowledge about the superset X^N, is pursued here. Suprisingly we show that by using mixed-integer rounding (combined with superadditive lifting), one obtains inequalities at least as strong as all the flow cover inequalities proposed earlier. Again the results presented here come from [45].

Let us now define the basic set studied in this chapter, for which we try to generate valid inequalities.

Definition 3.1 *The single node flow set, denoted $X^N(n_1, n_2, b, a, u)$ is the set of $(x, y, s) \in \mathbb{R}_+^{n_1+n_2} \times \mathbb{Z}_+^{n_1+n_2} \times \mathbb{R}_+$ satisfying*

$$\begin{gathered}\textstyle\sum_{j\in N_1} x_j - \sum_{j\in N_2} x_j \le b + s \\ x_j \le a_j y_j \text{ for } j \in N_1 \cup N_2 \\ y_j \le u_j \text{ for } j \in N_1 \cup N_2,\end{gathered}$$

where $n_1 = |N^1|, n_2 = |N^2|, n = n_1 + n_2, b \in \mathbb{Z},\ a \in \mathbb{R}^n, u \in \mathbb{Z}_+^n$.

In this chapter, we focus on two cases of Definition 3.1: the 0-1 single node flow set $X^N(n_1, n_2, b, a, 1)$ and the integer single node flow set where u is finite and different from 0. The 0-1 single node flow set was first studied as natural

generalizations of 0-1 knapsack sets. Then it was shown that every mixed integer row can be rewritten as a restrivtion of a single node flow set. The following development comes from [50].

Observation 3.1 *Any mixed integer set T of the form*

$$\begin{aligned} T = \{(x,y) \in \mathbb{R}^{|J_1 \cup J_2|} \times \mathbb{Z}^{|J_1 \cup J_3|} : & \\ & \sum_{j \in J_1} (\alpha_j x_j + \beta_j y_j) + \sum_{j \in J_2} \alpha_j x_j + \sum_{j \in J_3} \beta_j y_j \leq b \\ & 0 \leq x_j \leq a_j y_j \qquad \textit{for } j \in J_1 \\ & 0 \leq x_j \leq a_j \qquad \textit{for } j \in J_2 \\ & 0 \leq y_j \leq b_j \qquad \textit{for } j \in J_1 \cup J_3 \qquad \} \end{aligned}$$

can be written in the single node flow set X^N form.

Proof: Let $J_1^+ = \{j \in J_1 : \alpha_j > 0\}, J_1^- = J_1 \setminus J_1^+$, $J_2^+ = \{j \in J_2 : \alpha_j > 0\}, J_2^- = J_2 \setminus J_2^+$, and similarly for J_3. We define now, for all j, a new continuous variable w_j as

$$w_j = \begin{cases} \alpha_j x_j + \beta_j y_j & \text{for } j \in J_1^+ \\ -(\alpha_j x_j + \beta_j y_j) & \text{for } j \in J_1^- \\ \alpha_j x_j & \text{for } j \in J_2^+ \\ -\alpha_j x_j & \text{for } j \in J_2^- \\ \beta_j y_j & \text{for } j \in J_3^+ \\ -\beta_j y_j & \text{for } j \in J_3^-, \end{cases}$$

and we introduce some values u_j that will be used as upper bounds of w_j,

$$u_j = \begin{cases} \alpha_j a_j + \beta_j & \text{for } j \in J_1^+ \\ -(\alpha_j a_j + \beta_j) & \text{for } j \in J_1^- \\ \alpha_j a_j & \text{for } j \in J_2^+ \\ -\alpha_j a_j & \text{for } j \in J_2^- \\ \beta_j & \text{for } j \in J_3^+ \\ -\beta_j & \text{for } j \in J_3^-. \end{cases}$$

Now T is exactly the single node flow set determined by the constraint

$$\begin{aligned} \sum_{j \in N^+} w_j - \sum_{j \in N^-} w_j \leq b & \\ 0 \leq w_j \leq u_j y_j \quad \text{for } j \in N^+ \cup N^-, & \end{aligned}$$

with $N^+ = J_1^+ \cup J_2^+ \cup J_3^+$ and $N^- = N \setminus N^+$, and the additional constraints $y_j = 1$ for $j \in J_2$ and $w_j = u_j y_j$ for $j \in J_3$. □

The single node flow set is also the simplest subproblem of a fixed charge network flow problem. It represents the flow conservation constraint in a node

of the network. The variables $x_i, i \in N_1 (N_2$ respectively) represent the flow in each of the arcs $i \in N_1 (N_2$ resp.) entering (leaving resp.) the node. The variables y_i are the so-called set-up variables, indicating whether an arc is "open" or not. An example of such a single node flow set is shown in Figure 3.1 with $n_1 = 3, n_2 = 2, b = 4, a = (3, 4, 5, 2, 3)$ and $u = 1$.

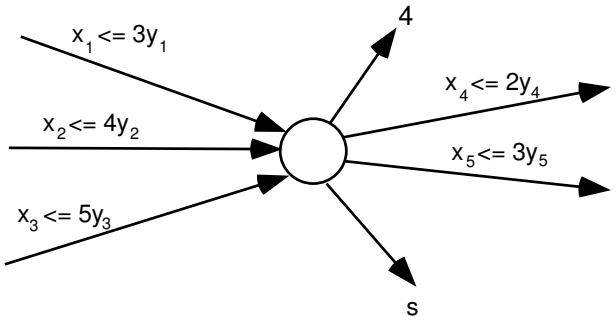

Figure 3.1: An example of Single Node Flow Set

The MIR approach Throughout we will use a standard approach to generate valid inequalities for the single node flow set X^N and its variants, which is a minor modification of the $c - MIR$ approach in [46].

Step 1 Using slack variables for the variable upper bound constraints, relax X^N to obtain a knapsack set with a continuous variable X^{KC}.
Step 1b (Optional) Fix the values of some variables giving a restricted set X^{KC-F}.
Step 2 Complement certain integer variables - those in an appropriately chosen "cover".
Step 3 Rescale the row.
Step 4 Generate a mixed integer rounding inequality for X^{KC} or X^{KC-F}, and rescale the inequality.
Step 4b If variables have been fixed in Step 1b, calculate the lifting function ϕ^1. If the lifting function is not superadditive on some appropriate domain, look for a valid superadditive lifting function $\hat{\phi}$. Generate a valid inequality for X^{KC}.
Step 5 By complementing again, and eliminating the slack variables introduced in Step 1, generate a valid inequality for X^N.

This chapter is organized as follows. The next two sections show how to generate valid inequalities using the MIR approach, for the 0-1 and integer single node flow sets respectively. In these sections, it is also shown how to

strengthen these valid inequalities. In the last section, we finally discuss particular cases of the single node flow sets and show that the projection of the flow cover inequalities provide sets of known valid inequalities for these smaller sets.

3.2 The 0-1 Single Node Flow Set

3.2.1 Generating the MIR flow cover inequalities

Consider the set $X^N(n_1, n_2, b, a, 1)$. We now use the MIR approach described above to derive basic valid inequalities for this set.

Definition 3.2 *(C_1, C_2) is a flow cover for X^N if*
i) $C_1 \subseteq N_1, C_2 \subseteq N_2$
ii) $\sum_{j\in C_1} a_j - \sum_{j\in C_2} a_j - b = \lambda > 0$.

Proposition 3.1 *Suppose that (C_1, C_2) is a flow cover, and choose $\bar{a} \in \mathbb{R}^1_+$ with $\bar{a} > \lambda$. Then the MIR flow cover inequality*

$$\begin{aligned}
&\sum_{j\in C_1} \{x_j + [a_j + \lambda F(-\frac{a_j}{\bar{a}})](1 - y_j)\} \\
&\quad + \sum_{j\in L_1} x_j - \sum_{j\in L_1} [a_j - \lambda F(\frac{a_j}{\bar{a}})]y_j \\
&\leq b + \sum_{j\in C_2} a_j - \sum_{j\in C_2} \lambda F(\frac{a_j}{\bar{a}})(1 - y_j) \\
&\quad - \sum_{j\in L_2} \lambda F(-\frac{a_j}{\bar{a}})y_j + \sum_{j\in R_2} x_j + s \qquad (3.1)
\end{aligned}$$

is valid for X^N, where (C_i, L_i, R_i) is a partition of N_i for $i = 1, 2$ and $F = F_\alpha$, with $\alpha = \frac{\bar{a}-\lambda}{\bar{a}}$, is the mixed integer function defined in Definition 1.8.

Proof:
Step 1 Starting from the inequality

$$\sum_{j\in N_1} x_j - \sum_{j\in N_2} x_j \leq b + s,$$

we introduce variables $t_j = a_j y_j - x_j$ for $j \in C_1 \cup L_1 \cup C_2 \cup L_2$. Using the nonnegativity of x_j for $j \in R_1$ and of t_j for $j \in C_2 \cup L_2$ gives the relaxation

$$\sum_{j\in C_1\cup L_1} a_j y_j - \sum_{j\in C_2\cup L_2} a_j y_j - \sum_{j\in R_2} x_j \leq b + \sum_{j\in C_1\cup L_1} t_j + s.$$

Step 2 Now introducing variables $\bar{y}_j = 1 - y_j$ for $j \in C_1 \cup C_2$, we obtain

$$-\sum_{j\in C_1} a_j\bar{y}_j + \sum_{j\in L_1} a_j y_j + \sum_{j\in C_2} a_j\bar{y}_j - \sum_{j\in L_2} a_j y_j \le -\lambda + \sum_{j\in R_2} x_j + \sum_{j\in C_1\cup L_1} t_j + s.$$

Step 3 We now divide by $\bar{a} > \lambda$.
Step 4 Generate the mixed integer rounding inequality giving

$$\sum_{j\in C_1} F(-\frac{a_j}{\bar{a}})\bar{y}_j + \sum_{j\in L_1} F(\frac{a_j}{\bar{a}})y_j + \sum_{j\in C_2} F(\frac{a_j}{\bar{a}})\bar{y}_j + \sum_{j\in L_2} F(-\frac{a_j}{\bar{a}})y_j \le -1 + \frac{1}{\lambda}(s + \sum_{j\in R_2} x_j + \sum_{j\in C_1\cup L_1} t_j).$$

Step 5 Multiplying by λ, and restating the inequality in terms of the original variables gives the required inequality. □

Observation 3.2 *i)* $\lambda F(-\frac{a_j}{\bar{a}}) \ge -\max[\min[\lambda, a_j], a_j - (\bar{a} - \lambda)]$ *with equality for* $a_j \le \bar{a} + \lambda$.
ii) $\lambda F(-\frac{a_j}{\bar{a}}) = -\min[\lambda, a_j]$ *for* $a_j \le \bar{a}$.
iii) $\lambda F(\frac{a_j}{\bar{a}}) \ge \min[\max[(a_j - (\bar{a} - \lambda), 0], \lambda]$ *with equality for* $a_j \le 2\bar{a} - \lambda$.
iv) $\lambda F(\frac{a_j}{\bar{a}}) = \max[(a_j - (\bar{a} - \lambda), 0]$ *if* $a_j \le \bar{a}$.

Corollary 3.2 *If* $\bar{a} = \max_{j\in C_1} a_j$, *the MIR flow cover inequality (3.1) takes the form*

$$\sum_{j\in C_1}\{x_j + [a_j - \lambda]^+(1 - y_j)\} + \sum_{j\in L_1} x_j - \sum_{j\in L_1}[a_j - \lambda F(\tfrac{a_j}{\bar{a}})]y_j \le b + \sum_{j\in C_2} a_j - \sum_{j\in C_2} \lambda F(\tfrac{a_j}{\bar{a}})(1 - y_j) - \sum_{j\in L_2} \lambda F(-\tfrac{a_j}{\bar{a}})y_j + \sum_{j\in R_2} x_j + s$$

and is at least as strong as the GFC2 inequality [58]

$$\sum_{j\in C_1} x_j + \sum_{j\in C_1}[a_j - \lambda]^+(1 - y_j) + \sum_{j\in L_1} x_j - \sum_{j\in L_1}(\max[a_j, \bar{a}] - \lambda)y_j \le b + \sum_{j\in C_2} a_j - \sum_{j\in C_2} \min[\lambda, (a_j - (\bar{a} - \lambda))^+](1 - y_j) + \sum_{j\in L_2} \max[a_j - (\bar{a} - \lambda), \lambda]y_j + \sum_{j\in R_2} x_j + s.$$

Corollary 3.3 *If* $\bar{a} = \max_{j\in C_1\cap L_2} a_j$ *and* $a_j > \lambda$ *for all* $j \in L_2$, *the MIR flow cover inequality (3.1) takes the form*

$$\sum_{j\in C_1}\{x_j + [a_j - \lambda]^+(1 - y_j)\} + \sum_{j\in L_1} x_j - \sum_{j\in L_1}[a_j - \lambda F(\frac{a_j}{\bar{a}})]y_j \le b + \sum_{j\in C_2} a_j - \sum_{j\in C_2} \lambda F(\frac{a_j}{\bar{a}})(1 - y_j) + \sum_{j\in L_2} \lambda y_j + \sum_{j\in R_2} x_j + s \qquad (3.2)$$

and is at least as strong as the GFC1 inequality [58]

$$\sum_{j\in C_1}\{x_j+[a_j-\lambda]^+(1-y_j)\}\le b+\sum_{j\in C_2}a_j+\sum_{j\in L_2}\lambda y_j+\sum_{j\in R_2}x_j+s.$$

3.2.2 A Strengthened MIR Flow Cover Inequality

Now we strengthen the inequality.

Proposition 3.4 *Suppose that (C_1, C_2) is a flow cover and $\bar{a} = \max_{j\in C_1\cup L_2} a_j > \lambda$. Then the lifted inequality*

$$\begin{aligned}&\sum_{j\in C_1}\{x_j+[a_j-\lambda]^+(1-y_j)\}\\&\quad+\sum_{j\in L_1}[x_j-(a_j-\phi^1(a_j))y_j]\\&\le b+\sum_{j\in C_2}a_j-\sum_{j\in C_2}\phi^1(a_j)(1-y_j)\\&\qquad-\sum_{j\in L_2}\lambda y_j+\sum_{j\in R_2}x_j+s\end{aligned}\tag{3.3}$$

is valid for X^N, where (C_i, L_i, R_i) is a partition of N_i for $i = 1, 2$ and $\phi^1 = G_{a,\lambda}$ on $\mathbb{R}^1_+$ with $a = (a^{C_1}, a^{L_2})$.

Proof: Proceeding as in the proof of Proposition 3.1, we modify as follows:
Step 1b Set $y_j = 0$ for $j \in L_1$ and $\bar{y}_j = 0$ for $j \in C_2$.
Step 2 The restricted set takes the form:

$$-\sum_{j\in C_1}a_j\bar{y}_j-\sum_{j\in L_2}a_jy_j\le-\lambda+\sum_{j\in R_2}x_j+\sum_{j\in C_1\cup L_1}t_j+s.$$

Step 3. Divide by $\bar{a} = \max_{j\in C_1\cup L_2} a_j$.
Step 4. Generate the MIR inequality and multiply by λ

$$\lambda\sum_{j\in C_1}F(-\frac{a_j}{\bar{a}})\bar{y}_j+\lambda\sum_{j\in L_2}F(-\frac{a_j}{\bar{a}})y_j\le\lambda F(-\frac{\lambda}{\bar{a}})+\lambda\frac{\sigma/\bar{a}}{\lambda/\bar{a}},$$

or, by using Observation 3.2,

$$-\sum_{j\in C_1}\min[\lambda,a_j]\bar{y}_j-\sum_{j\in L_2}\lambda y_j\le-\lambda+\sigma$$

where $\sigma = \sum_{j\in R_2} x_j + \sum_{j\in C_1\cup L_1} t_j + s$, and when we relax the coefficients of y_j, $j \in L_2$.
Step 4b Calculate the lifting function

$$\begin{array}{c}\phi^1(u) = \min \sum_{j\in C_1} \min[\lambda, a_j]\bar{y}_j + \sum_{j\in L_2} \lambda y_j - \lambda + \sigma,\\ -\sum_{j\in C_1} a_j\bar{y}_j - \sum_{j\in L_2} a_j y_j - \sigma \le -\lambda - u\\ \bar{y}_j \in \{0,1\} \text{ for } \ j \in C_1,\ y_j \in \{0,1\} \text{ for } \ j \in L_2,\ \sigma \ge 0.\end{array}$$

It can be shown that on $\mathbb{R}^1_+$, ϕ^1 is precisely the superadditive function $G_{a,\lambda}$ with $a = (a^{C_1}, a^{L_2})$.

Lift to obtain the inequality

$$-\sum_{j\in C_1} \min[\lambda, a_j]\bar{y}_j - \sum_{j\in L_2} \lambda y_j + \sum_{j\in L_1} G_{a,\lambda}(a_j) y_j + \sum_{j\in C_2} G_{a,\lambda}(a_j)\bar{y}_j \le -\lambda + \sigma.$$

Step 5. Uncomplement variables and substitute for t_j. □

Example: Consider the single node flow set

$$\begin{array}{c}x_1 + x_2 - x_3 + x_4 + x_5 - x_6 \le -8 + s\\ x_1 \le 10y_1, x_2 \le 9y_2, x_3 \le 7y_3, x_4 \le 16y_4, x_5 \le 5y_5, x_6 \le 19y_6\\ x \in \mathbb{R}^6_+, s \in \mathbb{R}^1_+, y \in [0,1]^6.\end{array}$$

Taking as flow cover $C_1 = \{1,2\}, C_2 = \{6\}$, we obtain $\lambda = 10 + 9 - 19 + 8 = 8$.
With $L_1 = \{4\}, L_2 = \emptyset$, we take $\bar{a} = \max_{j\in C_1\cup L_2} a_j = 10$, $\alpha = \frac{10-8}{10}$.
The resulting MIR flow cover inequality (3.2) is

$$x_1 + 2(1 - y_1) + x_2 + 1(1 - y_2) - x_3 + x_4 - 4y_4 \le 11 - 15(1 - y_6) + s.$$

To obtain the strengthened inequality (3.3), we calculate the lifting function ϕ^1. With $a = (10, 9), \lambda = 8$

$$G_{a,\lambda}(u) = \begin{cases} 0 & if \quad 0 \le u \le 2\\ u - 2 & if \quad 2 \le u \le 10\\ 8 & if \quad 10 \le u \le 11\\ u - 3 & if \quad 11 \le u. \end{cases}$$

As $G_{a,\lambda}(16) = 13$ and $G_{a,\lambda}(19) = 16$, we obtain the inequality

$$x_1 + 2(1 - y_1) + x_2 + 1(1 - y_2) - x_3 + x_4 - (16 - 13)y_4 \le 11 - 16(1 - y_6) + s$$

which in this case is stronger than the MIR inequality. □

3.2.3 The MIR Reverse Flow Cover Inequality

We now present an explicit expression for the reverse flow cover inequality for this set. This inequality is obtained by applying the results of the previous subsection to the single node flow set in which the directions of the flows are all reversed.

Definition 3.3 *(T_1, T_2) is a reverse flow cover for X^N if*
i) $T_1 \subseteq N_1, T_2 \subseteq N_2$
ii) $\sum_{j\in T_1} a_j - \sum_{j\in T_2} a_j - b = -\mu < 0$.

Proposition 3.5 *Suppose that (T_1, T_2) is a reverse flow cover and $\bar{a} > \mu$. Then the MIR reverse flow cover inequality*

$$\begin{aligned}
&\sum_{j\in T_1} x_j + \sum_{j\in T_1} \mu F(\frac{a_j}{\bar{a}})](1-y_j) \\
&\quad + \sum_{j\in L_1} x_j + \sum_{j\in L_1} \mu F(-\frac{a_j}{\bar{a}})]y_j \\
&\leq \sum_{j\in T_1} a_j - \sum_{j\in T_2} [a_j + \mu F(-\frac{a_j}{\bar{a}})](1-y_j) \\
&\quad + \sum_{j\in L_2} [a_j - \mu F(\frac{a_j}{\bar{a}})]y_j + \sum_{j\in R_2} x_j + s. \qquad (3.4)
\end{aligned}$$

is valid for X^N, where (T_i, L_i, R_i) is a partition of N_i for $i = 1, 2$ and $F = F_\alpha$ with $\alpha = \frac{\bar{a}-\mu}{\bar{a}}$.

Corollary 3.6 *If $\bar{a} = \max_{j\in T_2} a_j$, the MIR reverse flow cover takes the form*

$$\begin{aligned}
&\sum_{j\in T_1} x_j + \sum_{j\in T_1} \mu F(\frac{a_j}{\bar{a}})(1-y_j) + \sum_{j\in L_1} x_j + \sum_{j\in L_1} \mu F(-\frac{a_j}{\bar{a}})]y_j \\
&\leq \sum_{j\in T_1} a_j - \sum_{j\in T_2} (a_j - \mu)^+(1-y_j) + \sum_{j\in L_2} [a_j - \mu F(\frac{a_j}{\bar{a}})]y_j + \sum_{j\in R_2} x_j + s,
\end{aligned}$$

and is at least as strong as the inequality

$$\begin{aligned}
&\sum_{j\in T_1} x_j + \sum_{j\in T_1} \min[a_j - (\bar{a}-\mu)^+, \mu](1-y_j) \\
&+ \sum_{j\in L_1} x_j - \sum_{j\in L_1} \max[\min(\mu, a_j), a_j - (\bar{a}-\mu)]y_j \\
&\leq \sum_{j\in T_1} a_j - \sum_{j\in T_2} (a_j - \mu)^+(1-y_j) \\
&+ \sum_{j\in L_2} [\max(a_j - \mu, \min\{a_j, \bar{a}-\mu\}]y_j + \sum_{j\in R_2} x_j + s,
\end{aligned}$$

obtained by fixing the variable lower bounds to zero in the inequalities of Stallaert [57].

Corollary 3.7 *If $\bar{a}$ is large, the MIR reverse flow cover takes the form*

$$\sum_{j\in T_1} x_j + \sum_{j\in L_1}(x_j - \mu y_j) \le \sum_{j\in T_1} a_j - \sum_{j\in T_2}(a_j-\mu)^+(1-y_j) + \sum_{j\in L_2\cup R_2} x_j + s.$$

3.2.4 Further Remarks

The main difference between the MIR approach proposed here and the $c-MIR$ approach in [47] is the role of the cover in determining which variables to complement. None of the inequalities proposed in this section is really new. In particular the strengthened MIR flow cover inequality is essentially derived in [46] and can also be seen as a special case of the LSGFCI inequality in [26] when there is an unbounded continuous variable. Also as remarked above, the reverse flow cover inequalities are nothing but flow cover inequalities, and arc reversal was already used in [59].

3.3 Integer Single Node Flow Sets

3.3.1 The MIR Flow Cover Inequality

Consider now the set $X^N(n_1, n_2, b, a, u)$ with $u_j > 1$ for some $j \in N$. We now derive an MIR flow cover inequality for this set.

Definition 3.4 (C_1, C_2) *is an* integer flow cover *for* X^N *if*
i) $C_1 \subseteq N_1, C_2 \subseteq N_2$
ii) there exists $k \in C_1$ *such that* $\sum_{j\in C_1\setminus k} a_j u_j - \sum_{j\in C_2} a_j u_j < b$ *and there exists unique values* λ *and* η_k *such that*

$$a_k\eta_k + \sum_{j\in C_1\setminus k} a_j u_j - \sum_{j\in C_2} a_j u_j = b + \lambda$$

with $0 < \lambda < a_k$, *and* $\eta_k \in \mathbb{Z}^1$ *with* $1 \le \eta_k \le u_k$.

Proposition 3.8 *Suppose that* (C_1, C_2) *is an integer flow cover. Then the integer flow cover inequality*

$$\begin{aligned}\sum_{j\in C_1} x_j + (a_k-\lambda)(\eta_k - y_k) + \sum_{j\in C_1\setminus k}[a_j + \lambda F(-\frac{a_j}{a_k})](u_j - y_j)& \\ + \sum_{j\in L_1} x_j - \sum_{j\in L_1}[a_j - \lambda F(\frac{a_j}{a_k})]y_j& \\ \le b + \sum_{j\in C_2} a_j u_j - \sum_{j\in C_2}\lambda F(\frac{a_j}{a_k})(u_j - y_j)& \\ - \sum_{j\in L_2}\lambda F(-\frac{a_j}{a_k})y_j + \sum_{j\in R_2} x_j + s& \qquad (3.5)\end{aligned}$$

is valid for X^N, where (C_i, L_i, R_i) is a partition of N_i for $i = 1, 2$ and $F = F_\alpha$ with $\alpha = \frac{a_k - \lambda}{a_k}$.

