Subadditive Approaches to Mixed Integer Programming

by

Babak Moazzez

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Applied Mathematics

Carleton University
Ottawa, Ontario

Abstract

Correctness and efficiency of two algorithms by Burdet and Johnson (1975) and Klabjan (2007) for solving pure integer linear programming problems are studied. These algorithms rely on Gomory's corner relaxation and subadditive duality. Examples are given to demonstrate the failure of these algorithms on specific IPs and ways of correcting errors in some cases are given. Klabjan's Generator Subadditive Functions have been generalized for mixed integer linear programs. These are (subadditive) dual feasible functions which have desirable properties that allow us to use them as certificates of optimality and for sensitivity analysis purposes. Generating certificates of optimality has been done for specific families of discrete optimization problems such as set covering and knapsack problems using (mixed) integer programming formulations. Some computational results have been reported on these problems at the end. Also, subadditive generator functions have been used for finding a finite representation of the convex hull of MILPs which is a generalization of Bachem and Schrader's and Wolsey's works on finite representations and value functions of (mixed) integer linear programs.

Babak Moazzez, Carleton University, Thesis (PhD), 2014, Pages: 117.

Acknowledgements

I would like to express my special appreciation and thanks to my supervisor Professor Dr. Kevin Cheung, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless. You have always supported me academically and financially and this thesis would not have been possible without your help.

I would also like to thank my committee members Dr. Boyd, Dr. Chinneck, Dr. Steffy and Dr. Wang. I want to thank Dr. Mehrdad Kalantar and Mr. Ryan Taylor from School of Mathematics and Statistics for all their help.

A special thanks to my family. Words cannot express how grateful I am to my mother, father and brother for all of the sacrifices that you've made on my behalf.

Last but not least, I would like express appreciation to my beloved beautiful wife Azadeh who supported me with her love and patience in all stages of my life. Without you, I have no idea what I would do.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Most optimization problems in industry and science today are solved using commercial software. Such software packages are very reliable and fast but they generally lack one important feature: verifiability of computations and/or optimality. Commercial solvers rarely, if at all, release to the user the exact procedure that has been followed to solve a specific problem. Without this, on what basis does one trust on a solution generated using these solvers? Even if one uses his/her own code for optimization, there are possibilities of bugs, mistakes and etc.

The answer to this problem is to obtain a certificate of optimality: some information that proves the optimality of the solution at hand, but easier to check compared to the original optimization problem. In the case of a linear program, a certificate is given by an optimal dual vector. In the case of (mixed) integer linear programming, we could use an optimal subadditive dual function. However, finding such a function is not trivial whatsoever. An obvious example would be the value function of the MILP but finding this function is an arduous task. The best way would be to limit the large family of functions for the subadditive dual to a smaller family. This was done previously by Klabjan in [