Proof: We again use the MIR approach from the previous section. Starting from the inequality

$$\sum_{j \in N_1} x_j - \sum_{j \in N_2} x_j \leq b + s,$$

we introduce variables $t_j = a_j y_j - x_j$ for $j \in C_1 \cup L_1 \cup C_2 \cup L_2$. Using the nonnegativity of x_j for $j \in R_1$ gives the relaxation

$$\sum_{j \in C_1 \cup L_1} a_j y_j - \sum_{j \in C_2 \cup L_2} a_j y_j - \sum_{j \in R_2} x_j \leq b + \sum_{j \in C_1 \cup L_1} t_j + s.$$

Now introducing variables $\bar{y}_j = u_j - y_j$ for $j \in C_1 \cup C_2$, we obtain

$$\begin{aligned} &- \textstyle\sum_{j \in C_1} a_j \bar{y}_j + \sum_{j \in L_1} a_j y_j + \sum_{j \in C_2} a_j \bar{y}_j - \sum_{j \in L_2} a_j y_j \\ &\quad \leq a_k(\eta_k - u_k) - \lambda + \textstyle\sum_{j \in R_2} x_j + \sum_{j \in C_1 \cup L_1} t_j + s. \end{aligned} \tag{3.6}$$

We now divide by a_k, and then generate the mixed integer rounding inequality giving

$$\begin{aligned} \sum_{j \in C_1} F(-\frac{a_j}{a_k})\bar{y}_j + \sum_{j \in L_1} F(\frac{a_j}{a_k}) y_j + \sum_{j \in C_2} F(\frac{a_j}{a_k})\bar{y}_j + \sum_{j \in L_2} F(-\frac{a_j}{a_k}) y_j \\ \leq (\eta_k - u_k) - 1 + \frac{1}{\lambda}(s + \sum_{j \in R_2} x_j + \sum_{j \in C_1 \cup L_1} t_j). \end{aligned}$$

Multiplying by λ, and restating the inequality in terms of the original variables gives the required inequality. □

3.3.2 Strengthening the Integer Flow Cover Inequality

To obtain stronger inequalities, we use results of Atamturk [6] on integer knapsack sets. We start from inequality (3.6) in the proof of validity of Proposition 3.8 which we write more compactly, after recomplementing $\bar{y}_k$, as

$$\begin{aligned} a_k z_k - \textstyle\sum_{j \in I^-} a_j z_j + \sum_{j \in I^+} a_j z_j \leq a_k \eta_k - \lambda + \sigma \\ z_j \leq u_j, z_j \in \mathbb{Z}^1_+ \text{ for } j \in \{k\} \cup I^- \cup I^+, \sigma \in \mathbb{R}^1_+. \end{aligned}$$

where $I^- = C_1 \setminus \{k\} \cup L_2, I^+ = C_2 \cup L_1, \sigma = s + \sum_{j \in R_2} x_j + \sum_{j \in C_1 \cup L_1} t_j$ and z_j represents either y_j or $\bar{y}_j$ as appropriate.

Setting $z_j = 0$ for $j \in I^- \cup I^+$, leads to the reduced system

$$\begin{aligned} a_k z_k - \sigma \leq a_k \eta_k - \lambda \\ z_k \leq u_k, z_k \in \mathbb{Z}^1_+, \sigma \in \mathbb{R}^1_+ \end{aligned}$$

with valid inequality

$$\lambda z_k \leq \lambda(\eta_k - 1) + \sigma.$$

The lifting function for this inequality is easily calculated, is identical to $\lambda F_{\frac{a_k-\lambda}{a_k}}$ around the origin, and explicitly takes into account the upper and lower bounds on z_k:

$$\phi(v) = \begin{cases} (\eta_k - u_k - 1)\lambda & if \quad v \leq a_k(\eta_k - u_k) - \lambda \\ (j-1)\lambda + [v - (ja_k - \lambda)] & if \quad ja_k - \lambda \leq v \leq ja_k, \\ & \quad j = (\eta_k - u_k), \ldots, \eta_k - 1 \\ j\lambda & if \quad ja_k \leq v \leq (j+1)a_k - \lambda, \\ & \quad j = (\eta_k - u_k), \ldots, \eta_k - 1 \\ (\eta_k - 1)\lambda + [v - (\eta_k a_k - \lambda)] & if \quad \eta_k a_k - \lambda \leq v. \end{cases}$$

This function is superadditive on $\mathbb{R}^1_+$ and it is also superadditive on $\mathbb{R}^1_-$. Here we just consider the case where we first lift in the variables in I^-, and then the variables in I^+. Because ϕ is superadditive on $\mathbb{R}^1_-$, the lifting function does not change as these variables in I^- are lifted in. After lifting in the variables in I^-, we obtain the lifting function ϕ^+. This function turns out to be superadditive on $\mathbb{R}^1_+$. In the general case, we obtain a valid lifting function H^+ superadditive on $\mathbb{R}^1_+$ by dropping the nonnegativity constraint on z_k in the mixed integer program defining ϕ^+, namely

$$\begin{aligned} H^+(v) = \min\{\lambda(\eta_k - 1) - \lambda z_k + \sigma - \sum_{j \in I^-} \phi(-a_j) z_j : \\ a_k z_k - \sum_{j \in I^-} a_j z_j - \sigma \leq a_k \eta_k - \lambda - v, \\ z_k \leq u_k, z_k \in \mathbb{Z}^1, \ z_j \leq u_j, z_j \in \mathbb{Z}^1_+ \text{ for } j \in I^-, \sigma \in \mathbb{R}^1_+.\} \end{aligned}$$

To describe H^+ (and ϕ^+), we define an ordering on the indices in I^-, $i_1, \cdots, i_{|I^-|}$ such that $a_{i_1} \geq a_{i_2} \geq \cdots \geq a_{i_{|I^-|}}$. We also define the set $I^{--} = \{i : i \in I^- \text{ and } a_i \geq a_k(u_k - \eta_k + 1)\}$ with $\bar{r} = \max\{r : i_r \in I^{--}\}$. We also define $\rho_r = a_k(\eta_k - u_k) - \lambda + a_{i_r}$ for $r = 1, \ldots, \bar{r}$. Finally, for an index $r \leq \bar{r}$, we consider two types of aggregate, $U_r = u_{i_1} + \cdots + u_{i_r}$, and $M_r = a_{i_1} u_{i_1} + \cdots + a_{i_r} u_{i_r}$.

It is not difficult to see that as v increases from 0, the mixed integer program defining $H^+(v)$ has optimal solutions in which the variables $i_1, \ldots, i_r \in I^{--}$ are used in that order. Thus when $z_{i_s} = u_{i_s}$ for $s < r$ and $z_{i_r} = t$, $j = u_k - z_k$ takes values increasing from 0 to $u_k - \eta_k$. Once all of the variables in I^{--} are

at their upper bound, z_k then takes negative values. Specifically $H^+(v) =$

$$\begin{cases} (u_k - \eta_k + 1)\lambda(U_{r-1} + t) & \text{if } M_{r-1} + ta_{i_r} \le v \le M_{r-1} + ta_{i_r} + \rho_r \\ & \quad r = 1, \ldots, \bar{r},\ t = 0, \cdots, u_{i_r} - 1 \\ (u_k - \eta_k + 1)\lambda(U_{r-1} + t) + j\lambda & \text{if } M_{r-1} + ta_{i_r} + \rho_r + ja_k \le v \le \\ \quad + v - M_{r-1} - ta_{i_r} - ja_k - \rho_r & \quad M_{r-1} + ta_{i_r} + \rho_r + ja_k + \lambda \\ & r = 1, \ldots, \bar{r},\ t = 0, \cdots, u_{i_r} - 1, \\ & j = 0, \cdots, u_k - \eta_k \\ (u_k - \eta_k + 1)\lambda(U_{r-1} + t) & \text{if } M_{r-1} + ta_{i_r} + \rho_r + ja_k + \lambda \le v \le \\ \quad + (j+1)\lambda & \quad M_{r-1} + ta_{i_r} + \rho_r + (j+1)a_k \\ & r = 1, \ldots, \bar{r},\ t = 0, \cdots, u_{i_r} - 1, \\ & j = 0, \cdots, u_k - \eta_k - 1 \\ (u_k - \eta_k + 1)\lambda U_{\bar{r}} + j\lambda & \text{if } M_{\bar{r}} + ja_k \le v \le M_{\bar{r}} + (j+1)a_k - \lambda \\ & \quad j = 0, \cdots \\ (u_k - \eta_k + 1)\lambda U_{\bar{r}} + (j-1)\lambda & \text{if } M_{\bar{r}} + ja_k - \lambda \le v \le M_{\bar{r}} + ja_k \\ \quad + v - M_{\bar{r}} - ja_k + \lambda & \quad j = 1, \cdots \end{cases}$$

An example of the valid superadditive lifting function H^+ and an exact lifting function ϕ^+ are depicted in Figure 3.2. Finally H^+ can be used to lift

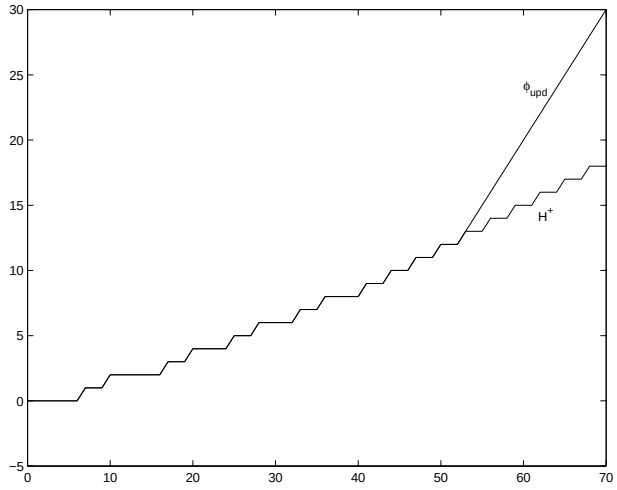

Figure 3.2: Lifting functions H^+ and ϕ^+

in all the variables in I^+.

Proposition 3.9 *The inequality*

$$\begin{array}{c}\lambda y_k + \sum_{j\in C_1\setminus\{k\}} \phi(-a_j)(1-y_j) + \sum_{j\in L_2} \phi(-a_j)y_j \\ + \sum_{j\in C_2} H^+(a_j)(1-y_j) + \sum_{j\in L_1} H^+(a_j)y_j \\ \leq \lambda(\eta_k - 1) + \sum_{j\in R_2} x_j + \sum_{j\in C_1\cup L_1}(a_j y_j - x_j) + s\end{array}$$

is valid for X^N.

In [6], similar functions are also calculated for the case when one first lifts variables in I^+, and then those in I^-, which leads to another family of strong inequalities.

Example: Consider the set $X^N(2,2,4,(3,4,5,2),(2,3,2,3))$, namely

$$\begin{array}{c}x_1 + x_2 - x_3 - x_4 \leq 4 + s \\ x_1 \leq 3y_1, x_2 \leq 4y_2, x_3 \leq 5y_3, x_4 \leq 2y_4 \\ y_1 \leq 2, y_2 \leq 3, y_3 \leq 2, y_4 \leq 3 \\ x \in \mathbb{R}^4_+, y \in \mathbb{Z}^4_+, s \in \mathbb{R}^1_+.\end{array}$$

Suppose that one wishes to cut off the fractional solution

$$x^* = (2, 12, 10, 0), y^* = (\frac{2}{3}, 3, 2, 0), s^* = 0.$$

$C = \{1,2,3\}$ with $k = 1$ is an integer flow cover with $a_k = 3, \lambda = 1$ and $\eta_k = 1$. Inequality (3.5) with $L_1 = L_2 = \emptyset$ gives

$$x_1 + (3-1)(1-y_1) + x_2 + [4 + F_{\frac{2}{3}}(-\frac{4}{3})](3-y_2) \leq 14 - F_{\frac{2}{3}}(\frac{5}{3})](2-y_3) + x_4 + s, \text{ or}$$

$$x_1 + 2(1-y_1) + x_2 + 2(3-y_2) \leq 14 - 1(2-y_3) + x_4 + s$$

which cuts off the fractional point, but is not facet-defining for X^N.

To try to obtain a stronger inequality, we again choose $C_1 = \{1,2\}, C_2 = \{3\}$ as a cover with $k = 1$. Thus we consider

$$3y_1 + 4y_2 - 5y_3 - x_4 \leq 4 + t_1 + t_2 + s.$$

Complementing gives

$$3y_1 - 4\bar{y}_2 + 5\bar{y}_3 - x_4 \leq 2 + t_1 + t_2 + s.$$

Setting $\bar{y}_2 = \bar{y}_3 = 0$ leaves the system

$$3y_1 - x_4 \leq 2 + t_1 + t_2 + s$$

with valid inequality

$$y_1 - x_4 \leq t_1 + t_2 + s.$$

Now the lifting function ϕ is given above. If we first lift in $\bar{y}_2$, $\phi(-4) = -2$ and we obtain

$$y_1 - 2\bar{y}_2 - x_4 \leq t_1 + t_2 + s.$$

Using the superadditive function H^+ to lift the variable $\bar{y}_3$, $H^+(5) = 1$, so we obtain the same inequality as above. If we calculate the exact lifting function function, it turns out that $\phi^+(5) = 2$ giving the inequality

$$x_1 - 2y_1 - 2\bar{y}_2 + 2\bar{y}_3 - x_4 \leq t_2 + s,$$

or

$$x_1 - 2y_1 + x_2 - 2y_2 - 2y_3 - x_4 \leq 2 + s,$$

which is facet-defining. □

3.3.3 The MIR Reverse Flow Cover Inequality

Here we give an explicit formula for the reverse flow cover inequality when the bounds are integer. We modify ii) in Definition 3.4.

Definition 3.5 (T_1, T_2) *is an* integer reverse flow cover *for* X^N *if*
i) $T_1 \subseteq N_1, T_2 \subseteq N_2$
ii) there exists $k \in T_2$ *such that* $\sum_{j\in T_1} a_j u_j - \sum_{j\in T_2\setminus k} a_j u_j > b$ *and there exist unique values* μ *and* η_k *such that*

$$\sum_{j\in T_1} a_j u_j - \sum_{j\in T_2\setminus k} a_j u_j - a_k\eta_k = b - \mu$$

with $0 < \mu < a_k$ *and* $\eta_k \in \mathbb{Z}^1$ *with* $1 \leq \eta_k \leq u_k$.

Proposition 3.10 *Suppose that* (T_1, T_2) *is an integer reverse flow cover for* X^N. *The following inequality*

$$\begin{aligned}
&\sum_{j\in T_1} [x_j + \mu F(\frac{a_j}{a_k})(u_j - y_j)] + \sum_{L_1} [x_j + \mu F(-\frac{a_j}{a_k})y_j] \\
\leq &\sum_{j\in T_1} a_j u_j - (a_k - \mu)(\eta_k - y_k) - \sum_{j\in T_2\setminus k} [a_j + \mu F(-\frac{a_j}{a_k})](u_j - y_j) \\
&+ \sum_{j\in L_2} [a_j - \mu F(\frac{a_j}{a_k})]y_j + \sum_{j\in R_2} x_j + s \qquad (3.7)
\end{aligned}$$

is valid for X^N, *where* (T_i, L_i, R_i) *is a partition of* N_i *for* $i = 1, 2$ *and* $F = F_\alpha$ *with* $\alpha = \frac{a_k - \mu}{a_k}$.

Example: We consider the same set X^N as before,

$$\begin{gathered}
x_1 + x_2 - x_3 - x_4 \leq 4 + s \\
x_1 \leq 3y_1, x_2 \leq 4y_2, x_3 \leq 5y_3, x_4 \leq 2y_4 \\
y_1 \leq 2, y_2 \leq 3, y_3 \leq 2, y_4 \leq 3 \\
x \in \mathbb{R}^4_+, y \in \mathbb{Z}^4_+, s \in \mathbb{R}^1_+.
\end{gathered}$$

Suppose that one wishes to cut off the fractional solution

$$x^* = (0, 12, 2, 6), y^* = (0, 3, \frac{2}{5}, 2), s^* = 0.$$

$T = \{2, 3, 4\}$ with $k = 3$ is an integer reverse flow cover with $a_k = 5, \eta_k = 1$ and $\mu = 3$. Inequality (3.7) with $L_1 = L_2 = \emptyset$ gives

$$x_2 + 3F_{\frac{2}{5}}(\frac{4}{5})(3 - y_2) \leq 12 - (5 - 3)(1 - y_3) - (2 + 3F_{\frac{2}{5}}(-\frac{2}{5})(3 - y_4) + s, \text{ or}$$

$$x_2 + 2(3 - y_2) \leq 12 - 2(1 - y_3) + s,$$

which cuts off the fractional solution, and turns out to be facet-defining for $\text{conv}(X^N)$. □

3.4 Particular cases

As we explained in the previous chapter, we can use the knowledge of simple sets to derive valid inequalities for supersets. But once a complicated set has been studied deeply, its knowledge can be used to generate valid inequalities for all its subsets. In this section, we explore this other direction and summarize the inequalities that can be deduced from the flow cover inequalities presented earlier in the chapter. We also show that some of the well-known inequalities for these sets are nothing but particular cases of MIR flow cover inequalities. In this section, we focus on giving simple inequalities by dropping some complicated terms.

Inflow arcs As a first particular case, we study the set where only entering arcs are considered. We try to find valid inequalities for

$$Y^{\leq} = \{(x, y) \in \mathbb{R}^n_+ \times \{0, 1\}^n : \sum_{j=1}^{n} x_j \leq b + s,\ 0 \leq x_j \leq a_j y_j \text{ for all } j\}. \quad (3.8)$$

By fixing the outflow arcs to 0, we easily get the following proposition.

Proposition 3.11 *Let C be a cover, i.e. such that $\sum_{j \in C} a_j = b + \lambda$, with $\lambda > 0$. The inequality*

$$\sum_{j \in C} x_j + \sum_{j \in C} (a_j - \lambda)^+ (1 - y_j) \leq b + s$$

is valid for (3.8).

Proof: The result follows from Proposition 3.1 by ignoring the variables in N_2, and by relaxing the coefficients of variables in $N \setminus C$. □

Proposition 3.12 *Let T be a reverse cover, i.e. such that $\sum_{j\in T} a_j = b - \mu$ with $\mu > 0$. The inequality*

$$\sum_{j\in T} x_j + \sum_{j\in N\setminus T} (x_j - \mu y_j) \leq \sum_{j\in T} a_j + s$$

is valid for (3.8).

Proof: The result follows from Proposition 3.5, by relaxing the coefficients of $\bar{y}_j$, for $j \in T$ and since $-\mu \leq \mu F(\frac{-a_j}{\bar{a}})$ if $\bar{a} \geq a_j$ for all j. □

Outflow arcs The second special case is the set related to outgoing arcs only. We consider thus

$$Y^{\geq} = \{(x,y) \in \mathbb{R}^n_+ \times \{0,1\}^n : \sum_{j=1}^{n} x_j + s \geq b,\ 0 \leq x_j \leq a_j y_j \text{ for all } j\}. \quad (3.9)$$

If we fix the inflow arcs to 0 in the flow cover inequalities, we obtain again inequalities for this particular case. This is contained in the next two propositions.

Proposition 3.13 *Let C be a cover, i.e. such that $\sum_{j\in C} a_j = b + \lambda$, with $\lambda > 0$. The inequality*

$$\sum_{j\in C} (a_j - \lambda)^+ (1 - y_j) \leq \sum_{j\in N\setminus C} x_j + s$$

is valid for (3.9).

Proof: The set $Y^{\geq}$ can be rewritten as

$$Y^{\geq} = \{(x,y) \in \mathbb{R}^n_+ \times \{0,1\}^n : -\sum_{j=1}^{n} x_j \leq -b + s,\ 0 \leq x_j \leq a_j y_j\}.$$

If $\sum_{j\in C} a_j = b + \lambda$, then $-\sum_{j\in C} a_j = -b - \lambda$. Therefore, C is a reverse cover without any ingoing arcs. Applying Proposition 3.5 yields the result if we suppose that $L_2 = \emptyset$. □

Proposition 3.14 *Let T be a reverse cover, i.e. such that $\sum_{j\in T} a_j = b - \mu$, with $\mu > 0$, and let (T, L, R) be a partition of $N = \{1, \ldots, n\}$. The inequality*

$$\sum_{j\in L} \lambda y_j + \sum_{j\in R} x_j \geq \mu - s$$

is valid for (3.9).

Proof: The set $Y^{\geq}$ can be rewritten as

$$Y^{\geq} = \{(x,y) \in \mathbb{R}^n_+ \times \{0,1\}^n : -\sum_{j=1}^{n} x_j \leq -b+s,\ 0 \leq x_j \leq a_j y_j\}.$$

If $\sum_{j\in T} a_j = b - \mu$, then $-\sum_{j\in T} a_j = -b + \mu$. Therefore T is a cover without any ingoing arcs. Applying Corollary 3.3 and relaxing the coefficients of $\bar{y}_j$ for $j \in T$ provides the result. □

Continuous knapsack The continuous knapsack was studied by Marchand and Wolsey [46]. We show here that some valid inequalities for it can be found as a particular case of basic flow covers. The set studied here is

$$Y^{CK} = \{(y,s) \in \{0,1\}^n \times \mathbb{R}_+ : \sum_{j=1}^{n} a_j y_j \leq b + s\}. \tag{3.10}$$

Proposition 3.15 *Let C be a cover, i.e. $\sum_{j\in C} a_j = b + \lambda$, with $\lambda > 0$. The inequality*

$$s + \sum_{j\in C} \min\{\lambda, a_j\}(1 - y_j) \geq \lambda$$

is valid for (3.10).

Proof: Remark that

$$Y^{CK} = Y^{\leq} \cap \{x_j = a_j y_j \text{ for all } j\}.$$

Therefore a valid inequality for $Y^{\leq}$ is also valid for Y^{CK} when one replaces x_j by $a_j y_j$ in the inequality. Applying Proposition 3.11, we obtain

$$\sum_{j\in C} a_j y_j + \sum_{j\in C} (a_j - \lambda)^+ (1 - y_j) \leq b + s$$

$$\sum_{j\in C} a_j - \sum_{j\in C} a_j (1 - y_j) + \sum_{j\in C} (a_j - \lambda)^+ (1 - y_j) \leq b + s$$

$$\lambda \leq s + \sum_{j\in C} (a_j - (a_j - \lambda)^+)(1 - y_j)$$

$$\lambda \leq s + \sum_{j\in C} \min\{\lambda, a_j\}(1 - y_j)$$

which is the expected result. □

Proposition 3.16 *Let T be a reverse cover, i.e. $\sum_{j\in T} a_j = b - \mu$, with $\mu > 0$, the inequality*

$$\sum_{j\in N\setminus T} (a_j - \mu)^+ y_j \leq \sum_{j\in T} a_j (1 - y_j) + s$$

is valid for (3.10).

Proof: We still have

$$Y^{CK} = Y^{\leq} \cap \{x_j = a_j y_j \text{ for all } j\}.$$

Applying Proposition 3.12 and replacing x_j by $a_j y_j$ yields

$$\sum_{j \in T} a_j y_j + \sum_{j \in N \setminus T} (a_j y_j - \mu y_j) \leq \sum_{j \in T} a_j + s$$
$$\sum_{j \in N \setminus T} (a_j - \mu) y_j \leq \sum_{j \in T} a_j (1 - y_j) + s.$$

Note that if $(a_j - \mu) < 0$, we can replace $(a_j - \mu)y_j$ by 0 (because then we can choose to put j in R^+ in Proposition 3.5). This remark provides the expected result. □

Binary knapsack We now turn to one of the simplest set that can also be viewed as a particular case of the single node flow set. Consider

$$Y^{BK} = \{y \in \{0,1\}^n : \sum_{j=1}^{n} a_j y_j \leq b\}. \tag{3.11}$$

This set is obtained by fixing s to 0 in Y^{CK}. Hence we can obtain valid inequalities by applying Propositions 3.15 and 3.16.

Proposition 3.17 *If C is a cover, i.e. $\sum_{j \in C} a_j = b + \lambda$, with $\lambda > 0$, then*

$$\sum_{j \in C} \min\{\lambda, a_j\}(1 - y_j) \geq \lambda \tag{3.12}$$

is a valid inequality for (3.11)*. If C is a minimal cover, then $a_j \geq \lambda$ for all j and* (3.12) *can be written as $\sum_{j \in C}(1 - y_j) \geq 1$, or in the familiar form*

$$\sum_{j \in C} y_j \leq |C| - 1.$$

Proof: Straightforward. □

Proposition 3.18 *If T is a reverse cover, i.e. $\sum_{j \in T} a_j = b - \mu$ with $\mu > 0$, then*

$$\sum_{j \in N \setminus T} (a_j - \mu)^+ y_j \leq \sum_{j \in T} a_j (1 - y_j)$$

is a valid inequality for (3.11)*.*

Chapter 4

Reformulations of group relaxations

4.1 Introduction

In the two previous chapters, we have focused on generating strong valid inequalities for the branch-and-cut algorithm. In this chapter, we propose alternatives to it based on other relaxations and reformulations. The new relaxation proposed is the group relaxation introduced by Gomory in 1969. We presented an example of a group relaxation in Section 1.2.7. Let us now recall the principle. We start with a linear integer program

$$\begin{array}{ll} \max & c^T x \\ \text{s. t.} & Ax = b \\ & x \in \mathbb{Z}^n_+, \end{array} \tag{4.1}$$

with $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$. We partition the set of variables $N = \{1,\ldots,n\}$ into two disjoint sets of basic and non-basic variables, $N = (B, \bar{B})$, with $|B| = m$. We can therefore write (4.1) as

$$\begin{array}{ll} \max & \sum_{j\in B} c_j x_j + \sum_{j\in \bar{B}} c_j x_j \\ \text{s. t.} & A_B x_B + A_{\bar{B}} x_{\bar{B}} = b \\ & x \in \mathbb{Z}^n_+. \end{array} \tag{4.2}$$

If we relax the nonnegativity requirements on the basic variables, the set of feasible non basic variables becomes, if we suppose that A_B is nonsingular,

$$Y(A_B^{-1}b) = \{x_{\bar{B}} \in \mathbb{Z}_+^{|\bar{B}|} : A_B^{-1}A_{\bar{B}}x_{\bar{B}} \equiv A_B^{-1}b \,(\text{mod } 1)\}. \tag{4.3}$$

We want now to study this set $Y(A_B^{-1}b)$. A complete study of the polyhedral structure of sets of type (4.3) has been carried out by Gomory in 1969 [19].

We propose here to study the set differently and in particular, we would like to use the information of sets of type (4.3) in a primal method like the Integral Basis Method proposed by Haus, Köppe and Weismantel [29]. Let us recall briefly what are the main ideas of the algorithm. The Integral Basis Method is a primal simplex algorithm based on a sequence of feasible all-integer tableaux. Rather than adding cutting planes, they use an *extended formulation* of the problem involving additional variables. The extended formulation is typically obtained from a row (or a relaxation of a row) of the all-integer tableau

$$X = \{x_N \in \mathbb{Z}_+^{|N|} : \sum_{j \in N} \bar{a}_j x_j \leq \bar{b}\}$$

with a distinguished variable x_1 for which $\bar{a}_1 > \bar{b}$. The set X is then replaced by an extended formulation

$$Y = \{x_N : x_N = C\lambda, \lambda \geq 0 \text{ and integer}\}$$

where $X \subseteq Y$ and the columns of C form a generating set for Y. This has two possible advantages. The main objective is to find an augmenting vector allowing one to move to an improved primal feasible solution, and the columns of C or simple integer combinations of these columns should provide good candidates for augmentation. The second advantage is that, after addition of the extended formulation, the new linear relaxation is stronger than before.

We suggest to use a group relaxation as a way to add new variables to the primal tableau. Therefore we propose to study different ways to produce extended formulations of a group relaxation. In the following sections of this chapter, we examine such sets. In Section 4.2 we present four straightforward (but probably not well-known) extended formulations for the group relaxation. The first one uses the concept of *irreducible solutions* of the group problem. We call it the *disaggregated formulation*. The second one is a first generalization where we reduce the number of new variables by aggregating variables with identical residue class. We call it the *aggregated formulation*. The third reformulation is based on an *advanced aggregation* technique that reduces even further the number of new variables. Finally we present a reformulation, related to [61], based on the representation of groups by paths in a digraph. Section 4.3 analyzes the different formulations. We show that the extended, the aggregated and the path reformulation satisfy the convex hull property, i.e. their projections onto the space of original variables coincide with the convex hull of the group problem. We also show how to generate valid inequalities tightening the advanced aggregation formulation. In Section 4.4 we address the question of how to generate the complete sets of irreducible solutions used in the extended formulations proposed in Section 4.2. In practice we can compute these sets for groups of order 30. We also show how for composite groups $G = G_1 \times G_2$, it is possible to generate the irreducibles for G from those of G_1 and G_2. In Section 4.5 we outline possible algorithms based on these extended

formulations. Finally Section 4.6 reports on some computational results. We first test the proposed extended formulations in a pure "relax and reformulate" algorithm that relies on linear relaxation. The second test shows how the group reformulations can be used within the Integral Basis Method. The results of this chapter are taken from [37].

4.2 Extended Formulations for Group Relaxations

4.2.1 Irreducible solutions

We consider a set of the form

$$S(d) = \{x \in \mathbb{Z}^n_+ : Bx \equiv d \pmod{\Delta}\},$$

where $\Delta \in \mathbb{Z}^r$, $B \in \mathbb{Z}^{r\times n}$, and $d \in \mathbb{Z}^r_+$. Associated with $S(d)$ is the abelian group

$$\begin{aligned} G &= \{y \in \mathbb{Z}^r_+ : y_i \in \{0, \ldots, \Delta_i - 1\}, i = 1, \ldots, r\} \\ &= (\mathbb{Z}/\Delta_1\mathbb{Z}) \times \cdots \times (\mathbb{Z}/\Delta_r\mathbb{Z}), \end{aligned}$$

consisting of all residue classes of $\mathbb{Z}^r$ modulo the vector Δ. For a set $S(d)$, we now examine several possible extended formulations. To do this we need to introduce the notion of *irreducible* solutions.

Definition 4.1 *A vector $x \in \mathbb{Z}^n_+$ is an* irreducible solution *of $S(d)$ if $x \in S(d)$, and there is no other distinct non zero $\tilde{x} \in S(d)$ with $\tilde{x} \leq x$. Every irreducible vector x in $S(0)$ is called* homogeneous. *An irreducible vector x in $S(d)$ is called* inhomogeneous *whenever $d \neq 0$.*

An important property is that, for $d \not\equiv 0 \pmod{\Delta}$, every integer point in $S(d)$ can be represented as the sum of exactly one inhomogeneous irreducible solution of $S(d)$ and a nonnegative integer combination of the homogeneous irreducible solutions of $S(0)$.

4.2.2 The disaggregated reformulation

In order to come up with a first reformulation, we investigate a group relaxation

$$Y(f) = \{x \in \mathbb{Z}^n_+ : Bx \equiv f \pmod{\Delta}\}. \tag{4.4}$$

We determine a matrix C whose column vectors correspond to all inhomogeneous irreducible solutions of $Y(f)$. Accordingly we introduce a matrix D whose column vectors are all the homogeneous irreducible solutions of $Y(0)$. We associate integer λ and μ variables with the columns of C and D, respectively, to define the first formulation.

Proposition 4.1

$$Y(f) = \{x \in \mathbb{R}^n_+ : x = C\lambda + D\mu,\ 1\lambda = 1,\ \lambda \in \mathbb{Z}^s_+, \mu \in \mathbb{Z}^t_+\}.$$

Therefore the right-hand side of the equation is a valid extended formulation for $Y(f)$ referred to as the disaggregated formulation.

Example: Consider the set $X = \{x_1, x_2, x_3 \in \mathbb{Z}_+ : 3x_1 + 7x_2 + 9x_3 = 22\}$. By taking the equation (mod 4), we obtain the valid group relaxation

$$Y(2) = \{x_1, x_2, x_3 \in \mathbb{Z}_+ : 3x_1 + 3x_2 + x_3 \equiv 2 \pmod 4\}. \tag{4.5}$$

The inhomogeneous irreducible solutions of $Y(2)$ are represented in the matrix

$$C = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

The homogeneous irreducible solutions of $Y(0)$ are represented in the matrix

$$D = \begin{pmatrix} 1 & 0 & 4 & 3 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 3 & 4 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix}.$$

We refer to Section 4.4 for more details on how to compute these matrices of irreducibles. A valid reformulation for $Y(2)$ is thus to associate a new variable to each irreducible solution and hence write

$$\begin{aligned} Y(2) = \{x \in \mathbb{R}^3_+ :\ & \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_4 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 4 & 3 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 3 & 4 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_8 \end{pmatrix}, \\ & \lambda_1 + \cdots + \lambda_4 = 1 \\ & \lambda_1, \ldots, \lambda_4 \in \{0,1\},\ \mu_1, \ldots, \mu_8 \in \mathbb{Z}_+ \qquad \}. \end{aligned}$$

□

4.2.3 The aggregated reformulation

Inspecting the irreducible solutions in detail suggests that we can classify some of the irreducible solutions according to the value given by the sum of the first and second component. For instance, all the vectors $\begin{pmatrix} 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ have the property that the sum of the two first components is 4. This property can be highlighted for some other sets of vectors of C and D and is due to the fact that x_1 and x_2 have the same coefficient in the group equation of (4.5).

This suggests the idea of aggregating these two variables into one variable w so as to reduce the size of the reformulation. We now let $N = \{1, \ldots, n\}$ and $N_\alpha = \{j \in N : B_{:j} \equiv \alpha \pmod{\Delta}\}$, where $\alpha \in G$ and $B_{:j}$ denotes the j^{th} column of B. We aggregate the variables with the same coefficients into a new variable $w_\alpha = \sum_{j \in N(\alpha)} x_j$ and consider the set

$$W(f) = \{w \in \mathbb{Z}_+^{|\tilde{G}|} : \sum_{\alpha \in \tilde{G}} \alpha w_\alpha \equiv f \pmod{\Delta}\},$$

where $\tilde{G} = \{\alpha \in G : \text{there exists } j \text{ with } B_{:j} \equiv \alpha \pmod{\Delta}\}$. We denote by $\tilde{C} \in \mathbb{Z}^{n \times \tilde{s}}$ and $\tilde{D} \in \mathbb{Z}^{n \times \tilde{t}}$ the matrices whose columns are the inhomogeneous and homogeneous irreducibles of $W(f)$ and $W(0)$ respectively. Now we are able to write a second formulation for $Y(f)$.

Proposition 4.2

$$Y(f) = \{x \in \mathbb{Z}_+^n : w = \tilde{C}\lambda + \tilde{D}\mu,\ 1\lambda = 1,\ \lambda \in \mathbb{Z}_+^{\tilde{s}},\ \mu \in \mathbb{Z}_+^{\tilde{t}},\ w_\alpha = \textstyle\sum_{j \in N(\alpha)} x_j,\ w \in \mathbb{Z}_+^{|\tilde{G}|}\}.$$

Therefore $\{x \in \mathbb{Z}_+^n : w = \tilde{C}\lambda + \tilde{D}\mu,\ 1\lambda = 1,\ \lambda \in \mathbb{Z}_+^{\tilde{s}},\ \mu \in \mathbb{Z}_+^{\tilde{t}}, w_\alpha = \sum_{j \in N(\alpha)} x_j,\ w \in \mathbb{Z}_+^{|\tilde{G}|}\}$ *is a valid extended formulation for* $Y(f)$ *referred to as the* aggregated formulation.

Example: Consider again the group relaxation (4.5),

$$Y(2) = \{x \in \mathbb{Z}_+^3 : 3x_1 + 3x_2 + x_3 \equiv 2 \pmod{4}\}.$$

We aggregate the first two variables in $w = x_1 + x_2$ and now consider an aggregated version of $Y(2)$,

$$W(2) = \{w, x_3 \in \mathbb{Z}_+ : 3w + x_3 \equiv 2 \pmod{4}\}.$$

The correponding matrices of irreducibles of $W(2)$ and $W(0)$ are

$$\tilde{C} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \quad \text{and} \quad \tilde{D} = \begin{pmatrix} 1 & 4 & 0 \\ 1 & 0 & 4 \end{pmatrix}.$$

The aggregated reformulation of $Y(2)$ is

$$\begin{aligned} Y(2) = \{x \in \mathbb{Z}_+^3 : {} & x_1 + x_2 = w_1 \\ & \begin{pmatrix} w_1 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} + \begin{pmatrix} 1 & 4 & 0 \\ 1 & 0 & 4 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{pmatrix} \\ & \lambda_1 + \lambda_2 = 1,\ \lambda \in \mathbb{Z}_+^2,\ \mu \in \mathbb{Z}_+^3,\ w \in \mathbb{Z}_+\}. \end{aligned}$$

Compared to the disaggregated formulation presented earlier, the aggregated formulation is much more compact, but we now need to keep the integrality constraints on x_1 and x_2. □

4.2.4 Advanced aggregation

One way of further reducing the size of the reformulation of $Y(f)$ consists in generalizing our aggregation technique to variables with different coefficients. We study here the case of a recursive aggregation of variables whose residue classes are integer multiples of each other. Consider any ordering of the variables. Without loss of generality, we assume that $\{1, \ldots, n\}$ reflects the ordering. We partition $L = \{1, \ldots, n\}$ iteratively in the following way:

(1) **Initialize** $L := \{1, \ldots, n\},\ I := \emptyset$.
(2) **While** $L \neq \emptyset$ **do**
(3) **Let** i be the first list element of L.
(4) **Set** $I := I \cup \{i\}$.
(5) Define a number $u_i \in \mathbb{Z}_+ \setminus \{0\}$ and a subset $L'(i) \subseteq L$.
(6) **Set** $N(i) := \{j \in L'(i) : \exists \lambda \in \mathbb{Z}_+, \lambda \leq u_i$ with $B_{:j} \equiv \lambda B_{:i} \pmod{\Delta}\}$.
(7) **Update** $L := L \setminus N(i)$.
(8) **Return** I and $N(i)$ for $i \in I$.

For every $i \in I$, we define a new variable z_i that collects all integer variables x_j with $j \in N(i)$. To define the coefficients of every variable in the aggregation, we denote by h_j the smallest positive integer such that $B_{:j} \equiv h_j B_{:i} \pmod{\Delta}$. The "aggregated" variable is then

$$z_i = \sum_{j \in N(i)} h_j x_j.$$

Using the z variables we can define a group relaxation

$$Z(f) = \Big\{ z \in \mathbb{Z}_+^{|I|} : \sum_{i \in I} B_{:i} z_i \equiv f \pmod{\Delta} \Big\}.$$

We denote by $\hat{C} \in \mathbb{Z}^{n \times \hat{s}}$ and $\hat{D} \in \mathbb{Z}^{n \times \hat{t}}$ the matrices whose columns are the inhomogeneous and homogeneous irreducibles of $Z(f)$ and $Z(0)$ respectively. We are now able to present a third formulation for $Y(f)$.

Proposition 4.3

$$\begin{aligned} Y(f) = \{x \in \mathbb{Z}_+^n : z = \hat{C}\lambda + \hat{D}\mu,\ 1\lambda = 1,\ \lambda \in \mathbb{Z}_+^{\hat{s}},\ \mu \in \mathbb{Z}_+^{\hat{t}}, \\ z_i = \textstyle\sum_{j \in N(i)} h_j x_j, \quad \textit{for all } i \in I \\ z \in \mathbb{Z}_+^{|I|}\}. \end{aligned} \tag{4.6}$$

Therefore the right hand side of (4.6) is a valid extended formulation of $Y(f)$. *We refer to it as the* advanced aggregation reformulation.

The advanced aggregation formulation is quite general and includes, in particular, the special cases discussed earlier. If we define $L'(i) = \{i\}$ in step (5) for

all $i \in \{1, \ldots, n\}$, then the advanced aggregation formulation coincides with the disaggregated formulation. Similarly setting $L'(i) = L$ and $u_i = 1$ for all $i \in I$ in step (5) yields the aggregated formulation. A natural generalization of the latter is obtained by setting $u_i = 2$ or $u_i = 3$ in step (5), respectively. We will show in Section 4.3 that we can analyze the strength of the corresponding extended formulation and enrich it by linear inequalities so as to come closer to the convex hull property.

Example: We study the set

$$X = \big\{ x \in \mathbb{Z}^5 : \quad x_1 + 7x_2 + 9x_3 + 8x_4 + 6x_5 = 22 \\ 3x_1 + 2x_2 + 6x_3 + 4x_4 + 5x_5 = 13 \big\},$$

and a corresponding group relaxation

$$Y\begin{pmatrix}2\\1\end{pmatrix} = \Big\{ x \in \mathbb{Z}^5_+ : \\ \begin{pmatrix}1\\3\end{pmatrix} x_1 + \begin{pmatrix}2\\2\end{pmatrix} x_2 + \begin{pmatrix}4\\2\end{pmatrix} x_3 + \begin{pmatrix}3\\0\end{pmatrix} x_4 + \begin{pmatrix}1\\3\end{pmatrix} x_5 \equiv \begin{pmatrix}2\\1\end{pmatrix} \bmod \begin{pmatrix}5\\4\end{pmatrix} \Big\}.$$

The aggregation is obtained as follows. We first set $L = \{1, \ldots, 5\}$ and $I = \emptyset$. For the first iteration, we choose $L'(1) = \{2, \ldots, 5\}$ and $u_1 = 3$. We note that the residue class of x_2 is twice the residue class of x_1. We put them together in the same aggregation. Therefore, $N(1) = \{1, 2\}$. Now $i = 3$ and with the parameters $L'(3) = \{4, 5\}$ and $u_3 = 3$, we note that the residue class of x_4 is twice that of x_3. We also observe that x_2 could be included in the same aggregation but as it is already in $N(1)$, we do not consider it here. Hence $N(3) = \{3, 4\}$. Finally, the last step provides $N(5) = \{5\}$ and $I = \{1, 3, 5\}$. The advanced aggregation provides the three variables $z_1 = x_1 + 2x_2$, $z_3 = x_3 + 2x_4$ and $z_5 = x_5$. Now to write the advanced aggregation formulation, we need to compute the irreducible solutions of

$$Z\begin{pmatrix}2\\1\end{pmatrix} = \left\{ z \in \mathbb{Z}^3_+ : \begin{pmatrix}1\\3\end{pmatrix} z_1 + \begin{pmatrix}4\\2\end{pmatrix} z_3 + \begin{pmatrix}1\\3\end{pmatrix} z_5 \equiv \begin{pmatrix}2\\1\end{pmatrix} \bmod \begin{pmatrix}5\\4\end{pmatrix} \right\},$$

which we do not do in this small example since it would involve too many vectors. □

4.2.5 Path reformulation

Finally we present a fourth reformulation that is based on the "path" structure of the group equation [61]. Let (V, A) be a digraph with $|G|$ nodes corresponding to each element of the group G, and arcs $(\alpha, \alpha + B_{\cdot j} \pmod{\Delta})$ for all $\alpha \in G$ and $j \in N$. Figure 4.1 shows such a graph related to the $x_1 + 2x_2 \equiv 2 \pmod 4$ group problem. The arcs above the nodes correspond to x_1 while the dotted arcs below the nodes correspond to $2x_2$. In the figure, any solution corresponds

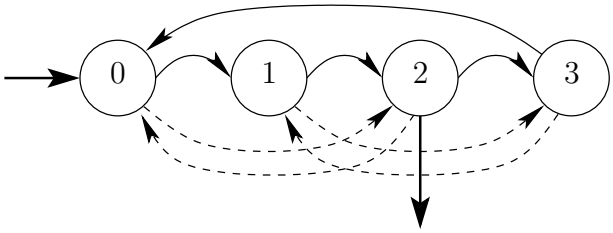

Figure 4.1: The path representation of $x_1 + 2x_2 \equiv 2 \pmod 4$, $x \in \mathbb{Z}^2_+$.

to a walk from 0 to 2, or equivalently to the pushing of one unit of flow from 0 to 2. Now if we go back to the general case, any walk from 0 to f corresponds to a point in $Y(f)$. Specifically let w^j_α be the number of times the arc $(\alpha, \alpha + B_{:j})$ occurs in the walk. Then

$$\sum_{\alpha \in G} \sum_{j \in N} B_{:j} w^j_\alpha \equiv f \pmod \Delta.$$

We can now formulate the group problem by flow constraints on each node of the graph, with a flow of 1 coming into node 0 and a flow of 1 going out of f, the new variables being the flow variables w.

Proposition 4.4

$$\begin{aligned} Y(f) = \{x : \quad & x_j = \sum_{\alpha \in G} w^j_\alpha \\ & \sum_{j \in N} w^j_0 - \sum_{j \in N} w^j_{\Delta - B_{:j}} = 1 \\ & \sum_{j \in N} w^j_\beta - \sum_{j \in N} w^j_{\beta - B_{:j}} = 0 \qquad \textit{for } \beta \neq 0, f \\ & \sum_{j \in N} w^j_f - \sum_{j \in N} w^j_{f - B_{:j}} = -1 \\ & w \in \mathbb{Z}^{n|G|}_+ \qquad\qquad \}, \end{aligned}$$

The right hand side of the equation above is a valid extended formulation of $Y(f)$ referred to as the path reformulation *of $Y(f)$.*

Example: Let us consider the following knapsack

$$x_1 + 5x_2 - 3x_3 = 3$$

and a corresponding group relaxation

$$Y(3) = \{x \in \mathbb{Z}^3_+ : x_1 + x_2 + 2x_3 \equiv 3 \pmod 4\}.$$

The path approach provides the following extended formulation

$$\begin{aligned} Y(3) = \{\, & x \in \mathbb{Z}^3_+ : \\ & x_1 = w^1_0 + w^1_1 + w^1_2 + w^1_3 \\ & x_2 = w^2_0 + w^2_1 + w^2_2 + w^2_3 \\ & x_3 = w^3_0 + w^3_1 + w^3_2 + w^3_3 \\ & w^1_0 + w^2_0 + w^3_0 - w^1_3 - w^2_3 - w^3_2 = 1 \\ & w^1_1 + w^2_1 + w^3_1 - w^1_0 - w^2_0 - w^3_3 = 0 \\ & w^1_2 + w^2_2 + w^3_2 - w^1_1 - w^2_1 - w^3_0 = 0 \\ & w^1_3 + w^2_3 + w^3_3 - w^1_2 - w^2_2 - w^3_1 = -1 \\ & w \in \mathbb{Z}^{12}_+. \qquad \} \end{aligned}$$

A solution $w^2_0 = w^3_1 = 1$ corresponds to $x_2 = x_3 = 1$. □

Example: The following example illustrates the sizes of the different formulations. We consider the single row group problem

$$3x_1 + 3x_2 + 3x_3 + 6x_4 + 5x_5 + 10x_6 + 7x_7 \equiv 1 \pmod{11}.$$

Table 4.1 shows the number of new variables for each of the different formulations. We remark that the corner polyhedron corresponding to this group problem has 18 nontrivial facets. □

	Irreducibles		
Formulation	Homogeneous	Inhomogeneous	Variables
Disaggregated	378	76	454
Aggregated	54	26	80
Advanced aggregated	13	8	21
Path			77

Table 4.1: The sizes of the different formulations

4.3 Analysis of the reformulations

In this section we analyze the four formulations presented in the previous section. We focus on showing under which conditions the convex hull property is satisfied for each formulation. We denote by P^1_Y the corresponding polyhedron when the integrality requirement of λ and μ is dropped in the disaggregated reformulation. Similarly we denote by P^2_Y, P^3_Y, P^4_Y the polyhedra corresponding to the aggregated, the advanced aggregation and the path reformulation respectively, when all the integrality constraints are dropped.

4.3.1 Tight reformulations

An important result is that the convex hull property holds for three among the four reformulations, namely for the disaggregated, the aggregated and the path reformulation. By convex hull property, we mean that the projection of an extended formulation onto the space of original variables is the convex hull of the original problem. To prove this result, two intermediate propositions are needed.

Proposition 4.5 *For $f \neq 0$, every extreme point of* $\text{conv}(Y(f))$ *is an irreducible inhomogeneous solution.*

Proof: Let y be a vertex of $\text{conv}(Y(f))$. We certainly have that y is an inhomogeneous solution of (4.4). Let us suppose now that y is reducible. Therefore there exists $y_1 \leq y, y_1 \neq y$, inhomogeneous solution of (4.4). We also have that $z = y - y_1$ is nonnegative and is an homogeneous solution of the group problem. Hence $y_2 = y + z$ is also a solution of (4.4) different from y. Furthermore

$$y = \frac{1}{2}y_1 + \frac{1}{2}y_2,$$

contradicting the fact that y is a vertex. □

It can also be noticed that every unit vector multiplied by the product of all Δ_i, for $i = 1, \ldots, r$ is definitely an homogeneous solution of the problem. Therefore there always exists a vector in the same direction which is irreducible.

Proposition 4.6 *For all* $\alpha \in \tilde{G}$, $\prod_{i=1}^{r} \Delta_i e_\alpha \in W(0)$. *Therefore* $\text{conv}(W(0)) = \mathbb{R}_+^{|\tilde{G}|}$. *Similarly,* $\text{conv}(Y(0)) = \mathbb{R}_+^{|N|}$.

Based on Propositions 4.5 and 4.6, it can be shown that the disaggregated, the aggregated and the path formulation not only model the group problem, but define "ideal formulations" in the sense that they produce the Gomory corner polyhedron, i.e., the convex hull of all the integer points of a group problem.

Theorem 4.1 $P_Y^1 = P_Y^2 = P_Y^4 = \text{conv}(Y(f))$.

Proof: For P_Y^1, the proof follows immediately from Proposition 4.5 and 4.6 as the extreme points are columns of C, and the extreme rays $k_j e_j$ are columns of D.

For P_Y^2, let us fix $x \in P_Y^2$ and show that we also have that $x \in P_Y^1$. The other inclusion is trivial. We have

$$w = \tilde{C}\lambda + \tilde{D}\lambda,\ 1\lambda = 1 \text{ and } w_\alpha = \sum_{j \in N(\alpha)} x_j,\ x, w, \lambda, \mu \geq 0.$$

Let us fix some α such that $w_\alpha \neq 0$. Then we can write

$$w_\alpha = x_\alpha^1 + \cdots + x_\alpha^{|N(a)|} = \sum_{i=1}^{\tilde{s}} \tilde{C}_{\alpha i}\lambda_i + \sum_{j=1}^{\tilde{t}} \tilde{D}_{\alpha j}\mu_j,$$

where $x_\alpha^1, \ldots, x_\alpha^{|N(\alpha)|}$ represent all the variables included in the set $N(\alpha)$. We also have

$$\begin{pmatrix} x_\alpha^1 \\ \vdots \\ x_\alpha^{|N(\alpha)|} \end{pmatrix} = \sum_{i=1}^{\tilde{s}} \begin{pmatrix} \tilde{C}_{\alpha i} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \lambda_i \frac{x_\alpha^1}{w_\alpha} + \cdots + \sum_{i=1}^{\tilde{s}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \tilde{C}_{\alpha i} \end{pmatrix} \lambda_i \frac{x_\alpha^{|N(\alpha)|}}{w_\alpha} + \sum_{j=1}^{\tilde{t}} \begin{pmatrix} \tilde{D}_{\alpha j} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \mu_j \frac{x_\alpha^1}{w_\alpha} + \cdots + \sum_{j=1}^{\tilde{t}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \tilde{D}_{\alpha j} \end{pmatrix} \mu_j \frac{x_\alpha^{|N(\alpha)|}}{w_\alpha}.$$

This can be rewritten using the unit vectors e_k as

$$\begin{pmatrix} x_\alpha^1 \\ \vdots \\ x_\alpha^{|N(\alpha)|} \end{pmatrix} = \sum_{k=1}^{|N(\alpha)|} \sum_{i=1}^{\tilde{s}} \tilde{C}_{\alpha i} \lambda_i \frac{x_\alpha^k}{w_\alpha} e_k + \sum_{k=1}^{|N(\alpha)|} \sum_{j=1}^{\tilde{t}} \tilde{D}_{\alpha j} \mu_j \frac{x_\alpha^k}{w_\alpha} e_k. \tag{4.7}$$

The expression (4.7) is valid when one fixes some α. We can write a similar expression for the complete vector x. In the summation, we need to consider all possible combinations of choosing an index $k(\alpha)$ for each $\alpha = 1, \ldots, |G|$. Remark that, to simplify the notation, we consider that $\frac{x_\alpha^k}{w_\alpha} = 0$ whenever $w_\alpha = 0$. We then have

$$\begin{aligned} x = &\sum_{i=1}^{\tilde{s}} \sum_{k_1=1}^{|N(1)|} \cdots \sum_{k_{|G|}=1}^{|N(|G|)|} \frac{x_1^{k_1} \cdots x_{|G|}^{k_{|G|}}}{w_1 \cdots w_{|G|}} \lambda_i \tilde{C}_{\alpha i} \begin{pmatrix} e_{k_1} \\ \vdots \\ e_{k_{|G|}} \end{pmatrix} \\ &+ \sum_{j=1}^{\tilde{t}} \sum_{k_1=1}^{|N(1)|} \cdots \sum_{k_{|G|}=1}^{|N(|G|)|} \frac{x_1^{k_1} \cdots x_{|G|}^{k_{|G|}}}{w_1 \cdots w_{|G|}} \mu_j \tilde{D}_{\alpha j} \begin{pmatrix} e_{k_1} \\ \vdots \\ e_{k_{|G|}} \end{pmatrix} \end{aligned} \tag{4.8}$$

The result follows from (4.8) since all the terms $\tilde{C}_{\alpha i} \begin{pmatrix} e_{k_1} \\ \vdots \\ e_{k_{|G|}} \end{pmatrix}$ lie in C and similarly for D. Furthermore we can check that the sum of the coefficients of those terms is 1, fulfilling the constraint $1\lambda = 1$.

For P_Y^4, the result follows from the fact that the matrix of flow constraints on the arcs w is totally unimodular. Hence every extreme point has integer values for w. Therefore x is integer as well. □

4.3.2 A bounded version of the path reformulation

The structure of the path reformulation is interesting because it leads to a totally unimodular matrix. This fact can be further used. We can also produce the convex hull for a bounded corner polyhedron as we outline below. Consider a bounded group relaxation

$$Y_B(f) = \{x \in \mathbb{Z}^n_+ : Bx \equiv f \pmod{\Delta},\ x \leq u\}, \tag{4.9}$$

where the vector $u \in \mathbb{Z}^n_+$ contains the bounds on the variables. We construct a digraph (V, A) with $n+1$ levels of nodes, one level per variable and a source level. Specifically the digraph has $(n+1)|G|$ nodes denoted by $V_{0\alpha}$ for the source level and by $V_{i\alpha}$ for $i = 1, \ldots, n$, $\alpha \in G$, corresponding to a group element at the i^{th} level. For each variable i, and each group element α, the arcs

$$(V_{(i-1),\alpha}, V_{i,\alpha}), (V_{(i-1),\alpha}, V_{i,((\alpha+B_{:i}) \mod \Delta)}), \cdots, (V_{(i-1),\alpha}, V_{i,((\alpha+u_iB_{:i}) \mod \Delta)})$$

belong to the graph. A solution to (4.9) is now any walk from the source node $V_{0,0}$ to the "target node" $V_{n,f}$. If we denote by $w(V_{(i-1)\alpha}, V_{i\beta})$ the flow going through the arc $(V_{(i-1),\alpha}, V_{i,\beta})$, for any $i \in N, \alpha, \beta \in G$, we have

$$\sum_{\alpha \in G} \sum_{i=1}^{n} \sum_{k=0}^{u_i} kB_{:i} w\left(V_{(i-1),\alpha}, V_{i,((\alpha+kB_{:i}) \mod \Delta)}\right) \equiv f \pmod{\Delta}.$$

Hence in any solution, the value of the variables is given by

$$x_i = \sum_{k=0}^{u_i} \sum_{\alpha \in G} kw\left(V_{(i-1),\alpha}, V_{i,((\alpha+kB_{:i}) \mod \Delta)}\right),$$

for all $i = 1, \ldots, n$. Figure 4.2 shows such a graph for the $x_1 + 2x_2 \equiv 2 \pmod 4$ group problem with $x_1 \in \{0, 1, 2\}$ and $x_2 \in \{0, 1\}$. A walk from $V_{0,0}$ to $V_{2,2}$ represents a solution. We note that several nodes can be removed from the formulation because all incident arc variables are fixed to zero in any solution (dashed in the figure).

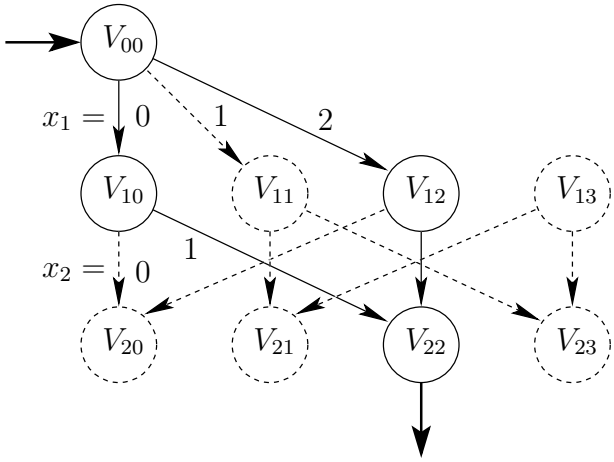

Figure 4.2: The graph related to the group problem $x_1+2x_2 \equiv 2 \pmod 4$, $x_1 \in \{0,1,2\}, x_2 \in \{0,1\}$.

Proposition 4.7

$$
\begin{aligned}
Y_B(f) = \{ & x \in \mathbb{Z}^n_+ : \\
& x_i = \sum_{k=0}^{u_i} \sum_{\alpha \in G} kw \left(V_{(i-1),\alpha}, V_{i,((\alpha+kB_{:i}) \bmod \Delta)} \right), \quad \textit{for } i = 1, \ldots, n \\
& \sum_{k=0}^{u_1} w(V_{0,0}, V_{1,(kB_{:1})}) = 1 \\
& \sum_{k=0}^{u_i} w(V_{(i-1),(\alpha - kB_{:(i-1)})}, V_{i,\alpha}) - \sum_{k=0}^{u_{i+1}} w(V_{i,\alpha}, V_{(i+1),(\alpha+kB_{:i})}) = 0 \\
& \qquad\qquad \textit{for all } i = 1, \ldots, n, \textit{ and } \alpha \in G, \ (i,\alpha) \neq (n,f) \\
& \sum_{k=0}^{u_n} w(V_{(n-1),(f-kB_{:i})}, V_{n,f}) = 1 \\
& w \in \mathbb{Z}^M_+ \qquad\qquad \},
\end{aligned}
$$

where $M = |G| \sum_{i=1}^{n} (u_i + 1)$. *Therefore, the expression in brackets is a valid extended formulation for* $Y_B(f)$*, referred to as the* bounded path reformulation.

The key argument to prove that the bounded path reformulation provides the convex hull of $Y_B(f)$ is again that the flow conservation constraints form a totally unimodular matrix. If we denote by P^5_Y the polyhedron obtained from the bounded path reformulation by relaxing the integrality constraints on the variables, we have the following result.

Proposition 4.8

$$P^5_Y = \text{conv}(Y_B(f)).$$

4.3.3 Strengthening the advanced aggregation reformulation

Here we suppose that we have used the advanced aggregation procedure outlined in Section 4.2.4. In general, the convex hull property does not hold any more for such a reformulation. However, we show in this section that we obtain a stronger formulation by adding inequalities and in one special case we can recover the convex hull property.

We consider the case in which each set $N(k)$ contains at most two elements. Thus we handle substitutions of the form

$$z_k = x_k + h_k x_{k'},$$

for $k \in I$, and $N(k) = \{k, k'\}$. Now the reformulation of Proposition 4.3 can be written row wise as

$$\tilde{P} = \Bigg\{ (x, \lambda, \mu) : x_k + h_k x_{k'} = \sum_{i=1}^{s} \tilde{C}_{ki}\lambda_i + \sum_{j=1}^{t} \tilde{D}_{kj}\mu_j \text{ for } k \in I, \\ x \in \mathbb{Z}^n_+, \quad \lambda \in \mathbb{Z}^s_+, \quad \mu \in \mathbb{Z}^t_+, \quad \sum_{i=1}^{s} \lambda_i = 1 \quad \Bigg\}.$$

This can be viewed as $\tilde{P} = \bigcap_{k\in I} \{ (x, \lambda, \mu) : x \in \mathbb{Z}^n_+, (x_k, x_{k'}, \lambda, \mu) \in \tilde{P}_k \}$ where

$$\tilde{P}_k = \Bigg\{ (x_k, x_{k'}, \lambda, \mu) : x_k + h_k x_{k'} = \sum_{i=1}^{s} \tilde{C}_{ki}\lambda_i + \sum_{j=1}^{t} \tilde{D}_{kj}\mu_j, \\ (x_k, x_{k'}) \in \mathbb{Z}^2_+, \quad \lambda \in \mathbb{Z}^s_+, \quad \mu \in \mathbb{Z}^t_+, \quad \sum_{i=1}^{s} \lambda_i = 1 \quad \Bigg\}.$$

Let P_k be the corresponding polyhedron obtained by dropping the integrality requirements on the variables.

Observation 4.1 *For fixed k,*

$$P_k \neq \text{conv}(\tilde{P}_k)$$

unless h_k divides $\tilde{C}_{ki}$ for all i and h_k divides $\tilde{D}_{kj}$ for all j.

If $h_k = 1$, the advanced aggregation can be viewed as a standard aggregated formulation. In this case, we have seen that the convex hull property is satisfied. But in general, the polyhedron P_k has fractional extreme points. However the convex hull of $\tilde{P}_k$ can be related to the convex hull of a group problem modulo h_k. We present a partial result. Let

$$\tilde{P}_{kl} = \{ (x_k, x'_k, \mu) : (x_k, x'_k, \lambda, \mu) \in \tilde{P}_k, \ \lambda_l = 1 \},$$

i.e. we fix some $\lambda_l = 1$ in $\tilde{P}_k$. The next proposition states that the convex hull of $\tilde{P}_{kl}$ is exactly obtained by adding all the corner polyhedron facets of the problem obtained by taking the equation defining $\tilde{P}_{kl}$ modulo h_k.

Proposition 4.9

$$\operatorname{conv}(\tilde{P}_{kl}) = \operatorname{conv}\Bigg\{ (x_k, x_{k'}, \mu) : -x_k + \sum_{j=1}^{t} \tilde{D}_{kj}\mu_j \equiv -\tilde{C}_{kl} \pmod{h_k}, \\ x_k \in \mathbb{Z}_+, \quad x'_k \in \mathbb{R}, \quad \mu \in \mathbb{Z}^t_+ \Bigg\} \tag{4.10a}$$

$$\cap \Bigg\{ (x_k, x_{k'}, \mu) : x_{k'} = \frac{1}{h_k}\Bigg(\tilde{C}_{kl} + \sum_{j=1}^{t} \tilde{D}_{kj}\mu_j - x_k \Bigg) \Bigg\}. \tag{4.10b}$$

Proof: Let us denote

$$Q_{kl} = \Bigg\{ (x_k, x_{k'}, \mu) : -x_k + \sum_{j=1}^{t} \tilde{D}_{kj}\mu_j \equiv -\tilde{C}_{kl} \pmod{h_k}, \\ x_k \in \mathbb{Z}_+, \quad x'_k \in \mathbb{R}, \quad \mu \in \mathbb{Z}^t_+ \Bigg\}.$$

Every integer point in $\tilde{P}_{kl}$ is also clearly in Q_{kl} since the modulo constraint (4.10a) is valid for the set $\tilde{P}_{kl}$. The difference is that $x_{k'}$ can be negative and not integer in Q_{kl}. We show now that every extreme point of (4.10a) with the definition (4.10b) is such that $x_{k'}$ is nonnegative and integer.

First, $x_{k'}$ is clearly integer for every extreme point by (4.10a). Then, we know that every extreme point of a modulo constraint of the type (4.10a) has the property that all variables take values less than the order of the group and thus $x_k \leq h_k - 1$. Hence

$$\tilde{C}_{kl} + \sum_{j=1}^{t} \tilde{D}_{kj}\mu_j - x_k \geq -h_k + 1,$$

as $\tilde{C}_{kl}, \tilde{D}_{kj}, \mu_j \geq 0$. But from (4.10a), we also know that the expression is a multiple of h_k. Therefore

$$\tilde{C}_{kl} + \sum_{j=1}^{t} \tilde{D}_{kj}\mu_j - x_k \geq 0,$$

which proves that $x_{k'} \geq 0$ for every extreme point. □

This proposition shows that we can strengthen a reformulation obtained by an advanced aggregation using the facets of a corner polyhedron. The size

of the group problem considered is related to the coefficient with which the aggregation is made in the case of the aggregation of two variables. It is well known that the number of facets of a group problem grows with the value of the modulus. Therefore, if the coefficient of the aggregation remains small, the number of facets to add to the reformulation stays small. The limit is, of course, if the coefficient $h_k = 1$, no facets have to be added.

The results of Proposition 4.9 can be partially extended to the set P_k. For this extension, we recall some results about corner polyhedra and integer programs. We define the corner polyhedron $P(C_{n,r})$ as the convex hull of the elements of the set

$$C_{n,r} = \{x \in \mathbb{Z}_+^{n-1} : \sum_{i=1}^{n-1} ix_i \equiv r \bmod n\}.$$

Let (π, γ), i.e. $\sum_{i=1}^{n-1} \pi_i x_i \geq \gamma$ be a non-trivial facet of $P(C_{n,r})$. One can prove that it satisfies the following properties

$$\pi_i \geq 0 \text{ for all } 1 \leq i \leq n-1 \tag{4.11}$$

$$\pi_i + \pi_j \geq \pi_k \text{ for all } 0 \leq i, j, k < n \text{ such that } i + j \equiv k \bmod n \tag{4.12}$$

$$\pi_i + \pi_j = \gamma \textit{ for all } 0 \leq i, j < n \text{ with } i + j \equiv r \bmod n \tag{4.13}$$

$$\pi_r = \gamma \tag{4.14}$$

We refer to (4.12) as the *subadditivity* property of the facets of the corner polyhedron, while (4.13) is the *complementarity* property. In the definition, we consider that $\pi_0 = 0$. These results initially come from [19] but can also be found in [4, 23, 16]. Applying a subadditive function to a valid equality of an integer program creates a new valid inequality for this problem [50].

Proposition 4.10 *Let $\mathcal{G} : \mathbb{R} \to \mathbb{R}$ be a subadditive function, i.e. such that*

$$\mathcal{G}(x) + \mathcal{G}(y) \geq \mathcal{G}(x + y) \qquad \textit{for all } x, y, x + y \in \mathbb{R},$$

and let $\sum_i a_i x_i = b$ be an equality satisfied for all $x \in X \subset \mathbb{Z}_+^n$. Then

$$\sum_{i=1}^{n} \mathcal{G}(a_i) x_i \geq \mathcal{G}(b)$$

is a valid inequality for X.

Using Proposition 4.10 and the properties of the facets of the corner polyhedron, we can now write valid inequalities for the set P_k.

Lemma 4.11 *Let (π_l, γ_l) be a facet-defining inequality for $\tilde{P}_{kl}$, the function $\Pi_l : \mathbb{R} \to \mathbb{R}$ defined by*

$$\begin{aligned} \Pi_l(x) = & \pi_l(\xi) && \textit{if } x \in \mathbb{Z} \\ & \mathcal{F}(\xi)(\pi_l(\lceil \xi \rceil) - \pi_l(\lfloor \xi \rfloor)) + \pi_l(\lfloor \xi \rfloor) && \textit{if } x \in \mathbb{R} \setminus \mathbb{Z}, \end{aligned}$$

is subadditive, when $\xi = x \ (mod\ h_k)$ *and* $\mathcal{F}(x) = x - \lfloor x \rfloor$.

Proof: This follows from the subadditivity property (4.12) of π_l and from the fact that the *fill in* function preserves the subadditivity (see [3]). □

Proposition 4.12 *Let* (π_l, γ_l) *be a facet-defining inequality for* $\tilde{P}_{kl}$*, and let* Π_l *be the function defined by Lemma 4.11. The inequality*

$$\Pi_l(1)x_k + \sum_{j=1}^{t} \Pi_l(-\tilde{D}_{kj})\mu_j \geq \sum_{i=1}^{s} \Pi_l(\tilde{C}_{ki})\lambda_i \tag{4.15}$$

is valid for $\tilde{P}_k$.

Proof: From Lemma 4.11, we know that Π_l is subadditive. Now we fix a point $x \in \tilde{P}_k$ and show that (4.15) is satisfied by x. Since $x \in \tilde{P}_k$, we have that $\lambda_{\bar{l}} = 1$ for some $1 \leq \bar{l} \leq s$ and also that

$$x_k + h_k x_{k'} - \sum_{j=1}^{t} \tilde{D}_{kj} = \tilde{C}_{k\bar{l}}$$

is valid for x. Since $\lambda_m = 0$ for all $m \neq \bar{l}$, we also have that

$$x_k + h_k x_{k'} - \sum_{j=1}^{t} \tilde{D}_{kj} = \sum_{i=1}^{s} \Pi_l(\tilde{C}_{ki})\lambda_i$$

is valid for x. Therefore

$$\Pi_l(1)x_k + \sum_{j=1}^{t} \Pi_l(-\tilde{D}_{kj})\mu_j \geq \Pi_l(\tilde{C}_{k\bar{l}})\lambda_{\bar{l}} \tag{4.16}$$

is valid for x which shows the result. □

Proposition 4.9 has been shown for the particular case of a fixed $\lambda_l = 1$. In general Proposition 4.12 allows one to add valid inequalities to any reformulation obtained by an advanced aggregation, but does not guarantee the convex hull property. When $h_k = 2$, the result of Proposition 4.9 can be generalized to obtain the convex hull property.

Proposition 4.13 *The polyhedron*

$$\begin{aligned} P_k^2 \quad = \quad & \Big\{ (x_1, x_2, \lambda, \mu) \in \mathbb{R}^{2+s+t} : \\ & x_1 + 2x_2 - \sum_{j=1}^{t} \hat{D}_{lj}\mu_j = \sum_{i=1}^{s} \hat{C}_{li}\lambda_i \qquad (4.17) \\ & x_2 - \sum_{j=1}^{t} \lceil \frac{\hat{D}_{lj}}{2} \rceil \mu_j \leq \sum_{i=1}^{s} \lfloor \frac{\hat{C}_{li}}{2} \rfloor \lambda_i \qquad (4.18) \\ & \sum_{i=1}^{s} \lambda_i = 1 \qquad \Big\} \end{aligned}$$

is integral.

Proof: We must show that every extreme point of P_k^2 is integer. First we consider the extreme points for which exactly one λ_i is non zero. Let us fix $1 \leq k \leq s$ such that $\lambda_k = 1$. The inequality (4.18) is the only facet that needs to be added to obtain the convex hull of the group problem

$$x_1 - \sum_{j=1}^{t} \hat{D}_{lj}\mu_j \equiv \hat{C}_{lk} \pmod 2.$$

Therefore we can apply Proposition 4.9 and we now that every extreme point corresponding to $\lambda_k = 1$ is integer.

We show now that no extreme point x^* can be such that $\lambda^*_{i_1}, \lambda^*_{i_2} \neq 0$, for $i_1 \neq i_2$. If this were the case, then exactly one other variable would be non zero making the three constraints tight. It clearly cannot be one of the other μ variables because it would not fulfill (4.17). Suppose $x^*_1 \neq 0$. From (4.17), we have $x^*_1 = \hat{C}_{li_1}\lambda^*_1 + \hat{C}_{li_2}\lambda^*_2$. This is the convex combination of the two feasible points $(x^1_1 = \hat{C}_{li_1}, \lambda^1_{i_1} = 1, \lambda^1_{i_2} = 0)$ and $(x^2_1 = \hat{C}_{li_2}, \lambda^2_1 = 0, \lambda^2_2 = 1)$. Therefore, x^* cannot be extreme. Suppose now that $x^*_2, \lambda^*_{i_1}, \lambda^*_{i_2} \neq 0$. As (4.17) and (4.18) are tight, it means that $\hat{C}_{li_1}$ and $\hat{C}_{li_2}$ are even because otherwise (4.17) cannot be tight. Then x^* is a convex combination of two points with $\lambda_{i_1} = 1$ and $\lambda_{i_2} = 1$ respectively like in the case where x_1 is non zero. Therefore no point with more than one λ_i non zero can be extreme.

Now, all the possible cases have been explored, and they all lead to either an integer extreme point or to an unbounded solution. Therefore, all the extreme points of the polyhedron are integer. □

Example: Consider the set

$$\begin{aligned} Q = \{ & (x_1, x_2, \lambda_1, \lambda_2, \mu_1, \mu_2, \mu_3) \in \mathbb{Z}^7_+ : \\ & x_1 + 2x_2 = \lambda_1 + 2\lambda_2 + 7\mu_1 + 2\mu_2 + 9\mu_3 \\ & \lambda_1 + \lambda_2 = 1 \qquad \} \end{aligned}$$

Its convex hull is given by Proposition 4.13. Therefore

$$\begin{aligned} Q^* = \{ & (x_1, x_2, \lambda_1, \lambda_2, \mu_1, \mu_2, \mu_3) \in \mathbb{R}^7_+ : \\ & x_1 + 2x_2 = \lambda_1 + 2\lambda_2 + 7\mu_1 + 2\mu_2 + 9\mu_3 \\ & \lambda_1 + \lambda_2 = 1 \\ & x_2 \leq \lambda_2 + 4\mu_1 + \mu_2\mu_3 + 5\mu_3 \end{aligned}$$

is integral and $Q^* = \text{conv}(Q)$. □

4.4 Computation of irreducible group solutions

In this section, we explain how to obtain the matrices C and D of irreducible solutions in order to be able to write the formulations presented in Section 4.2. The irreducible solutions of a group problem can be computed by using a Buchberger-type algorithm (see [3]) or by lexicographic enumeration. Both types of methods are very slow, and are therefore not suited to be used within an iterative integer programming algorithm.

4.4.1 Connection to knapsack master solutions

In [29, 35], it was proposed to pre-compute tables of irreducible solutions, in order to make use of them in an iterative algorithm. Tables of the irreducible solutions to a knapsack master equation

$$\begin{aligned} \sum_{i=1}^{n} ix_i - \sum_{i=1}^{n} iy_i = 0 \\ x \in \mathbb{Z}_+^n, \quad y \in \mathbb{Z}_+^n \end{aligned} \tag{4.19}$$

up to $n = 27$ were computed with a specialized recursive algorithm. These tables and the implementation of the algorithm are available at [36].

D	Knapsack	Group	D	Knapsack	Group
2	1	2	15	58 171	6 375
3	5	7	16	99 328	9 369
4	15	15	17	181 514	17 208
5	47	38	18	287 239	18 852
6	102	56	19	502 116	36 591
7	276	143	20	775 710	40 031
8	578	209	21	1 239 710	63 472
9	1 261	402	22	1 956 334	87 618
10	2 465	598	23	3 210 736	145 717
11	5 362	1 267	24	4 660 786	147 231
12	9 285	1 445	25	7 297 823	258 184
13	18 900	3 238	26	10 997 235	318 010
14	33 269	4 054	27	16 536 803	450 183

Table 4.2: The cardinality of knapsack and group master sets

We can make use of these tables to read off the irreducible solutions to the single-row master group problems

$$x_1 + 2x_2 + 3x_3 + \cdots + (D-1)x_{D-1} \equiv D_0 \pmod{D} \tag{4.20}$$

with varying right-hand side D_0 as follows. Equation (4.20) can be written in the form

$$x_1 + 2x_2 + 3x_3 + \cdots + (D-1)x_{D-1} - D_0 y - Dz = 0, \tag{4.21}$$

where the y variable can be left out in the homogeneous case $D_0 = 0$. The irreducible solutions to (4.21) are contained in the pre-computed table. The solutions with $y = 1$ correspond to the inhomogeneous solutions to (4.20), and the solutions with $y = 0$ correspond to the homogeneous solutions ($D_0 = 0$).

Because there are only two variables with negative coefficients in (4.21) and variable y is bounded above by 1, the knapsack master database stores more solutions than we need. Table 4.2 shows the cardinality of both the knapsack master database and the subset that contains the irreducible group solutions for all possible right-hand sides D_0. The irreducibke knapsack solutions do not include trivial solutions of the form (e^i, e^i) and the symmetric solutions (x, y) and (y, x) are only counted once.

4.4.2 The disaggregation procedure

Suppose, for example, that we construct a table $\mathcal{T}_D$ of the irreducibles of the master group problem

$$x_1 + 2x_2 + 3x_3 + \cdots + (D-1)x_{D-1} \equiv D_0 \pmod{D}, \tag{4.22}$$

for some $D \in \mathbb{Z}_+$ and all possible right-hand sides $0 \leq D_0 \leq D-1$. Once this table is precomputed, we are able to read off the irreducibles of every (mod D) group problem from the irreducibles given in the table. Suppose that we want to find the irreducibles for the group problem

$$\sum_{\alpha \in G} \sum_{k=1}^{\rho_\alpha} \alpha x_{\alpha k} \equiv D_0 \pmod{D}, \tag{4.23}$$

where ρ_α denotes the number of times the coefficient α appears in (4.23) for $\alpha \in G$.

It was shown in [29] how to read off solutions from master solution tables. We summarize the procedure given there. The irreducible solutions to (4.23) correspond to the solutions v to (4.22) where all components v_α whose coefficients α do not occur in (4.23) are zero.

Now let $v \in T_D$ be such a solution where $\rho_\alpha = 0$ implies $v_\alpha = 0$. For every $\alpha \in G$ with $\rho_\alpha \neq 0$, we consider all possible "number partitions"

$$v_\alpha = \sum_{k=1}^{\rho_\alpha} w_{\alpha k}, \quad w_{\alpha k} \in \mathbb{Z}_+. \tag{4.24}$$

Set $\mathcal{I} := \emptyset$.
For each vector $v \in \mathcal{T}_D$ do
  If for all $\alpha \in G$, $\rho_\alpha = 0$ implies $v_\alpha = 0$ **then**
    Create irreducibles w^j corresponding to the solutions of the number partition problem
    $$\sum_{k=1}^{\rho_\alpha} w_{\alpha k} = v_\alpha$$
    for all $\alpha \in G$ such that $\rho_\alpha > 0$.
    Set $\mathcal{I} := \mathcal{I} \cup \left(\bigcup_j \{w^j\} \right)$.
Return $\mathcal{I}$.

Table 4.3: Disaggregation Algorithm

There are $\binom{\rho_\alpha + v_\alpha - 1}{\rho_\alpha}$ number partitions. The combinations of all possible number partitions (4.24) for all $\alpha \in G$ define the irreducible solutions to (4.23).

We show the disaggregation algorithm in Table 4.3. In the following proposition we summarize its properties.

Proposition 4.14 (i) *The disaggregation algorithm provides the set of all irreducibles of (4.23).*

(ii) *The number of irreducibles generated from a vector v for which $\rho_\alpha = 0$ implies $v_\alpha = 0$ for all $\alpha \in G$ is*

$$\prod_{\alpha \in G : v_\alpha > 1} \binom{\rho_\alpha + v_\alpha - 1}{\rho_\alpha}. \tag{4.25}$$

The number (4.25) of irreducible solutions of a group problem grows exponentially with the size of ρ_α. This particularly affects the disaggregated formulation. Conversely the other formulations based on irreducibles group together the variables having the same coefficients so that ρ_α is small for all $\alpha \in G$.

4.4.3 An iterative procedure using subgroups

It is computationally intractable to build tables for single-row master group problems if the modulus D is too big. However, there is an alternative for composite groups with D non prime, in the case of a single-row group problem. It is possible to build up the set of irreducibles from the irreducibles of smaller groups. Suppose specifically that $D = pq$ with $p, q > 1$ integer. Starting from

$$W(a_0) = \left\{ w \in \mathbb{Z}_+^D : \sum_{j=1}^{D-1} a_j w_j \equiv a_0 \pmod{D} \right\},$$

we consider the relaxation

$$\tilde{W}(g_0) = \Big\{ w \in \mathbb{Z}_+^D : \sum_{j=1}^{D-1} g_j w_j \equiv g_0 \pmod{p} \Big\}$$

where $a_j = f_j p + g_j$ for all $0 \leq j \leq D-1$. Given the matrices of irreducibles $\tilde{C}$ and $\tilde{D}$ for $\tilde{W}(g_0)$, we have that

$$\tilde{W}(g_0) = \big\{ w : w = \tilde{C}\lambda + \tilde{D}\mu,\ 1\lambda = 1, \lambda_i \in \mathbb{Z}_+, \mu_j \in \mathbb{Z}_+ \big\}.$$

Substituting for w in $W(a_0)$, we obtain that

$$a^T \tilde{C}\lambda + a^T \tilde{D}\mu \equiv a_0 \pmod{D},$$

where $a^t = pf^t + g$, $g\tilde{C} = g_0(1 \cdots 1), g\tilde{D} = 0$. In other words

$$\begin{aligned} &pf^T\tilde{C}\lambda + g^T\tilde{C}\lambda + pf^T\tilde{D}\mu + g\tilde{D}\mu \equiv f_0 p + g_0 \pmod{pq} \\ &1\lambda = 1. \end{aligned}$$

Dividing by p, and using $g^T\tilde{C}\lambda + g\tilde{\mu} \equiv g_0$, we obtain

$$\begin{aligned} &f^T\tilde{C}\lambda + f^T\tilde{D}\mu \equiv f_0 \pmod{q} \\ &1\lambda = 1 \end{aligned}$$

We now consider the possible choices for λ separately. If λ is the s-th unit vector, $s \in \{1, \ldots, S\}$, let A^s be the matrix of inhomogeneous irreducibles for

$$\big\{ \mu : f^T\tilde{D}\mu \equiv f_0 - f^T\tilde{C}_{:s} \pmod{q} \big\}$$

and B the matrix of homogeneous irreducibles. We now have the representation

$$\begin{aligned} &\mu = A^s\alpha^s + B\beta \\ &\sum_j \alpha_j^s = 1 \\ &\lambda_s = 1 \end{aligned} \tag{4.26}$$

By collecting all the matrices A^s for all s, we finally obtain the representation

$$\mu = \sum_s A^s\alpha^s + B\beta \tag{4.27a}$$

$$\sum_{s=1}^{S} \alpha^s = 1 \tag{4.27b}$$

The columns of the representation (4.27) contain all the irreducible inhomogeneous and homogeneous solutions. We remark, however, that the same solution can appear multiple times, and it is also possible that reducible solutions

show up. As it is easy to remove the duplicates and the reducible solutions from (4.27), the construction above opens up the possibility of computing the set of irreducibles of bigger groups from the database that we have precomputed. In other words, it suffices to compute group master sets for cyclic groups of prime order.

We remark that the iterative procedure above can be applied in a more general situation. Above we applied it to compute irreducible solutions to a group problem in $\mathbb{Z}/(pq)\mathbb{Z}$ from the (homogeneous and inhomogeneous) irreducible solutions to the group problems in the subgroup $\mathbb{Z}/p\mathbb{Z}$ and in the factor group $\mathbb{Z}/q\mathbb{Z}$. The procedure can be applied for an arbitrary (abelian) group G to compute the irreducible solutions from the irreducible solutions to a given subgroup $G' \leq G$ and to the corresponding factor group G/G'. We will make use of this observation when we deal with multi-row group relaxations.

Example: Consider the problem of finding irreducibles for

$$\begin{aligned} & w_1 + 2w_2 \equiv 3 \pmod 4 \\ & w \in \mathbb{Z}^2_+. \end{aligned}$$

By first considering the equation (mod 2), it yields

$$\begin{aligned} & w_1 + 0w_2 \equiv 1 \pmod 2 \\ & w \in \mathbb{Z}^2_+, \end{aligned}$$

which means that

$$\begin{aligned} & \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \lambda_1 + \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \mu \\ & \lambda_1 = 1. \end{aligned} \tag{4.28}$$

Substituting gives

$$\begin{aligned} & \lambda_1 + 2\mu_1 + 2\mu_2 \equiv 3 \pmod 4 \\ & \lambda_1 = 1. \end{aligned}$$

By replacing λ_1 by its value and by dividing by 2, we now have

$$\mu_1 + \mu_2 \equiv 1 \pmod 2.$$

By using the representation by irreducibles, we can write

$$\begin{aligned} & \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \alpha + \begin{pmatrix} 2 & 1 & 0 \\ 0 & 1 & 2 \end{pmatrix} \beta \\ & \alpha_1 + \alpha_2 = 1. \end{aligned} \tag{4.29}$$

By substituting (4.29) in (4.28), we finally obtain

$$\begin{aligned} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \alpha + \begin{pmatrix} 4 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix} \beta \\ &= \begin{pmatrix} 3 & 1 \\ 0 & 1 \end{pmatrix} \alpha + \begin{pmatrix} 4 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix} \beta \end{aligned} \tag{4.30}$$

with $\alpha_1 + \alpha_2 = 1$ and $\alpha, \beta \geq 0$ and integer. In this example, we obtain a representation that consists of irreducible solutions only. □

4.4.4 Multi-row group relaxations

So far we have focused on single-row group relaxations, which lead to cyclic groups $\mathbb{Z}/D\mathbb{Z}$. For multi-row group relaxations with moduli $\Delta_1, \ldots, \Delta_k$, we are investigating a group problem in the direct product

$$G = (\mathbb{Z}/\Delta_1\mathbb{Z}) \times \cdots \times (\mathbb{Z}/\Delta_k\mathbb{Z}).$$

Its structure depends on the existence of common divisors of the moduli. If $\Delta_1, \ldots, \Delta_k$ are pairwise coprime, then

$$(\mathbb{Z}/\Delta_1\mathbb{Z}) \times \cdots \times (\mathbb{Z}/\Delta_k\mathbb{Z}) \cong \mathbb{Z}/(\Delta_1 \cdots \Delta_k)\mathbb{Z}. \tag{4.31}$$

Therefore, we could read off the irreducible solutions to the multi-row group relaxations from the single-row master group table of modulus $\Delta_1 \cdots \Delta_k$.

In the more general case, we can consider a chain of subgroups of G and apply the iterative construction above. For instance, we could take the chain

$$\begin{aligned}\mathbb{Z}/\Delta_1\mathbb{Z} &\leq (\mathbb{Z}/\Delta_1\mathbb{Z}) \times (\mathbb{Z}/\Delta_2\mathbb{Z}) \\ &\leq (\mathbb{Z}/\Delta_1\mathbb{Z}) \times (\mathbb{Z}/\Delta_2\mathbb{Z}) \times (\mathbb{Z}/\Delta_3\mathbb{Z}) \leq \cdots \leq G.\end{aligned}$$

Table 4.4 shows the number of irreducible solutions to two-row master group problems. One can see that in the case of coprime moduli, we obtain the same number of irreducible solutions as in the single-row master table of the product of the moduli.

4.5 Algorithms based on reformulations

We now discuss two ways, one dual and one primal, to incorporate the extended reformulations presented above algorithmically. We are trying to solve an integer program

$$\begin{array}{ll}\max & c^T x \\ \text{s. t.} & Ax = b \\ & x \in \mathbb{Z}^n_+.\end{array} \tag{4.32}$$

For the primal algorithm we assume that a solution $x_0 \in \mathbb{Z}^n_+$ feasible for (4.32) is given.

In Table 4.5 we present the dual scheme. On the other hand the primal approach closely follows the Integral Basis Method; it is shown in Table 4.6.

Second modulus	First modulus			
	2	3	4	5
2	10			
3	56*	252		
4	132	1445*	5228	
5	598*	6375*	40031*	169892
6	1048	12954	106022	
7	4054*	63472*		
8	6324	147231*		
9	18852*	311253		

Table 4.4: The number of irreducibles for two-row master group problems. An asterisk marks the numbers corresponding to coprime pairs of moduli.

1. **Initialization**
   Take any valid formulation of the integer program.

2. **Geometric Search for Fractionality**
   Compute a fractional point x_{frac} of the polyhedron, for instance the linear relaxation optimum.
   If no such point exists, return the linear relaxation optimum as an optimal solution of (4.32).

3. **Group Relaxation**
   Generate a group relaxation

   $$Y^G = \{x \in \mathbb{Z}^n_+ : Bx \equiv f \pmod{\Delta}\}$$

   from a subset of tight constraints defining x_{frac}.

4. **Reformulation**
   Compute an extended reformulation for Y^G.

5. **Make Compact**
   Use bounds on variables, as well as other problem constraints or problem knowledge to eliminate as many of the new variables as possible, adding also GUB or SOS constraints to the description of Y^G. The goal here is to prevent an excessive increase in the number of variables. If the number is still too large, the set Y^G must be relaxed further.

6. **Update Problem Formulation**

7. **Go to** Step 2.

Table 4.5: The dual algorithmic scheme

1. **Initialization**
   Compute an integer tableau for which x_0 is the basic feasible solution.
2. **Group Relaxation**
   How to choose the appropriate group relaxation is less clear, as there is no obvious point to be cut off.
3. **Reformulation**
4. **Make Compact**
5. **Update Simplex Tableau**
   Update the integer tableau introducing the new columns and rows. Select the right variables to enter the basis in order to recover a tableau.
6. **Augmentation**
   Check for augmentation, i.e., check if there exists a new column v of positive reduced cost such that $x_i + v$ is feasible. In that case, pivot in v in an integer fashion and obtain an integer tableau representing the new feasible solution $x_{i+1} = x_i + v$.
7. **Go to** Step 2.

Table 4.6: The primal algorithmic scheme

In our tests, we have combined the two in the sense that we take a primal all-integer tableau to launch the dual algorithm, and we use the group relaxation of the dual scheme to provide a group for the primal. The Integral Basis Method introduced by Haus, Köppe and Weismantel [29] provides the algorithmic frame for testing our algorithm. In particular the following four phases have been described in [29]: Initialization, Make Compact, Update Simplex Tableau, Augmentation. In Section 4.4, we addressed the question of computing the reformulation. In the following, we address some questions related to the phases Search for Fractionality and Group Relaxation.

Geometric Search for Fractionality. We start from an integer point x. Associated with this integer point is an algebraic tableau representation that encodes the geometry of the underlying linear programming relaxation. Investigating a small subset of the rows of this tableau means geometrically that we only consider the vicinity of x. However the "interesting" part of the linear programming polyhedron is hidden. These heuristic arguments motivate to inspect the linear programming optimum and use its tableau representation to select few rows from which a group relaxation can be built that cuts off the

fractional point.

Group Relaxation. We compute the linear programming optimum of our current tableau using floating point arithmetic (with CPLEX). We extract one or two fractional basic variables whose fractional part is closest to 1/2. For those variables, we reconstruct in exact rational arithmetic the corresponding rows. It turns out that on our entire test set, the least common multiple of all the denominators occuring in a fractional row is gigantic, namely up to hundreds of digits. Clearly one cannot work with a group of this size. Our strategy is to define a tractable group problem by selecting a modulus (or two moduli) between 2 and 20 (between 2 and 8) for which the irreducible solutions have been tabulated in our database. Within this range, we choose a modulus that

(i) is not a divisor of the right hand side, in order to cut off the optimum point with our group relaxation,

(ii) is a divisor of the coefficient of the basic variable, if possible. Otherwise, there always exists some optimal solution to the group problem that has a zero reduced cost and the value of the dual bound does not change.

(iii) produces the maximum number of 0-residue coefficients in order to reduce the size of the reformulation.

Example: Suppose that an interesting row provided by the fractional tableau at the optimum LP point is

$$x_0 + \frac{930}{1000}x_1 + \frac{1724}{1000}x_2 - \frac{937}{1000}x_3 - \frac{620}{1000}x_4 + \frac{30}{1000}x_5 = \frac{127}{1000},$$

where x_0 is the basic variable. It can also be written in integer form

$$1000x_0 + 930x_1 + 1724x_2 - 937x_3 - 620x_4 + 30x_5 = 127.$$

As we pointed out before, it is computationally intractable to work with a (mod 1000) group. We decide therefore to choose some modulus between 2 and 20 in order to be able to use our precomputed table of irreducible solutions. We preferably choose a divisor of 1000. If we choose 5 or 10 as a modulus, four variables will disappear from the group relaxation, which can be interesting in the point of view of having a reformulation that is not too large. Furthermore, as 127 is not divisible by 5, the group relaxation will cut off the fractional point. Choosing 5 as the order of the group, we obtain

$$Y(2) = \{x_2, x_3 \in \mathbb{Z}_+ : 4x_2 + 3x_3 \equiv 2 \pmod 5\}.$$

p0033, IP value 3089			lseu, IP value 1120		
Iteration	Columns	LP bound	Iteration	Columns	LP bound
0	33	2520.6	0	89	837.0
10	95	2610.2	10	146	854.8
20	155	2647.4	20	199	901.6
30	195	2687.8	30	223	1001.1
40	221	2824.1	40	281	1006.8
50	290	2830.9	50	329	1008.2
70	361	2832.7	60	414	1008.9
110	626	2833.9	70	454	1009.2
120	653	2840.6	80	490	1009.3

Table 4.7: Closing the IP gap for some MIPLIB instances

4.6 Computational experiments

4.6.1 Improving the dual bound

In our first set of computational experiments, we explored how the strength of the formulation is changed by the reformulation steps. For several 0-1 problems from the benchmark library MIPLIB, we set up an all-integer simplex tableau corresponding to the optimal integer solution. Starting from this formulation, we apply aggregated reformulations based on single-row group relaxations, in order to improve the linear programming dual bound. Table 4.7 shows the results on some instances.

Some of the MIPLIB instances, like the Padberg–Crowder–Johnson instances p0548 and p2756, involve rows with big-M coefficients. It is clear that a group reformulation with such rows is very weak or even meaningless. Therefore, we ran these instances through the IP preprocessor of CPLEX 8.1 before starting our procedure. Table 4.8 shows the results.

In other instances, such as the Padberg–Crowder–Johnson instance p0201, we found that it was impossible to get an improvement of the LP value with the reformulation technique unless we started from a formulation augmented with strong cutting planes. To this end, we used CPLEX 8.1 to generate cutting planes at the root node of its computation. For technical reasons, we disabled all cut classes that lead to fractional coefficients. Table 4.9 shows the results for the instance p0201, where GUB cover cuts and cover cuts were applied. It also shows the results for the instance p0548, where only clique cuts were applied. We remark that CPLEX 8.1 can solve the latter instance in the root node, so we had to disable most cut classes for the experiment.

In all instances that we tested, we had to observe that after a number of

p0548, IP value 8691			p2756, IP value 3124		
Iteration	Columns	LP bound	Iteration	Columns	LP bound
0	371	7665.6	0	1557	2808.8
10	464	7681.1	10	1659	2809.5
20	613	7699.4	20	1715	2814.6
30	736	7710.1	30	1782	2816.6
40	2070	7764.2	50	2225	2817.6
50	3307	7801.6	60	2311	2818.0
60	3574	7807.7	70	2379	2818.9
70	3893	7842.2	80	2431	2824.7
80	3979	7941.4	90	2486	2827.5
90	4034	7975.1	100	2564	2836.0
100	4129	7997.3	110	2633	2853.9
110	4199	8017.2	120	2703	2861.7
120	4251	8026.7	130	2764	2862.5
130	4355	8047.7	150	2939	2863.3

Table 4.8: Closing the IP gap for MIPLIB instances after IP preprocessing

p0201, IP value 7615			p0548, IP value 8691		
Iteration	Columns	LP bound	Iteration	Columns	LP bound
0	159	7185.0	0	362	8135.6
2	187	7235.5	5	376	8142.7
4	201	7239.6	10	394	8152.1
6	229	7243.5	15	437	8154.5
8	263	7250.7	20	477	8156.3
10	494	7260.1	25	528	8162.3
12	504	7276.0	30	567	8166.1
14	513	7281.6	35	676	8184.5
16	525	7284.6	40	1072	8191.1
18	570	7286.8	45	1584	8204.0
20	588	7288.3	50	1929	8212.2
22	655	7290.6	53	1934	8237.0

Table 4.9: Closing the IP gap for MIPLIB instances augmented with strong cuts

iterations, the LP bound will not change any more, or only improve by tiny amounts. The reason is that the determinant of the optimal simplex basis of the problem increases during the reformulation algorithm. This implies that the effect of the group reformulations with small moduli becomes smaller.

4.6.2 Search for augmentation

Some experiments were also carried out to check that the group reformulation provides augmenting vectors in the reformulation. The algorithm used is the same as presented above. However the focus has sometimes to be put differently when the search for augmentation is a priority. For example, when one looks for a local augmentation, it seems better to inspect local rows. In this respect, Step 2 "Geometric Search for Fractionality" is reduced to search for interesting rows in the current tableau. The choice of the Group Relaxation in Step 3, is done in the same way except that we do not need to chose moduli that will likely cut off the fractional LP point. Indeed one only focuses on finding augmenting vectors within the reformulation and even sometimes do not use the reformulation further as explained now. In Step 4, if one uses an aggregated or path reformulation, the augmenting vectors are still hidden in the rows modeling the aggregation or in the rows modeling the flow conservation constraints. On the other hand, a disaggregated reformulation shows explicitly all the basic solutions to the group problem and therefore all the candidates for an augmentation. That is why we chose to perform disaggregated reformulations or aggregated reformulations with a heuristic to implicitly visit the vectors.

We tested the search for augmentation on some instances of the MIPLIB. Each time we find an augmentation, we recompute a tableau with the original variables in order to keep a size for which disaggregated formulations are computable. The computation is stopped when no augmentation vector can be found on a row of the current tableau. The results are presented in Table 4.10 and 4.11.

Objective	Row	Modulus	Gap closed
1660	R123	10	35 %
1472	R123	10	66 %
1303			

Table 4.10: Augmentation for the MIPLIB instance `lseu`, IP value 1120

Objective	Row	Modulus	Gap closed
366 777	R1026	5	34 %
329 640	R1026	5	38 %
325 655	R1056	5	41 %
322 538			

Table 4.11: Augmentation for the MIPLIB instance `p0282`, IP value 258411

Chapter 5

Basis reduction for a structured problem

5.1 Introduction

In this chapter, we present the results of [44]. We introduced in Chapter 1 the problem of sharing objects of an attic in the fairest possible way. In this chapter we treat the same problem but in its financial form. Consider a banker who must establish a certain number of portfolios for his clients. Client i's portfolio must consist of d_i shares, and the banker holds a_j shares of type j whose estimated profit per share is p_j. The banker's problem is to divide up the $\sum a_j$ shares among the clients so that the expected profit per share of each client is as close as possible to the average value. Mathematically, he has the problem of finding a solution of

$$\begin{cases} \sum_j x_{ij} = d_i & 1 \leq i \leq m \\ \sum_i x_{ij} = a_j & 1 \leq j \leq n \\ \frac{1}{d_i}\left(\sum_j p_j x_{ij}\right) \simeq \bar{c} & 1 \leq i \leq m \\ x \in \mathbb{Z}_+^{mn}, \end{cases} \tag{5.1}$$

where $\bar{c}$ is the average expected profit $\frac{\sum_i \sum_j p_{ij} x_{ij}}{\sum x_{ij}}$ and $\simeq$ means that we want to be as close as possible to equality. One natural way to attempt to solve this problem is to introduce nonnegative slack variables s_i^+ and s_i^-, write $\frac{1}{d_i}\sum_j p_j x_{ij} + s_i^+ - s_i^- = \bar{c}$, take as objective function: min $\sum_i s_i^+ + \sum_i s_i^-$, and then solve the resulting mixed integer program with a commercial system using branch-and-bound or branch-and-cut such as Cplex or Xpress. As we already said in the introduction, this does not work. Even for small problems with $m = 6$ clients and $n = 15$ share types, an optimal solution cannot be found within hours.

The banker's problem is a variant with integer variables of the market share problem [60]. An alternative viewpoint is to see it as a closest vector problem. Specifically we have to find a nonnegative integer combination of the vectors $(\underline{e}_i, \underline{e}_j, p_j\underline{e}_i)^T$ that is as close as possible to the vector $(\underline{d}, \underline{a}, \bar{c}\underline{d})^T$.

The close relation to lattice problems suggests use of the reformulation proposed by Aardal et al. [2] that was introduced in Section 1.2.8. Specifically if we want to find points in the set

$$Z = \{x \in \mathbb{Z}_+^N : Ax = b\},$$

where $A \in \mathbb{Z}^{M\times N}$ and $b \in \mathbb{Z}^M$, we use the following two-step approach based on basis reduction. In step 1 we use basis reduction on the associated lattice $\begin{pmatrix} I & 0 \\ 0 & 1 \\ A & -b \end{pmatrix}$ of dimension $N+M+1$ to construct an alternative representation of the feasible set Z of the form

$$Z = \{x : x = q + P\lambda, \lambda \in \mathbb{Z}^{N-M}, x \geq 0\}, \tag{5.2}$$

where q, P are integer, $Aq = b$, P is an integral basis of the null space of A and due to the basis reduction algorithm q and the columns of P are "short". In step 2, we use a standard approach to solve the problem on the reformulated set Z with $\lambda \in \mathbb{Z}^{N-M}$ as the variables. The standard approach usually is a branch-and-bound or branch-and-cut system such as Cplex or Xpress. For small instances of the banker's problem, this approach works whereas the original MIP approach does not.

However difficulties arise when we try to solve larger instances by this approach. The dimension of the lattice to be reduced is $(M+N+1)$, and the basis reduction algorithm is $\mathcal{O}((M+N)^4)$ [11, 32]. For $M+N$ greater than 300, the basis reduction algorithm becomes too time consuming. To overcome this difficulty, we propose to take advantage of the special structure of Z. Specifically we consider sets of the form

$$Z = \{X \in \mathbb{Z}_+^{m\times n} : XA = C,\ BX = D\}, \tag{5.3}$$

where $A \in \mathbb{Z}^{n\times K}$, $B \in \mathbb{Z}^{L\times m}$, $C \in \mathbb{Z}^{m\times K}$ and $D \in \mathbb{Z}^{L\times n}$. Note that with $A = \begin{pmatrix} 1 & \cdots & 1 \\ p_1 & \cdots & p_n \end{pmatrix}^T$ and $B = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}$, we obtain the system (5.1) arising in the banker's problem presented above.

For the system (5.3), the direct approach of Aardal et al. involves reducing a basis of dimension $(mn + Km + Ln + 1)$ which is impractical. Instead, we work with the separate lattices $\mathcal{L}_A = \{\underline{x} \in \mathbb{Z}^n : \underline{x}A = \underline{0}\}$ and $\mathcal{L}_B = \{\underline{x} \in \mathbb{Z}^m : B\underline{x} = \underline{0}\}$, and we use the same approach to compute bases of $\mathcal{L}_A$ and $\mathcal{L}_B$, and use the resulting basis vectors to construct a reduced basis for the large lattice

$$\mathcal{L} = \{X \in \mathbb{Z}^{m\times n} : XA = \underline{\underline{0}},\ BX = \underline{\underline{0}}\}.$$

In this chapter, it is important not to be confused by the different sizes of the vectors and matrices. Therefore we choose to use a more careful notation than in the previous chapters. Specifically

- An underlined letter $\underline{a}$ represents a vector (of any dimension).
- a_i represents the i^{th} component of the vector $\underline{a}$.
- $\underline{a}^p$ represents the p^{th} vector of a collection of vectors.
- a_i^p represents the i^{th} component of the p^{th} vector of the collection.
- A capital letter A or a double underlined letter $\underline{\underline{a}}$ represents a matrix.
- $\langle \underline{x}, \underline{y} \rangle$ represents the standard inner product of $\underline{x}$ by $\underline{y}$.

5.2 Constructing an integral basis for $\mathcal{L}$

The problem we address in this section is to find an alternative representation of

$$\mathcal{L} = \{X \in \mathbb{Z}^{m\times n} : XA = \underline{\underline{0}},\ BX = \underline{\underline{0}}\}, \tag{5.4}$$

with given matrices $A \in \mathbb{Z}^{n\times K}$, with $K \leq n$ of rank K, and $B \in \mathbb{Z}^{L\times m}$, with $L \leq m$ of rank L.

Considering the second equation of (5.4), each column of X has to be a solution of the system $B\underline{x} = \underline{0}$. Let $\underline{\underline{\beta}} = \left(\underline{\beta}^1 \ \cdots \ \underline{\beta}^{m-L}\right)$ denote an integral basis of the lattice $\mathcal{L}_B = \{\underline{x} \in \mathbb{Z}^m : B\underline{x} = \underline{0}\}$. Thus each matrix $X \in \mathcal{L}$ can be written

$$X = \underline{\underline{\beta}} \begin{pmatrix} \underline{\lambda}^1 \\ \vdots \\ \underline{\lambda}^{m-L} \end{pmatrix}, \tag{5.5}$$

with $\underline{\lambda}^1, \cdots, \underline{\lambda}^{m-L} \in \mathbb{Z}^n$. Similarly we can use an integer basis of the lattice $\mathcal{L}_A = \{\underline{x} \in \mathbb{Z}^n : \underline{x}^T A = \underline{0}\}$ to describe the vectors of $\mathcal{L}$. Letting $\underline{\underline{\alpha}} = \begin{pmatrix} \underline{\alpha}^1 \\ \vdots \\ \underline{\alpha}^{n-K} \end{pmatrix}$ be such an integral basis, each solution X of (5.4) can be written

$$X = \left(\underline{\mu}^1 \ \cdots \ \underline{\mu}^{n-K}\right) \underline{\underline{\alpha}}, \tag{5.6}$$

where $\underline{\mu}^1, \cdots, \underline{\mu}^{n-K} \in \mathbb{Z}^m$.

Now we can state the main result of this section.

Theorem 5.1 *The matrices $X \in \mathcal{L}$ are precisely the matrices of the form*

$$X = \underline{\underline{\beta}} \Lambda \underline{\underline{\alpha}}, \tag{5.7}$$

with $\Lambda \in \mathbb{Z}^{(m-L)\times(n-K)}$.

To prove this theorem, we need several intermediate results.

Observation 5.1 *For fixed unimodular matrices $M \in \mathbb{Z}^{m\times m}$ and $N \in \mathbb{Z}^{n\times n}$, any $Y \in \mathbb{Z}^{m\times n}$ can be written as*

$$Y = M\Lambda N, \tag{5.8}$$

with $\Lambda \in \mathbb{Z}^{m\times n}$.

Proof: This is obvious by taking $\Lambda = M^{-1}YN^{-1}$ which is integral. □

The next two lemmas use the notion of Smith normal form that we now need to introduce.

Proposition 5.1 (Smith Normal Form) *Let A be an integer matrix of $\mathbb{Z}^{m\times n}$ with $m \geq n$ and rank$(A) = r$. There exist unimodular matrices $U \in \mathbb{Z}^{m\times m}$ and $V \in \mathbb{Z}^{n\times n}$ such that*

$$UAV = \begin{pmatrix} s_1 & & & \\ & \ddots & & \\ & & s_r & \\ & & & 0 \\ & \underline{\underline{0}}_{(m-n)\times n} & & \end{pmatrix}, \tag{5.9}$$

with $s_1|s_2|\ldots|s_r$. For $1 \leq k \leq r$, $\Pi_{i=1}^{k} s_i$ is the gcd of the determinants of all $k \times k$ submatrices of A. The matrix in the right hand side of (5.9) *is unique and called the Smith normal form of A.*

The Smith normal form allows us to characterize the bases of integral solutions of linear systems.

Lemma 5.2 *Consider the lattice $\{\underline{x} \in \mathbb{Z}^n : C\underline{x} = \underline{0}\}$, where $C \in \mathbb{Z}^{m\times n}$, $m \leq n$ and rank$(C) = m$. Let $Y = (\underline{y}^1, \ldots, \underline{y}^{n-m}) \in \mathbb{Z}^{n\times(n-m)}$ be a set of solutions of $C\underline{x} = \underline{0}$, then the following statements are equivalent*

(i) The Smith normal form of Y is $\begin{pmatrix} I \\ 0 \end{pmatrix}$.

(ii) Y is an integer basis of the lattice.

Proof: $(ii) \Rightarrow (i)$ As $\text{rank}(Y) = n - m$, there exist two unimodular matrices U and V such that

$$Y = U \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_{n-m} \\ & \underline{\underline{0}}_{m\times(n-m)} & \end{pmatrix} V,$$

with $U \in \mathbb{Z}^{n\times n}$ and $V \in \mathbb{Z}^{(n-m)\times(n-m)}$ and $s_{n-m} \geq 1$. As V is unimodular, its inverse exists, and we can write

$$YV^{-1} = U \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_{n-m} \\ & \underline{\underline{0}}_{m\times(n-m)} & \end{pmatrix}.$$

Then

$$U \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_{n-m} \\ & \underline{\underline{0}}_{m\times(n-m)} & \end{pmatrix} = \begin{pmatrix} s_1\underline{u}^1 & \cdots & s_{n-m}\underline{u}^{n-m} \end{pmatrix} \tag{5.10}$$

is also an integer basis of the integer solutions of $C\underline{x} = \underline{0}$ since it is obtained by multiplying a basis by the unimodular matrix V^{-1}. So $s_{n-m}\underline{u}^{n-m}$ is a solution of $C\underline{x} = \underline{0}$ and $\underline{u}^{n-m}$ is an integral solution as well. However it is only spanned by the basis (5.10) if $s_{n-m} = 1$. Hence, $s_{n-m} = 1$ and $s_i = 1$ for $i = 1, \cdots, n-m$ by the divisibility property of the Smith normal form.

$(i) \Rightarrow (ii)$: There exists a basis of all the integer solutions of $C\underline{x} = \underline{0}$. Let us call it W. By hypothesis, we can find two unimodular matrices $U \in \mathbb{Z}^{n\times n}$ and $V \in \mathbb{Z}^{(n-m)\times(n-m)}$ such that

$$UYV = \begin{pmatrix} I_{n-m} \\ \underline{\underline{0}}_{m\times(n-m)} \end{pmatrix}. \tag{5.11}$$

As each column of Y is a solution of $C\underline{x} = \underline{0}$, each of them is an integer combination of the columns of W. Thus, there exists a nonsingular integer matrix $\Lambda \in \mathbb{Z}^{(n-m)\times(n-m)}$ such that

$$Y = W\Lambda. \tag{5.12}$$

To prove that Y is an integer basis of the solutions of $C\underline{x} = \underline{0}$, we just have to prove that this matrix Λ is unimodular. We now show that ΛV is unimodular. Let S be its Smith normal form, so

$$P\Lambda VQ = S,$$

with P and Q unimodular. Rewriting (5.11), we now have

$$UW\Lambda V = UWP^{-1}SQ^{-1} = \begin{pmatrix} I \\ \underline{\underline{0}} \end{pmatrix}.$$

As U and Q^{-1} are unimodular, the Smith normal form of $WP^{-1}S$ is $\begin{pmatrix} I \\ \underline{\underline{0}} \end{pmatrix}$. In $WP^{-1}S$, all the elements of the last column are multiples of the last element of the diagonal of S, and thus $s_{n-m} = 1$. Therefore the Smith normal form of ΛV is $\begin{pmatrix} I \\ \underline{\underline{0}} \end{pmatrix}$, and ΛV is unimodular. Therefore Λ is unimodular and Y is a basis. □

Lemma 5.3 *A matrix $Y \in \mathbb{Z}^{n\times(n-m)}$ whose Smith normal form is $\begin{pmatrix} I \\ 0 \end{pmatrix}$ can be extended to a unimodular matrix.*

Proof: There exist unimodular matrices M and N such that

$$Y = \begin{pmatrix} M_1 & M_2 \end{pmatrix} \begin{pmatrix} I \\ 0 \end{pmatrix} N,$$

with $M_1 \in \mathbb{Z}^{n\times(n-m)}$, $M_2 \in \mathbb{Z}^{n\times m}$. Thus, we have that

$$Y = M_1 N.$$

Now, consider the completion $\begin{pmatrix} Y & M_2 \end{pmatrix}$, we have that

$$\begin{pmatrix} Y & M_2 \end{pmatrix} = \begin{pmatrix} M_1N & M_2 \end{pmatrix} = \begin{pmatrix} M_1 & M_2 \end{pmatrix} \begin{pmatrix} N & 0 \\ 0 & I \end{pmatrix}.$$

Thus,

$$|\det\begin{pmatrix} Y & M_2 \end{pmatrix}| = |\det M||\det N||\det I| = 1,$$

and M_2 completes Y as a unimodular matrix. □

Proof of Theorem 5.1: By using the two lemmas, we know that we can complete $\underline{\underline{\alpha}}$ and $\underline{\underline{\beta}}$ into unimodular matrices. We denote these completions by $\underline{\underline{\alpha}}^c$ and $\underline{\underline{\beta}}^c$. Therefore, by Observation 5.1, $X \in \mathbb{Z}^{m\times n}$ can be expressed as

$$X = \begin{pmatrix} \underline{\underline{\beta}} & \underline{\underline{\beta}}^c \end{pmatrix} \begin{pmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_3 & \Lambda_4 \end{pmatrix} \begin{pmatrix} \underline{\underline{\alpha}} \\ \underline{\underline{\alpha}}^c \end{pmatrix},$$

or expanding

$$\begin{aligned} X &= \begin{pmatrix} \underline{\underline{\beta}}\Lambda_1 + \underline{\underline{\beta}}^c\Lambda_3 & \underline{\underline{\beta}}\Lambda_2 + \underline{\underline{\beta}}^c\Lambda_4 \end{pmatrix} \begin{pmatrix} \underline{\underline{\alpha}} \\ \underline{\underline{\alpha}}^c \end{pmatrix} \\ &= (\underline{\underline{\beta}}\Lambda_1 + \underline{\underline{\beta}}^c\Lambda_3)\underline{\underline{\alpha}} + (\underline{\underline{\beta}}\Lambda_2 + \underline{\underline{\beta}}^c\Lambda_4)\underline{\underline{\alpha}}^c. \end{aligned}$$

Recall that each solution of (5.4) can be written in both forms (5.5) and (5.6). From the necessary condition (5.6), we see that X must be a combination of the rows of $\underline{\underline{\alpha}}$. Since $\underline{\underline{\alpha}}$ and $\underline{\underline{\alpha}}^c$ are linearly independent, $\underline{\underline{\beta}}\Lambda_2 + \underline{\underline{\beta}}^c\Lambda_4 = \underline{\underline{0}}$. But now, since the columns of $\underline{\underline{\beta}}$ and $\underline{\underline{\beta}}^c$ are linearly independent, we can conclude that, for each solution X of (5.4),

$$\Lambda_2 = \Lambda_4 = \underline{\underline{0}}.$$

Identically, by taking the condition (5.5), we can conclude that

$$\Lambda_3 = \Lambda_4 = \underline{\underline{0}}.$$

Hence,

$$X = \begin{pmatrix} \underline{\underline{\beta}} & \underline{\underline{\beta}}^c \end{pmatrix} \begin{pmatrix} \Lambda_1 & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{0}} \end{pmatrix} \begin{pmatrix} \underline{\underline{\alpha}} \\ \underline{\underline{\alpha}}^c \end{pmatrix}.$$

and the set of solutions of (5.4) can be characterized as

$$X = \underline{\underline{\beta}}\Lambda_1\underline{\underline{\alpha}},$$

for a $\Lambda_1 \in \mathbb{Z}^{(m-L)\times(n-K)}$. Conversely, it is clear that every X of this form is a member of (5.4). □

We have a general form of the vectors of (5.4). Now, we introduce some classical notions that allow us to simplify notation.

Definition 5.1 *Given a matrix Y, vec(Y) denotes the column composed of the first column of Y followed by the second, etc.*

Definition 5.2 *The Kronecker product of two matrices $C \in \mathbb{R}^{m\times n}$ and $D \in \mathbb{R}^{p\times q}$ is*

$$C \otimes D = \begin{pmatrix} c_{11}D & \cdots & c_{1n}D \\ \vdots & \ddots & \vdots \\ c_{m1}D & \cdots & c_{mn}D \end{pmatrix},$$

belonging to $\mathbb{R}^{mp\times nq}$.

Proposition 5.4 *For matrices C, Y, D such that CYD exists,*

$$vec(CYD) = (D^T \otimes C)vec(Y).$$

Proof: See, for example, [38]. □

Proposition 5.5 *$(\underline{\underline{\alpha}}^T \otimes \underline{\underline{\beta}})$ is a basis for the vectorized solutions of (5.4).*

Proof: Applying Proposition 5.4 to the matrix X given by Theorem 5.1, we obtain

$$\mathrm{vec}(X) = (\underline{\underline{\alpha}}^T \otimes \underline{\underline{\beta}})\mathrm{vec}(\Lambda).$$

□

So in other words, a basis for $\mathcal{L}$ is obtained by taking the Kronecker product of the bases $\underline{\underline{\alpha}}$ and $\underline{\underline{\beta}}$ of $\mathcal{L}_A$ and $\mathcal{L}_B$ respectively.

Example Consider the following linear system with $m = 3$ and $n = 4$.

$$\left.\begin{array}{l} x_{11} + x_{21} + x_{31} = 0 \\ x_{12} + x_{22} + x_{32} = 0 \\ x_{13} + x_{23} + x_{33} = 0 \\ x_{14} + x_{24} + x_{34} = 0 \end{array}\right\} \text{Equations } BX = \underline{\underline{0}}$$

$$\left.\begin{array}{l} x_{11} + x_{12} + x_{13} + x_{14} = 0 \\ 16x_{11} + 57x_{12} + 23x_{13} + 66x_{14} = 0 \\ x_{21} + x_{22} + x_{23} + x_{24} = 0 \\ 16x_{21} + 57x_{22} + 23x_{23} + 66x_{24} = 0 \\ x_{31} + x_{32} + x_{33} + x_{34} = 0 \\ 16x_{31} + 57x_{32} + 23x_{33} + 66x_{34} = 0 \end{array}\right\} \text{Equations } XA = \underline{\underline{0}}$$

$$x_{ij} \in \mathbb{Z}, \text{ for } i = 1, \cdots, 3 \text{ and } j = 1, \cdots, 4.$$

The set of solutions form a lattice $\mathcal{L} = \{X \in \mathbb{Z}^{3\times 4} : XA = 0,\ BX = 0\}$, with

$$A = \begin{pmatrix} 1 & 16 \\ 1 & 57 \\ 1 & 23 \\ 1 & 66 \end{pmatrix},\ B = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \text{ and } X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix}.$$

By Proposition 5.5 an integer basis can be found by computing the integer bases of the two lattices separately. First of all, we consider the lattice $\mathcal{L}_B$

$$\begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = 0, \quad y_1, y_2, y_3 \in \mathbb{Z}.$$

It is readily verified that $\underline{\underline{\beta}} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}^T$ is an integer basis of $\mathcal{L}_B$ (such a basis can be found by basis reduction). Similarly, for the lattice $\mathcal{L}_A$,

$$\begin{pmatrix} y_1 & y_2 & y_3 & y_4 \end{pmatrix} \begin{pmatrix} 1 & 16 \\ 1 & 57 \\ 1 & 23 \\ 1 & 66 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}, \quad y_1, y_2, y_3, y_4 \in \mathbb{Z},$$

an integer basis $\underline{\underline{\alpha}} = \begin{pmatrix} -1 & -4 & 2 & 3 \\ 10 & -3 & -11 & 4 \end{pmatrix}$ is obtained. Taking the Kronecker product of the two bases, we obtain that

$$(\underline{\underline{\alpha}}^T \otimes \underline{\underline{\beta}}) = \begin{pmatrix} -1 & 1 & 0 & -4 & 4 & 0 & 2 & -2 & 0 & 3 & -3 & 0 \\ 0 & -1 & 1 & 0 & -4 & 4 & 0 & 2 & -2 & 0 & 3 & -3 \\ 10 & -10 & 0 & -3 & 3 & 0 & -11 & 11 & 0 & 4 & -4 & 0 \\ 0 & 10 & -10 & 0 & -3 & 3 & 0 & -11 & 11 & 0 & 4 & -4 \end{pmatrix}^T$$

is a basis of the lattice $\mathcal{L}$.

5.3 A reduced basis of $\mathcal{L}$

In this section we show that, up to a reordering of the vectors, the basis constructed by computing the Kronecker product of reduced bases of the small lattices is itself reduced.
Observe that each column of the Kronecker product of the two matrices $\underline{\underline{\alpha}}$ and $\underline{\underline{\beta}}$ is the Kronecker product of a row (basis vector) of $\underline{\underline{\alpha}}$ with a column of $\underline{\underline{\beta}}$ (basis vector). We also need to take inner products of such vectors.

Proposition 5.6 *Let* $\underline{v}^1 := \underline{\gamma}^1 \otimes \underline{\delta}^1$ *and* $\underline{v}^2 := \underline{\gamma}^2 \otimes \underline{\delta}^2$ *with* $\underline{\gamma}^i \in \mathbb{R}^n$ *and* $\underline{\delta}^i \in \mathbb{R}^m$. *Then*

$$\langle \underline{v}^1, \underline{v}^2 \rangle = \langle \underline{\gamma}^1, \underline{\gamma}^2 \rangle \langle \underline{\delta}^1, \underline{\delta}^2 \rangle.$$

Proof:

$$\begin{aligned} \langle \underline{v}^1, \underline{v}^2 \rangle &= \langle (\gamma_1^1 \underline{\delta}^1, \cdots, \gamma_n^1 \underline{\delta}^1), (\gamma_1^2 \underline{\delta}^2, \cdots, \gamma_n^2 \underline{\delta}^2) \rangle \\ &= \gamma_1^1 \gamma_1^2 \langle \underline{\delta}^1, \underline{\delta}^2 \rangle + \cdots + \gamma_m^1 \gamma_m^2 \langle \underline{\delta}^1, \underline{\delta}^2 \rangle \\ &= \langle \underline{\gamma}^1, \underline{\gamma}^2 \rangle \langle \underline{\delta}^1, \underline{\delta}^2 \rangle. \end{aligned}$$

□

We also need to work with reduced bases. Let us recall Definition 1.13 of Section 1.2.8.

Definition 5.3 *Given* r *linearly independent vectors* $\underline{b}^j \in \mathbb{Z}^n$, $(\underline{b}^j)_{j=1,\cdots,r}$ *is a* reduced basis *if*

(i) $|\mu_{ij}| \leq \frac{1}{2}$ *for all* $i < j$

(ii) $\|\hat{\underline{b}}^{j+1} + \mu_{j,j+1} \hat{\underline{b}}^j\|^2 \geq \frac{3}{4} \|\hat{\underline{b}}^j\|^2$ *for* $1 \leq j \leq r-1$

where $\mu_{ij} = \frac{\langle \underline{b}^j, \hat{\underline{b}}^i \rangle}{\langle \hat{\underline{b}}^i, \hat{\underline{b}}^i \rangle}$, *and* $(\hat{\underline{b}}^j)_{j=1,\cdots,r}$ *is the Gram-Schmidt orthogonalization of* $(\underline{b}^j)_{j=1,\cdots,r}$.

For the rest of this section, we study the integral basis $V = (\underline{v}^{pq}) = (\underline{\alpha}^p \otimes \underline{\beta}^q)$ constructed in Section 5.2, where $(\underline{\alpha}^p)_{p=1}^P$ and $(\underline{\beta}^q)_{q=1}^Q$ are now *reduced* bases with $P = n-K$ and $Q = m-L$. To simplify the notation, we suppose that $\underline{\alpha}^p$ represents the p^{th} *row* of $\underline{\underline{\alpha}}$ while $\underline{\beta}^q$ represents the q^{th} *column* of $\underline{\underline{\beta}}$. Let $(\underline{\hat{\alpha}}^p)_p$, $(\underline{\hat{\beta}}^q)_q$ denote the Gram-Schmidt orthogonalization of the bases $(\underline{\alpha}^p)_p$ and $(\underline{\beta}^q)_q$ respectively, with Gram-Schmidt coefficients denoted μ^α and μ^β respectively.

We now consider possible orderings of the set V of basis vectors. Throughout this section $(p,q) < (p',q')$ means that $p \le p'$ and $q \le q'$ and $(p,q) \ne (p',q')$. On the other hand $(i,j) \prec (i',j')$ means that v^{ij} comes before $v^{i'j'}$ in the ordering.

Definition 5.4 *A total ordering of the basis V is called* a monotone ordering *if, for any distinct pair of vectors $\underline{v}^{pq}$ and $\underline{v}^{p'q'}$ with $(p,q) < (p',q')$, $\underline{v}^{pq}$ precedes $\underline{v}^{p'q'}$ in the ordering (or $(p,q) \prec (p',q')$).*

Proposition 5.7 *For any monotone ordering $\prec$ of the integral basis V, $(\underline{\hat{v}}^{pq}) = (\underline{\hat{\alpha}}^p \otimes \underline{\hat{\beta}}^q)$ is the Gram-Schmidt orthogonalization of V.*

Proof: Clearly, $\underline{\hat{v}}^{11} = \underline{v}^{11} = \underline{\alpha}^1 \otimes \underline{\beta}^1 = \underline{\hat{\alpha}}^1 \otimes \underline{\hat{\beta}}^1$. We then proceed by induction on $\prec$. We have

$$\begin{aligned}
\underline{\hat{\alpha}}^p \otimes \underline{\hat{\beta}}^q &= (\underline{\alpha}^p - \sum_{i=1}^{p-1} \mu_{ip}^\alpha \underline{\hat{\alpha}}^i) \otimes (\underline{\beta}^q - \sum_{j=1}^{q-1} \mu_{jq}^\beta \underline{\hat{\beta}}^j) \text{ by the Gram-Schmidt procedure} \\
&= \underline{v}^{pq} - \underline{\alpha}^p \otimes (\sum_{j=1}^{q-1} \mu_{jq}^\beta \underline{\hat{\beta}}^j) - (\sum_{i=1}^{p-1} \mu_{ip}^\alpha \underline{\hat{\alpha}}^i) \otimes \underline{\beta}^q + \sum_{i=1}^{p-1}\sum_{j=1}^{q-1} \mu_{ip}^\alpha \mu_{jq}^\beta (\underline{\hat{\alpha}}^i \otimes \underline{\hat{\beta}}^j) \\
&= \underline{v}^{pq} - (\underline{\hat{\alpha}}^p + \sum_{k=1}^{p-1} \mu_{kp}^\alpha \underline{\hat{\alpha}}^k) \otimes (\sum_{j=1}^{q-1} \mu_{jq}^\beta \underline{\hat{\beta}}^j) - (\sum_{i=1}^{p-1} \mu_{ip}^\alpha \underline{\hat{\alpha}}^i) \otimes (\underline{\hat{\beta}}^q + \sum_{l=1}^{q-1} \mu_{lq}^\beta \underline{\hat{\beta}}^l) \\
&\quad + \sum_{i=1}^{p-1}\sum_{j=1}^{q-1} \mu_{ip}^\alpha \mu_{jq}^\beta (\underline{\hat{\alpha}}^i \otimes \underline{\hat{\beta}}^j) \\
&= \underline{v}^{pq} - \sum_{(i,j)<(p,q)} \tilde{\mu}_{ij} (\underline{\hat{\alpha}}^i \otimes \underline{\hat{\beta}}^j).
\end{aligned}$$

Now by induction, $\underline{\hat{v}}^{ij} = \underline{\hat{\alpha}}^i \otimes \underline{\hat{\beta}}^j$ for all $(i,j) \prec (p,q)$. Because $(i,j) < (p,q)$ implies $(i,j) \prec (p,q)$, it follows that

$$\underline{\hat{v}}^{ij} = \underline{\hat{\alpha}}^i \otimes \underline{\hat{\beta}}^j \quad \text{for all } (i,j) < (p,q).$$

Hence,

$$\underline{\hat{\alpha}}^p \otimes \underline{\hat{\beta}}^q = \underline{v}^{pq} - \sum_{(i,j)<(p,q)} \tilde{\mu}_{ij} \underline{\hat{v}}^{ij}. \tag{5.13}$$

We now need to show that $\underline{\hat{\alpha}}^p \otimes \underline{\hat{\beta}}^q$ is orthogonal to all the $\underline{\hat{v}}^{ij}$ with $(i,j) \prec (p,q)$. Indeed, we have

$$\begin{aligned}\langle (\hat{\alpha}^p \otimes \hat{\beta}^q), \hat{v}^{ij} \rangle &= \langle \underline{\hat{\alpha}}^p \otimes \underline{\hat{\beta}}^q, \underline{\hat{\alpha}}^i \otimes \underline{\hat{\beta}}^j \rangle \quad \text{for all } (i,j) \prec (p,q) \\ &= \langle \hat{\alpha}^p, \hat{\alpha}^i \rangle \langle \hat{\beta}^q, \hat{\beta}^j \rangle \quad \text{by Proposition 5.6} \\ &= 0 \quad \text{for all } (i,j) \prec (p,q),\end{aligned}$$

because either $i \neq p$ or $j \neq q$. □

Let us remark that in the expression (5.13), only the indices $(i,j) < (p,q)$ have a corresponding μ non zero. This leads to the following observation.

Observation 5.2 *For a monotone ordering on V,*

$$\mu_{(i,j)(p,q)} = 0 \qquad \textit{for all } (i,j) \prec (p,q) \textit{with } (i,j) \not\prec (p,q).$$

Proposition 5.8 *For any monotone ordering of the basis V, condition (i) of Definition 5.3 of a reduced basis is satisfied.*

Proof: Let (p_1,q_1), (p_2,q_2) be a pair of indices, with $(p_1,q_1) \prec (p_2,q_2)$. If $(p_1,q_1) \not\prec (p_2,q_2)$, $\mu_{(p_1,q_1)(p_2,q_2)} = 0$ by Observation 5.2. Let us now consider $(p_1,q_1) < (p_2,q_2)$. We have

$$\begin{aligned}\mu_{(p_1,q_1)(p_2,q_2)} &= \frac{\langle \underline{v}^{p_2 q_2}, \underline{\hat{v}}^{p_1 q_1} \rangle}{\langle \underline{\hat{v}}^{p_1 q_1}, \underline{\hat{v}}^{p_1 q_1} \rangle} \\ &= \frac{\langle \underline{\alpha}^{p_2}, \underline{\hat{\alpha}}^{p_1} \rangle \langle \underline{\beta}^{q_2}, \underline{\hat{\beta}}^{q_1} \rangle}{\langle \underline{\hat{\alpha}}^{p_1}, \underline{\hat{\alpha}}^{p_1} \rangle \langle \underline{\hat{\beta}}^{q_1}, \underline{\hat{\beta}}^{q_1} \rangle} \quad \text{by Proposition 5.6} \\ &= \mu^{\alpha}_{p_1 p_2} \mu^{\beta}_{q_1 q_2}.\end{aligned}$$

Now as $(\underline{\alpha}^j)$ and $(\underline{\beta}^i)$ are reduced bases, $|\mu^{\alpha}_{p_1 p_2}|, |\mu^{\beta}_{q_1 q_2}| \leq \frac{1}{2}$, and thus

$$\begin{aligned}|\mu_{(p_1,q_1)(p_2,q_2)}| &= |\mu^{\alpha}_{p_1 p_2}| \, |\mu^{\beta}_{q_1 q_2}| \\ &\leq \frac{1}{4}.\end{aligned}$$

□

Now we need to refine the ordering of the basis V in order to satisfy condition (ii) of Definition 5.3.

Definition 5.5 *A monotone ordering of the basis V is called* regular *if whenever $\underline{v}^{p_1 q_1}$ directly precedes $\underline{v}^{p_2 q_2}$ in the ordering and $(p_1,q_1) \not\prec (p_2,q_2)$, $\|\underline{\hat{v}}^{p_1 q_1}\| \leq \|\underline{\hat{v}}^{p_2 q_2}\|$.*

Proposition 5.9 *If the basis V has a regular monotone ordering, it is a reduced basis.*

Proof: It suffices to show that the Condition (ii) of Definition 5.3 of a reduced basis is satisfied. Consider two pairs $(p_1, q_1), (p_2, q_2)$ where (p_1, q_1) directly precedes (p_2, q_2) in the ordering. There are two cases.
Case 1: $(p_1, q_1) < (p_2, q_2)$. Because the ordering is monotone, this implies that either $p_2 = p_1 + 1$ and $q_1 = q_2$ or $p_1 = p_2$ and $q_2 = q_1 + 1$. Suppose without loss of generality that $p_2 = p_1 + 1$ and $q_1 = q_2$.

$$
\begin{aligned}
&\|\hat{\underline{v}}^{p_1+1,q_1} + \mu_{(p_1,q_1)(p_1+1,q_1)}\hat{\underline{v}}^{p_1 q_1}\|^2 \\
&\quad = \|\hat{\underline{\alpha}}^{p_1+1} \otimes \hat{\underline{\beta}}^{q_1} + \tfrac{\langle \underline{\alpha}^{p_1+1}, \hat{\underline{\alpha}}^{p_1}\rangle \langle \underline{\beta}^{q_1}, \hat{\underline{\beta}}^{q_1}\rangle}{\langle \hat{\underline{\alpha}}^{p_1}, \hat{\underline{\alpha}}^{p_1}\rangle \langle \hat{\underline{\beta}}^{q_1}, \hat{\underline{\beta}}^{q_1}\rangle}(\hat{\underline{\alpha}}^{p_1} \otimes \hat{\underline{\beta}}^{q_1})\|^2 \\
&\quad = \|\hat{\underline{\alpha}}^{p_1+1} \otimes \hat{\underline{\beta}}^{q_1} + \mu^{\alpha}_{p_1,p_1+1}(\hat{\underline{\alpha}}^{p_1} \otimes \hat{\underline{\beta}}^{q_1})\|^2 \quad \text{since } \langle \underline{\beta}^{q_1}, \hat{\underline{\beta}}^{q_1}\rangle = \langle \hat{\underline{\beta}}^{q_1}, \hat{\underline{\beta}}^{q_1}\rangle \\
&\quad = \|(\hat{\underline{\alpha}}^{p_1+1} + \mu^{\alpha}_{p_1,p_1+1}\hat{\underline{\alpha}}^{p_1}) \otimes \hat{\underline{\beta}}^{q_1}\|^2 \\
&\quad = \|\hat{\underline{\alpha}}^{p_1+1} + \mu^{\alpha}_{p_1,p_1+1}\hat{\underline{\alpha}}^{p_1}\|^2 \|\hat{\underline{\beta}}^{q_1}\|^2 \\
&\quad \geq \tfrac{3}{4}\|\hat{\underline{\alpha}}^{p_1}\|^2 \|\hat{\underline{\beta}}^{q_1}\|^2 \qquad \text{since } (\underline{\alpha}^j) \text{ is reduced} \\
&\quad = \tfrac{3}{4}\|\hat{\underline{\alpha}}^{p_1} \otimes \hat{\underline{\beta}}^{q_1}\|^2 = \tfrac{3}{4}\|\hat{\underline{v}}^{p_1 q_1}\|^2.
\end{aligned}
$$

Case 2: $(p_1, q_1) \not< (p_2, q_2)$
By Observation 5.2, we know that $\mu_{(p_1,q_1)(p_2,q_2)} = 0$. So

$$
\begin{aligned}
\|\hat{\underline{v}}^{p_2 q_2} + \mu_{(p_1,q_1)(p_2,q_2)}\hat{\underline{v}}^{p_1 q_1}\|^2 &= \|\hat{\underline{v}}^{p_2 q_2}\|^2 \\
&\geq \|\hat{\underline{v}}^{p_1 q_1}\|^2 \quad \text{as the ordering is regular} \\
&> \frac{3}{4}\|\hat{\underline{v}}^{p_1 q_1}\|^2.
\end{aligned}
$$

□

We now present an algorithm to construct a regular monotone ordering of the basis V.

Definition 5.6 *Given a vector $\underline{v}^{pq} \in V$, the* direct successors *of $\underline{v}^{pq}$ are the vectors $\underline{v}^{p+1,q}$ and $\underline{v}^{p,q+1}$ (if they exist) and the* predecessors *are the vectors $\underline{v}^{p'q'}$ with $(p', q') < (p, q)$.*

Ordering Algorithm: RB is an ordered set of vectors from V.
$\mathcal{S}$ is a set of vectors from V.

Initialization: $RB := \{\underline{v}^{11}\}$
$\mathcal{S} := \{\underline{v}^{12}, \underline{v}^{21}\}$.

Loop: While $\mathcal{S} \neq \emptyset$ do
Choose $\underline{v} \in \mathcal{S}$ such that $\underline{v} = \arg\min\{\|\underline{\hat{x}}\| : \underline{x} \in \mathcal{S}\}$
$RB := RB \cup \{\underline{v}\}$
$\mathcal{S} := \mathcal{S} \setminus \{\underline{v}\}$
Add to $\mathcal{S}$ any direct successor of $\underline{v}$
that has all its predecessors in RB.
end

Return RB in order.

Proposition 5.10 *The ordering algorithm terminates with a regular monotone ordering of V.*

Proof: The monotonicity is obvious because of the criterion of selection of vectors entering $\mathcal{S}$. Indeed, if $(p, q) < (p', q')$, a vector $\underline{v}^{p'q'}$ cannot come before $\underline{v}^{pq}$ because it cannot enter $\mathcal{S}$ until $\underline{v}^{pq}$ is in RB.

Now we have to prove the regularity. On each loop, we choose a vector $\underline{v} \in \mathcal{S}$. None of the remaining vectors in $\mathcal{S}$ are direct successors of $\underline{v}$, and all the vectors that are added to $\mathcal{S}$ during the loop are direct successors of $\underline{v}$. Therefore on the following loop, either we choose a vector $\underline{w}$ that is not a direct successor of $\underline{v}$ and thus $\|\underline{\hat{w}}\| \geq \|\underline{\hat{v}}\|$ because $\underline{v}$ was chosen ahead of $\underline{w}$ on the previous loop, or we choose a vector $\underline{w}$ that is a direct successor of $\underline{v}$. Therefore the condition of regularity is satisfied.

Finally we have to check that the algorithm terminates with $|RB| = |V|$. On each loop, exactly one vector is added to RB. So the algorithm has to stop. Suppose now that the algorithm terminates with $RB \subset V$. Select $p := \min\{p : \exists\, j \text{ with } \underline{v}^{pj} \notin RB\}$. Now select $q := \min\{q : \underline{v}^{pq} \notin RB\}$. Clearly, $\underline{v}^{pq}$ can be added to $\mathcal{S}$ since $\underline{v}^{p-1,q} \in RB$ (or $p = 1$) and $\underline{v}^{p,q-1} \in RB$ (or $q = 1$) and thus all its direct predecessors are in RB. Therefore $\mathcal{S} \neq \emptyset$, and the algorithm cannot have ended, a contradiction. So $|RB| = |V|$. □

Theorem 5.2 *RB is a reduced basis of the lattice $\mathcal{L}$.*

To end this section, we observe that stronger properties hold for the reduced basis RB of $\mathcal{L}$ than for a general reduced basis. The next two propositions give the results for a general lattice [40] and for $\mathcal{L}$ respectively. We first recall the notion of determinant of a lattice.

Definition 5.7 *Let $\mathcal{L}_\gamma \subseteq \mathbb{Z}^n$ be a lattice and $(\underline{\gamma}^k)_{k=1}^r$ be one of its bases. The determinant of $\mathcal{L}_\gamma$ is defined by*

$$det\ \mathcal{L}_\gamma = \prod_{k=1}^{r} \|\hat{\underline{\gamma}}_k\|,$$

and is independent of the chosen basis.

Proposition 5.11 *Let $(\underline{\gamma}^k)_{k=1}^r$ be a reduced basis of a r-dimensional lattice $\mathcal{L}_\gamma$ in $\mathbb{R}^n$. Then,*

(i) $\|\underline{\gamma}^1\| \leq 2^{(r-1)/4}(\det \mathcal{L}_\gamma)^{\frac{1}{r}}$

(ii) $\|\underline{\gamma}^1\| \leq 2^{(r-1)/2} \min\{\|\underline{w}\| : \underline{w} \in \mathcal{L}_\gamma, \underline{w} \neq \underline{0}\}$

(iii) $\prod_{k=1}^{r} \|\underline{\gamma}^k\| \leq 2^{r(r-1)/4} \det(\mathcal{L}_\gamma).$

Proposition 5.12 *For the reduced basis $(\underline{v}^{pq})_{p=1,\cdots,P,q=1,\cdots,Q}$ of $\mathcal{L}$,*

(i) $\|\underline{v}^{11}\| \leq 2^{(P+Q-2)/4}(det\ \mathcal{L})^{\frac{1}{PQ}}$

(ii) $\|\underline{v}^{11}\| \leq 2^{(P+Q)/2} \min\{\|\underline{w}\| : \underline{w} \in \mathcal{L}, \underline{w} \neq \underline{0}\}$

(iii) $\prod_{p=1}^{P}\prod_{q=1}^{Q} \|\underline{v}^{pq}\| \leq 2^{PQ(P+Q-2)/4} det\ \mathcal{L}.$

Proof: We start by proving (i). We first need to express the determinant of $\mathcal{L}$. We have $\det \mathcal{L}_A = \prod_{p=1}^{P} \|\hat{\underline{\alpha}}^p\|$ and $\det \mathcal{L}_B = \prod_{q=1}^{Q} \|\hat{\underline{\beta}}^q\|$. Therefore

$$\begin{aligned}
\det \mathcal{L} &= \prod_{p=1}^{P}\prod_{q=1}^{Q} \|\hat{\underline{\alpha}}^p \otimes \hat{\underline{\beta}}^q\| = \prod_{p=1}^{P}\prod_{q=1}^{Q} \|\hat{\underline{\alpha}}^p\|\|\hat{\underline{\beta}}^q\| \\
&= \left(\prod_{p=1}^{P} \|\hat{\underline{\alpha}}^p\|\right)^Q \left(\prod_{q=1}^{Q} \|\hat{\underline{\beta}}^q\|\right)^P \\
&= (\det \mathcal{L}_A)^Q (\det \mathcal{L}_B)^P. \qquad (5.14)
\end{aligned}$$

By Proposition 5.11, we now have

$$\begin{aligned}
\|\underline{\alpha}^1\|\|\underline{\beta}^1\| &\leq 2^{\frac{P-1}{4}}(\det \mathcal{L}_A)^{\frac{1}{P}} 2^{\frac{Q-1}{4}}(\det \mathcal{L}_B)^{\frac{1}{Q}} \\
\|\underline{v}^{11}\| &\leq 2^{\frac{P+Q-2}{4}} \left((\det \mathcal{L}_A)^Q\right)^{\frac{1}{PQ}} \left((\det \mathcal{L}_B)^P\right)^{\frac{1}{PQ}} \\
\|\underline{v}^{11}\| &\leq 2^{\frac{P+Q-2}{4}}(\det \mathcal{L})^{\frac{1}{PQ}},
\end{aligned}$$

using (5.14) for the last line.

We now turn to (ii). Let $\underline{l} = \arg\min\{\|\underline{w}\| : \underline{w} \in \mathcal{L}, \underline{w} \neq \underline{0}\}$. As $\underline{l} \in \mathcal{L}$, there exist $\lambda^{pq} \in \mathbb{Z}$ such that $\underline{l} = \sum_{p=1}^{P}\sum_{q=1}^{Q} \lambda^{pq}\underline{v}^{pq}$. By the Gram-Schmidt procedure, we also know that for all p, q,

$$\underline{v}^{pq} = \hat{\underline{v}}^{pq} + \sum_{(i,j)\prec(p,q)} \mu_{(i,j)(p,q)}\hat{\underline{v}}^{ij}.$$

Therefore $\underline{l}$ can be written as a linear combination of Gram-Schmidt vectors

$$\underline{l} = \sum_{p=1}^{P}\sum_{q=1}^{Q} \nu^{pq}\hat{\underline{v}}^{pq}.$$

Furthermore, if we consider the largest index (i, j) in the ordering $\prec$ for which $|\lambda^{ij}| \geq 1$, we also have that $|\nu^{ij}| \geq 1$. Since the Gram-Schmidt vectors are orthogonal, we also have that

$$\|\underline{l}\|^2 \geq \|\hat{\underline{v}}^{ij}\|^2 = \|\hat{\underline{\alpha}}^{i}\|^2\|\hat{\underline{\beta}}^{j}\|^2. \tag{5.15}$$

From the definition of a reduced basis, we can easily deduce that

$$\|\hat{\underline{\alpha}}^{i}\|^2 \geq \left(\frac{1}{2}\right)^{P} \|\hat{\underline{\alpha}}^{1}\|^2 \tag{5.16}$$

$$\|\hat{\underline{\beta}}^{j}\|^2 \geq \left(\frac{1}{2}\right)^{Q} \|\hat{\underline{\beta}}^{1}\|^2. \tag{5.17}$$

From (5.15) combined with (5.16) and (5.17), we now obtain

$$\|\underline{l}\| \geq \left(\frac{1}{2}\right)^{P+Q} \|\underline{\alpha}^{1}\|^2\|\underline{\beta}^{1}\|^2$$

since $\hat{\underline{\alpha}}^{1} = \underline{\alpha}^{1}$ and $\hat{\underline{\beta}}^{1} = \underline{\beta}$.

Finally we prove (iii). From Proposition 5.11,

$$\prod_{p=1}^{P} \|\underline{\alpha}^{p}\| \leq 2^{P(P-1)/4}\det(\mathcal{L}_A), \text{ and} \tag{5.18}$$

$$\prod_{q=1}^{Q} \|\underline{\beta}^{q}\| \leq 2^{Q(Q-1)/4}\det(\mathcal{L}_B). \tag{5.19}$$

For the lattice $\mathcal{L}$,

$$\begin{aligned}
\prod_{p=1}^{P}\prod_{q=1}^{Q}\|\underline{v}^{pq}\| &= \prod_{p=1}^{P}\prod_{q=1}^{Q}\|\underline{\alpha}^{p}\|\|\underline{\beta}^{q}\| \\
&= \left(\prod_{p=1}^{P}\|\underline{\alpha}^{p}\|\right)^{Q}\left(\prod_{q=1}^{Q}\|\underline{\beta}^{q}\|\right)^{P} \\
&\leq 2^{QP(P-1)/4}(\det\mathcal{L}_A)^{Q}2^{PQ(Q-1)/4}(\det\mathcal{L}_B)^{P} \quad (5.20)\\
&= 2^{PQ(P+Q-2)/4}\det\mathcal{L}, \quad (5.21)
\end{aligned}$$

where the inequality (5.20) comes from (5.18) and (5.19), and the equality (5.21) from (5.14) □

Thus we see that if we want to find the shortest vector of the lattice, (ii) shows that we obtain a guarantee that $\|\underline{v}^{11}\| \leq 2^{\frac{P+Q}{2}}\min\{\|\underline{w}\| : \underline{w}\in\mathcal{L}, \underline{w}\neq 0\}$ while the general bound is $\|\underline{v}^{11}\| \leq 2^{\frac{PQ-1}{2}}\min\{\|\underline{w}\| : \underline{w}\in\mathcal{L}, \underline{w}\neq 0\}$.

5.4 The Banker's problem

5.4.1 Theoretical point of view

Here we show how the results of Section 5.2 can be used to tackle the banker's problem. Specifically we want to solve the problem

Problem 5.1

$$\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{m}\frac{|s_i|}{d_i} & \\
s.t. & \displaystyle\sum_{i=1}^{m}x_{ij}=c_j & \textit{for } j=1,\cdots,n \\
& \displaystyle\sum_{j=1}^{n}x_{ij}=d_i & \textit{for } i=1,\cdots,m \\
& \displaystyle\sum_{j=1}^{n}p_jx_{ij}+s_i=d_i\frac{\sum c_jp_j}{\sum d_i} & \textit{for } i=1,\cdots,m \\
& x\in\mathbb{Z}_+^{mn}, \quad s\in\mathbb{Z}^m. &
\end{array} \tag{5.22}$$

We rescale the last set of equations so that all the coefficients and variables are integer. This yields

$$(\sum_{k=1}^{m}d_k)\sum_{j=1}^{n}p_jx_{ij}+\tilde{s}_i=d_i(\sum_{j=1}^{n}c_jp_j).$$

We can see that this system is of the form studied in Section 5.2 except for the additional variables s_i. However, the system still has the same structure. Indeed, taking $X = \begin{pmatrix} x_{11} & \cdots & x_{1n} & \tilde{s}_1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mn} & \tilde{s}_m \end{pmatrix}$, $A = \begin{pmatrix} 1 & \cdots & 1 & 0 \\ (\sum d_i)p_1 & \cdots & (\sum d_i)p_n & 1 \end{pmatrix}^T$ and $B = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}$, we get the system (5.22), with one more equation, namely $\sum_{i=1}^{m} \tilde{s}_i = 0$. However every solution of (5.22) automatically satisfies this equation.

To use the method of Aardal et al., we need the two reduced bases $\underline{\underline{\alpha}}$ and $\underline{\underline{\beta}}$ of $\mathcal{L}_A$ and $\mathcal{L}_B$ respectively, and also an integer solution $X = Q$ to the equation system of (5.22) with the nonnegativity constraints on $\underline{\underline{x}}$ dropped (this is easily found by inspection). Now the MIP to be solved in Step 2 becomes

$$\begin{aligned} \min \quad & \sum_i \frac{\tilde{s}_i^+}{d_i \sum_k d_k} + \sum_i \frac{\tilde{s}_i^-}{d_i \sum_k d_k} \\ \text{s. t.} \quad & \begin{pmatrix} x_{11} & \cdots & x_{1n} & \tilde{s}_1^+ - \tilde{s}_1^- \\ \vdots & \ddots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mn} & \tilde{s}_m^+ - \tilde{s}_m^- \end{pmatrix} = Q + \underline{\underline{\beta}} \Lambda \underline{\underline{\alpha}} \\ & \underline{\underline{x}}, \underline{\underline{\tilde{s}}}^+, \underline{\underline{\tilde{s}}}^- \geq 0, \ \Lambda \in \mathbb{Z}^{(m-L)\times(n-K)}. \end{aligned}$$

5.4.2 Computational results

Our set of test instances are randomly generated feasible instances of the banker's problem with expected profits uniformly distributed between 5 and 105, based on a real instance. The supplies are uniformly distributed between 1 and 50, and the demands are randomly generated while maintaining $\sum_i d_i = \sum_j a_j$. In all the computations, LiDIA was used to calculate the reduced bases and Cplex, version 6.6 to solve the MIPs, both running on a Sun Sparc Ultra 60.

In Table 5.1, we compare the basis reduction approach with the direct MIP approach on five relatively small instances. The basis reduction approach permits us to solve all of the instances within a few seconds. On the other hand, none of the instances were solved by the direct MIP approach because, for each instance, memory problems were encountered after more than an hour. In fact, the linear programming relaxations stay at zero throughout the enumeration tree (column LB = *lower bound*), but good feasible solutions are found (column UB = *upper bound*).

In Table 5.2, we compare the computation times of the basis composition approach using the product of two small reduced bases, and the direct basis reduction approach. For both approaches, Time part 1 is the time in seconds to construct the alternative MIP formulation by computing the integral basis of the homogeneous system, and Time B&B is the time spent to prove optimality

Sizes		Basis reduction			Direct MIP approach		
n	m	Time B&B	LB	UB	Time B&B	LB	UB
16	5	1	$1.5\,10^{-2}$		***	0	$1.5\,10^{-2}$
16	5	1	$2.5\,10^{-2}$		***	0	$2.5\,10^{-2}$
16	5	0	$8.9\,10^{-3}$		***	0	$8.9\,10^{-3}$
16	5	1	$1.4\,10^{-2}$		***	0	$1.4\,10^{-2}$
16	5	1	$1.7\,10^{-2}$		***	0	$1.7\,10^{-2}$
*** : Memory limit reached. Average time: 2 hours							

Table 5.1: Comparison of direct MIP or basis reduction

with Cplex. For both approaches, the variable selection rule used in Cplex is "pseudo-reduced costs" as it leads to better and more stable results. For the direct basis reduction approach, we observe that the time required for basis reduction becomes a limiting factor as the problem gets bigger. On the other hand, using the composite basis approach, the time required in the MIP step increases, but only slightly. This suggests that the product vectors are perhaps not as short as those from direct basis reduction. But if we consider the total computation time, it is always more interesting to use the composite basis approach.

In Table 3 we run some larger instances with just the composite basis approach as the direct approach is now too time consuming. It appears that for the banker's problem, the difficulty depends on the number m of clients. Instances with $m = 15$ are more difficult than instances with $m = 8$, even for large n. As m approaches 20, proving optimality becomes difficult.

Sizes		Composite basis				Direct basis reduction			
n	m	Time part 1	Time B&B	LB	UB	Time part 1	Time B&B	LB	UB
22	9	1	1	$3.6\,10^{-2}$		552	2	$3.6\,10^{-2}$	
22	9	1	2	$5\,10^{-2}$		527	1	$5\,10^{-2}$	
22	9	1	3	$6\,10^{-2}$		632	1	$6\,10^{-2}$	
22	9	1	1	$3.4\,10^{-2}$		554	1	$3.4\,10^{-2}$	
22	9	1	1	$3.4\,10^{-2}$		519	1	$3.4\,10^{-2}$	
30	12	1	12	$8.3\,10^{-2}$		4121	669	$8.3\,10^{-2}$	
30	12	1	***	$5.5\,10^{-2}$	$6.7\,10^{-2}$	3955	7	$5.5\,10^{-2}$	
30	12	1	36	$6.4\,10^{-2}$		4315	8	$6.4\,10^{-2}$	
30	12	1	10	$6.6\,10^{-2}$		4003	16	$6.6\,10^{-2}$	
30	12	1	11	$7.5\,10^{-2}$		4438	5	$7.5\,10^{-2}$	
15	15	1	24	$2.1\,10^{-1}$		1110	10	$2.1\,10^{-1}$	
15	15	1	438	$1.9\,10^{-1}$		1241	6	$1.9\,10^{-1}$	
15	15	1	12	$2.1\,10^{-1}$		1415	23	$2.1\,10^{-1}$	
15	15	1	128	$2.2\,10^{-1}$		1416	9	$2.2\,10^{-1}$	
15	15	1	123	$3.1\,10^{-1}$		1083	6	$3.1\,10^{-1}$	
60	8	2	11	$1.1\,10^{-2}$		10252	2	$1.1\,10^{-2}$	
60	8	2	7	$9.5\,10^{-3}$		10482	6	$9.5\,10^{-3}$	
60	8	2	17	$1.3\,10^{-2}$		11223	7	$1.3\,10^{-2}$	
60	8	2	7	$1.4\,10^{-2}$		10965	6	$1.4\,10^{-2}$	
*** = more than one hour									

Table 5.2: Comparison of constructed and direct basis reduction

Sizes		Composite basis approach			
n	m	Time B&B	LB	UB	GAP
50	15	156	$6.4\,10^{-2}$		
50	15	140	$4.4\,10^{-2}$		
50	15	57	$5.2\,10^{-2}$		
50	15	***	$5.37\,10^{-2}$	$5.48\,10^{-2}$	2%
50	15	118	$4.5\,10^{-2}$		
30	20	***	$2\,10^{-1}$	$6.6\,10^{-1}$	69%
30	20	***	$8.54\,10^{-2}$	$6.8\,10^{-1}$	88%
30	20	1177	$2.2\,10^{-1}$		
30	20	409	$1.1\,10^{-1}$		
30	20	***	$2\,10^{-1}$	$2.4\,10^{-1}$	17%
*** = more than one hour					

Table 5.3: Critical sizes of the problems solved by this method

Chapter 6

Conclusion

We now review the main results presented in the thesis and propose some directions of future research in the areas explored.

Lifting The second chapter proposed a theory for the lifting of valid inequalities and is based on [45]. We consider the set

$$Z^2(b) = \{z^1 \in X^1, z^2 \in X^2, s \in \mathbb{R}^m_+ : A^1 z^1 + A^2 z^2 \leq b + s\} \tag{6.1}$$

for which we want to generate valid inequalities. As this set may be too complicated, we want to first fix the value of the variables z^2 in order to obtain an easier set Z^1. We generate a valid inequality for Z^1 and then find coefficients for z^2 to lift the variables in the inequality and obtain a valid inequality for $Z^2(b)$.

A first important point is how to fix the variables z^2. We have shown on an example that we have to consider a face of $Z^2(b)$, $Cz^2 \leq e$ for the lifting to be feasible. In other words, we find some valid inequality $Cz^2 \leq e$ for all the feasible points of $Z^2(b)$. Then we set $Cz^2 = e$ to obtain the set Z^1. A main characteristic of our method is that we always want to fix variables at 0. If we do not want to lose generality, we need to change the variables and include variables t with $Cz^2 + t = e$. Fixing $t = 0$ is now equivalent to setting $Cz^2 = e$. The advantage of setting variables to 0 only is that we avoid having to compute two lifting functions. A second important point is the concept of lifting function. We start again with a set of type (6.1) and consider that we fix $z^2 = 0$. We obtain the lower dimensional set

$$Z^1(b) = \{z^1 \in X^1, s \in \mathbb{R}^m : A^1 z^1 \leq b + s\}$$

with a corresponding valid inequality $\pi^1 z^1 \leq \lambda + \nu s$. The lifting function $\phi^1(u)$

is computed by

$$\begin{array}{rlll} \phi^1(u) = & \min & \lambda + \nu s - \pi^1 z^1 \\ & \text{s. t.} & A^1 z^1 \leq b + s - u \\ & & z^1 \in X^1. \end{array}$$

The lifting function allows us to compute the lifting coefficients by finding π^2 such that $\pi^2 v \leq \phi^1(A^2 v)$ for all v in the domain of z^2. We have shown that the set of feasible π^2 is a *polyhedron* and therefore that the computation can be carried out. Another topic treated in the chapter is how to lift several blocks of variables $z^2 \in X^2, z^3 \in X^3, \ldots$ We show how the lifting function changes after each computation of a new block of coefficients. This computation simplifies when the lifting function is *superadditive*. In that case, the blocks can be lifted simultaneously as the lifting function does not change throughout the computation. When $\phi^1(u)$ is not superadditive, one can simplify the computation by finding $\hat{\phi}(u) \leq \phi^1(u)$ with $\hat{\phi}$ superadditive. We can use $\hat{\phi}$ to find the coefficients. The inequalities obtained are weaker but the computation time is reduced since we do not need to recompute the lifting function after each computation of the lifted coefficients of a block.

We come now to the open questions and future research in this area. It is worth exploring in further detail the field of application of our theory, i.e. showing in which case fixing the variables to 0 can be easily carried out and in which case the substitution of variables is hard. It is also interesting to check whether the computation of the lifting coefficients can still be carried out when the continuous variables s are not included in the model. A characterization of when a lifted inequality is facet-defining is also missing in our theory. Such a study is already present in [53]. Finally it is also interesting to study more deeply superadditive lifting. This involves finding an efficient way to characterize superadditive functions or at least an efficient way to compute superadditive lower bounds. The strength of the valid inequalities obtained by superadditive lifting is also a topic that has not been studied.

Single Node Flow Sets In Chapter 3, based on [45], we have studied a particular set that often occurs in mixed integer programming, the single node flow set

$$X^N(n_1, n_2, b, \underline{a}, \underline{u}) = \{(x, y) \in \mathbb{R}_+^{n_1+n_2} \times \mathbb{Z}_+^{n_1+n_2} : \sum_{j \in N_1} x_j - \sum_{j \in N_2} x_j \leq b + s \\ 0 \leq x_j \leq a_j y_j \qquad \}. \quad (6.2)$$

The main results about this set are the well-known flow cover inequalities presented in 1985 by Padberg, Van Roy and Wolsey [55]. What we did in this chapter is to show how we can obtain the flow cover inequalities using two basic operations: the MIR inequalities and the lifting presented in Chapter 2. We show that using the same procedure but with different choices of variables

to fix, we obtain the different variants: the flow cover, the reverse flow cover inequalities and the corresponding inequalities for the integer case. We also showed how we can strengthen these inequalities using superadditive lifting. Finally we obtained valid inequalities for simpler sets from the flow cover inequalities. The simplest set we considered is the basic $0/1$ knapsack set. We showed that the well-known cover inequalities are nothing but particular cases of the flow cover.

There is still some flexibility in the choices we made in our MIR procedure. For example, we always divide the inequality by some a_k before generating the MIR inequality. It could be interesting to examine other possibilities such as the consequences of dividing by any number. A number of papers study valid inequalities for sets related to (6.2). In [45] it is shown that the inequalities they find can often be viewed as particular cases of flow cover inequalities. It may be interesting to see whether other valid inequalities are also particular cases of flow cover inequalities. A more interesting question is, of course, to determine other classes of inequalities for single node flow sets and in particular inequalities that cannot be obtained by the MIR procedure we described. Finally we have not talked about the separation problem which is crucial in practice, i.e. for a given fractional solution x^* to the linear relaxation of (6.2), find a flow cover inequality that cuts off the point x^*.

Group Relaxation Chapter 4 reports the results of [37]. The main subject is to study extended formulations for the group relaxation. We start with the set

$$Y(f) = \{x \in \mathbb{Z}^n_+ : Bx \equiv f \pmod{\Delta}\} \tag{6.3}$$

for which we want to present various extended formulations. The main objects that we use are irreducible solutions of $Y(f)$. An irreducible solution of $Y(f)$ is a solution $y \in Y(f)$ such that for every $z \in Y(f)$, we have $z \not\leq y$. We compute the homogeneous and inhomogeneous irreducible solutions of $Y(b)$ (and $Y(0)$ resp.) and associate a new variable with each solution. This provides the first extended formulation. However we have seen on some examples that the number of new variables added may be large which is impractical. It particularly occurs when several variables have the same coefficient in the equations defining (6.3). The second extended formulation therefore aggregates the variables with the same coefficient in (6.3) into one new variable. Then we compute again the irreducible solutions for the aggregated system and create a new variable for each irreducible solution. This reduces the size of the extended formulation. We propose a third reformulation that also aggregates variables with different coefficients which can reduce even further the size of the extended formulation. Finally a fourth extended formulation is presented that relies on the path structure of the group problem.

In a second step, we analyze the different reformulations and in particular we focus on comparing the projection of the different extended formulations

onto the original space with $\text{conv}(Y(f))$. As among the inhomogeneous irreducible solutions, we find the extreme points of $\text{conv}(Y(f))$ and among the homogeneous irreducible solutions, we find the extreme rays of $\text{conv}(Y(f))$, the projection of the formulation using all the irreducibles is exactly $\text{conv}(Y(f))$. We also prove that this property still holds when we aggregate variables with the same coefficient. In a similar way, the formulation using the path structure of the group satisfies the same property. It means that, in terms of the linear relaxation, reformulating with variables is as strong as using the facets of the group problem. On the other hand, the property does not hold when we aggregate variables with different coefficients. In that case, some fractional extreme points appear in the projection. We studied how to tighten the formulation by adding valid linear inequalities. For this purpose, we have studied the set

$$X_h = \{x_1, x_2, \mu_1, \dots, \mu_t \in \mathbb{Z}_+ : hx_1 + x_2 = b + \sum_{j=1}^{t} a_j \mu_j\}. \tag{6.4}$$

We showed that its convex hull is given by the facet-defining inequalities of the group problem coming from taking the equation (6.4) modulo h. This means that the size of the representation of $\text{conv}(X_h)$ grows with the size of h. In particular we have seen that $\text{conv}(X_1) = LP(X_1)$, where $LP(X_1)$ is the polyhedron obtained from X_1 by dropping the integrality constraints, and the representation of $\text{conv}(X_2)$ needs only one more linear inequality compared with $LP(X_2)$.

In a third step, we focus on the computation of the irreducible solutions used for our extended formulations. In practice it is a slow operation to compute the irreducible solutions. But it is possible to compute them in advance, store them in a table and read off the irreducibles in the table once it is needed. The reading operation is fast and can be done efficiently in an iterative algorithm. In practice, it is possible to store the irreducible solutions of all group problems up to a size of 30. However we can also combine the irreducible solutions of a group mod p and of a group mod q in order to compute the irreducible solutions of a group mod pq.

Finally in a fourth step, we present some computational results related to the effect of using a group relaxation. We tested two algorithms. First we tested the effect of the reformulation in itself. For some problems of the Miplib, we compute the linear relaxation optimum x^*, find a small group problem that cuts off x^* and reformulate using for example the aggregated reformulation then iterate. The results obtained with this algorithm lead to the same type of conclusion as that obtained in the 1970's. Using small groups is not enough to completely close the duality gap of integer programs but it allows to make some steps towards the integer optimum. The second algorithm we tested tries to reformulate with a group relaxation within the Integral Basis Method. We keep an integral tableau representing a feasible solution. At each iteration,

we perform a well chosen reformulation of a group relaxation of a single row. Among the new variables, we look for a good candidate for augmentation. Some examples show that it is possible to find such augmenting vectors by a group reformulation. However aggregating makes the search for augmenting vectors difficult. Therefore we choose to use disaggregated reformulations. But it is difficult to use several disaggregated reformulations subsequently because the size quickly explodes. Therefore the approach is quickly limited.

We see two main topics of future research in this area: one theoretical and one computational. First it should be interesting to study more deeply the facial structure of sets of type (6.4) and see what the corner polyhedron can bring with this respect. In particular a generalization of the result with several variables in the left-hand side should be needed. It should also be interesting to study the effect of adding a right hand side of the type $\sum_i b_i\lambda_i + \sum_j a_j\mu_j$ with $\sum_i \lambda_i = 1$. On the computational side, it seems that some more effort is needed to be able to use aggregated formulations in a clever way and in particular finding augmentation vectors within an aggregated formulation. This would also open the possibility of using aggregate formulations on other problems such as mixed integer programs. All the tests carried out were on binary problems. However it seems more natural to use the group approach on general integer programs. Some computational experiments around this line would also be interesting. Finally it seems important to use the knowledge of larger groups since small groups do not seem to bring as much as wanted. Maybe a partial formulation using the structure of large groups could be promising.

Lattice basis reduction Chapter 5 reports the results of [44]. We study a particular problem, called the banker's problem, that the standard methods cannot solve in reasonable time. On the other hand, using a reformulation of the problem based on lattice basis reduction allows us to solve the problem very rapidly. However when the size of the problem becomes too big, the computation of the reformulation (which can be achieved in polynomial time) becomes a bottleneck. In the chapter we show that we can take advantage of the particular structure of the problem to speed up the computation of the reformulation. The structure of the problem is the following. We look for an integer basis of

$$\mathcal{L} = \{X \in \mathbb{Z}^{m\times n} : XA = \underline{0}, BX = \underline{0}\},$$

with $A \in \mathbb{Z}^{n\times K}$ and $B \in \mathbb{Z}^{L\times m}$. Such a basis can be found by computing an integer basis of

$$\{\underline{x} \in \mathbb{Z}^n : \underline{x}A = \underline{0}\},$$

that we call $\underline{\underline{\alpha}}$ and an integer basis of

$$\{\underline{x} \in \mathbb{Z}^m : B\underline{x} = \underline{0}\},$$

that we call $\underline{\underline{\beta}}$. We have shown that the matrices $X \in \mathcal{L}$ are precisely the matrices of the form $X = \underline{\underline{\beta}}\Lambda\underline{\underline{\alpha}}$, with $\Lambda \in \mathbb{Z}^{(m-L)\times(n-K)}$. This can also be

expressed by saying that $(\underline{\underline{\alpha}}^T \otimes \underline{\underline{\beta}})$ is a basis for the vectorized elements of $\mathcal{L}$. Obtaining $\underline{\underline{\alpha}}$ and $\underline{\underline{\beta}}$ separately is computationally cheaper than computing directly a basis for $\mathcal{L}$.

We also showed in the chapter that the reducedness property is not lost by taking the Kronecker product. In other words, we have shown that if $\underline{\underline{\alpha}}$ and $\underline{\underline{\beta}}$ are reduced bases, then $(\underline{\underline{\alpha}}^T \otimes \underline{\underline{\beta}})$ is also a reduced basis up to a reordering of the vectors. Some computational results were also presented. They show that trying to solve the banker's problem directly with a branch and cut system is hopeless whereas the reformulation can be handled. The computational results also show that using a reduced basis coming from the Kronecker product of two smaller reduced bases is more efficient than computing directly a reduced basis. The respective strengths of the two bases obtained are similar.

Related to this topic, the main open question is to characterize the type of problem that is well solved by a reformulation based on lattice basis reduction and to obtain an explanation to why the method works. We can also ask whether there are other problems for which we can take advantage of a decomposition of the type proposed. Finally a mathematical curiosity is to study lattices of the form $\mathcal{L}(\underline{\underline{\alpha}}^T \otimes \underline{\underline{\beta}})$ and see which properties remain valid with respect to the bases $\underline{\underline{\alpha}}$ and $\underline{\underline{\beta}}$. For example, if one knows a shortest vector $\underline{v}_\alpha$ of $\mathcal{L}(\underline{\underline{\alpha}})$ and a shortest vector $\underline{v}_\beta$ of $\mathcal{L}(\underline{\underline{\beta}})$, is it true that $\underline{v}_\alpha \otimes \underline{v}_\beta$ is the shortest vector of $\mathcal{L}(\underline{\underline{\alpha}}^T \otimes \underline{\underline{\beta}})$?

Bibliography

[1] K. Aardal, R. E. Bixby, C. A. J. Hurkens, A. K. Lenstra, and J. W. Smeltink. Market split and basis reduction: towards a solution of the Cornuéjols-Dawande instances. *INFORMS Journal on Computing*, 12(3):192–202, 2000.

[2] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra. Solving a system of linear diophantine equations with lower and upper bounds on the variables. *Mathematics of Operations Research*, 25(3):427–442, 2000.

[3] K. Aardal, R. Weismantel, and L. Wolsey. Non-standard approaches to integer programming. *Discrete Applied Mathematics*, 123:5–74, 2002.

[4] J. Aráoz, L. Evans, R. E. Gomory, and E. L. Johnson. An approach to integer programming. *Mathematical Programming Series B*, 96(2):377–408, 2003.

[5] A. Atamtürk. Flow pack facets of the single node fixed-charge flow polytope. *Operations Research Letters*, 29(3):107–114, 2001.

[6] A. Atamtürk. On the facets of single-constraint mixed-integer sets. Research Report, Department of Industrial and Operations Research, University of California at Berkeley, 2002.

[7] A. Atamtürk and J. C. Munoz. A study of the lot-sizing polytope. *Mathematical Programming*, 99, 2004.

[8] A. Atamtürk, G. Nemhauser, and M. Savelsbergh. Valid inequalities for problems with additive variable upper bounds. *Mathematical Programming*, 91:145–162, 2001.

[9] E. Balas. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, 8:146–164, 1975.

[10] S. Ceria, C. Cordier, H. Marchand, and L. A. Wolsey. Cutting plane algorithms for integer programs with general integer variables. *Mathematical Programming*, 81:201–214, 1998.

[11] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1996.

[12] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, 1971.

[13] H. Crowder, E. Johnson, and M. Padberg. Solving large scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

[14] R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965.

[15] G. B. Dantzig. Programming in a linear structure. *Econometrica*, 17, 1949.

[16] L. Evans, R. E. Gomory, and E. Johnson. Corner polyhedra and their connection with cutting planes. *Mathematical Programming Series B*, 96:321–339, 2003.

[17] M. X. Goemans. Valid inequalities and separation for mixed 0-1 constraints with variable upper bounds. *Operations Research Letters*, 8:315–322, 1989.

[18] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 64:275–278, 1958.

[19] R. E. Gomory. Some polyhedra related to combinatorial problems. *Linear Algebra and Its Applications*, 2:451–558, 1969.

[20] R. E. Gomory and E. Johnson. Some continuous functions related to corner polyhedra I. *Mathematical Programming*, 3:23–85, 1972.

[21] R. E. Gomory and E. Johnson. Some continuous functions related to corner polyhedra II. *Mathematical Programming*, 3:359–389, 1972.

[22] R. E. Gomory and E. Johnson. The group problem and subadditive functions. *Mathematical Programming*, pages 157–184, 1973.

[23] R. E. Gomory and E. Johnson. T-space and cutting planes. *Mathematical Programming Series B*, 96:341–375, 2003.

[24] G. A. Gorry, W. D. Northup, and J. F. Shapiro. Computational experience with a group theoretic integer programming algorithm. *Mathematical Programming*, 4:171–192, 1973.

[25] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Lifted cover inequalities for 0–1 integer programs: computation. *INFORMS Journal on Computing*, 10:427–438, 1998.

[26] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Lifted cover inequalities for mixed 0–1 integer programs. *Mathematical Programming*, 85(3):439–467, 1999.

[27] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Sequence independent lifting in mixed integer programming. *Journal of Combinatorial Optimization*, 4:109–129, 2000.

[28] P. L. Hammer, E. L. Johnson, and U. N. Peled. Facets of regular 0-1 polytopes. *Mathematical Programming*, 8:179–206, 1975.

[29] U.-U. Haus, M. Köppe, and R. Weismantel. A primal all-integer algorithm based on irreducible solutions. *Mathematical Programming Series B*, 96:205–246, 2003.

[30] R. G. Jeroslow. An introduction to the theory of cutting planes. *Annals of Discrete Mathematics*, 5:71–95, 1979.

[31] E. L. Johnson. Cyclic groups, cutting planes and shortest paths. *Mathematical Programming*, pages 185–211, 1973.

[32] A. Joux. La réduction de réseaux en cryptographie. PhD thesis, Ecole Polytechnique, Palaiseau, France, 1998., 1998.

[33] D. Klabjan and G. L. Nemhauser. A polyhedral study of variable upper bounds. *Mathematics of Operations Research*, 27:711–739, 2002.

[34] M. Köppe. Erzeugende Mengen für gemischt-ganzzahlige Programme. Diploma thesis, Otto-von-Guericke-Universität Magdeburg, 1999.

[35] M. Köppe. *Exact Primal Algorithms for General Integer and Mixed-Integer Linear Programs*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2003. Published by Shaker Verlag, Aachen, 2003.

[36] M. Köppe. Tables of primitive partition identities. Available from `http://www.math.uni-magdeburg.de/ mkoeppe/art/ppi/`, 2003.

[37] M. Köppe, Q. Louveaux, R. Weismantel, and L. Wolsey. Extended formulations for gomory corner polyhedra. University of Magdeburg, Department of Mathematics, Preprint 04-04, 2004.

[38] P. Lancaster and M. Tismenetsky. *The theory of matrices*. Computer Science and Applied Mathematics. Academic Press Inc., Orlando, Fla., second edition, 1985.

[39] A. Land and A. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[40] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[41] J. K. Lenstra, A. H. G. Rinnoy Kan, and A. Schrijver. *History of Mathematical Programming: A Collection of Personal Reminiscences*. Elsevier Science Publishers B. V., Amsterdam, 1991.

[42] LiDIA. *A library for computational number theory*. TH Darmstadt/ Universität des Saarlandes, Fachbereich Informatik, Institut für Theoretische Informatik, 1999.

[43] Q. Louveaux. Résolution de problèmes d'optimisation en nombres entiers par un algorithme de réduction de base dans les treillis. Mémoire de la Faculté des Sciences Appliquées, Université catholique de Louvain, 1999.

[44] Q. Louveaux and L. A. Wolsey. Combining problem structure with basis reduction to solve a class of hard integer programs. *Mathematics of Operations Research*, 27(3):470–484, 2002.

[45] Q. Louveaux and L. A. Wolsey. Lifting, superadditivity, mixed integer rounding and single node flow sets revisited. *4OR*, 1(3):173–207, 2003.

[46] H. Marchand and L. A. Wolsey. The 0–1 knapsack problem with a single continuous variable. *Mathematical Programming*, 85:15–33, 1999.

[47] H. Marchand and L. A. Wolsey. Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49(3):363–371, 2001.

[48] A. J. Miller, G. L. Nemhauser, and M. W. P. Savelsbergh. A multi-item production planning model with setup times: algorithms, reformulations, and polyhedral characterizations for a special case. *Mathematical Programming*, 95(1):71–90, 2003.

[49] A. J. Miller, G. L. Nemhauser, and M. W. P. Savelsbergh. On the polyhedral structure of a multi-item production planning model with setup times. *Mathematical Programming Series B*, 94:375–406, 2003.

[50] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[51] G. L. Nemhauser and L. A. Wolsey. A recursive procedure to generate all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46:379–390, 1990.

[52] M. Newman. *Integral Matrices*, volume 45 of *Pure and Applied Mathematics, a Series of Monographs and Textbooks*. Academic Press, 1972.

[53] M. Oosten. *A polyhedral approach to grouping problems*. PhD thesis, University of Maastricht, 1996.

[54] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

[55] M. W. Padberg, T. J. Van Roy, and L. A. Wolsey. Valid inequalities for ficed charge problems. *Mathematical Programming*, 33:842–861, 1985.

[56] A. Schrijver. *Theory of linear and integer programming*. Wiley, Chichester, 1986.

[57] J. I. Stallaert. The complementary class of generalized flow cover inequalities. *Discrete Applied Mathematics*, 77:73–80, 1997.

[58] T. J. van Roy and L. A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14:199–213, 1986.

[59] T. J. van Roy and L. A. Wolsey. Solving mixed 0-1 programs by automatic reformulation. *Operations Research*, 35:45–57, 1987.

[60] H. P. Williams. *Model Building in Mathematical Programming*. Wiley & Sons, Chichester, 1993.

[61] L. A. Wolsey. A number theoretic reformulation and decomposition method for integer programming. *Discrete Mathematics*, 7:393–403, 1974.

[62] L. A. Wolsey. Faces for the linear inequalities in 0-1 variables. *Mathematical Programming*, 8:165–178, 1975.

[63] L. A. Wolsey. Facets and strong valid inequalities for integer programs. *Operations Research*, 24:367–372, 1976.

[64] L. A. Wolsey. Valid inequalities and superadditivity for 0-1 integer programs. *Mathematics of Operations Research*, 2:66–77, 1977.

[65] L. A. Wolsey. Submodularity and valid inequalities in capacitated fixed charge networks. *Operations Research Letters*, 8:119–124, 1989.

[66] L. A. Wolsey. Valid inequalities for mixed integer programs with generalised upper bound constraints. *Discrete Applied Mathematics*, 25:251–261, 1990.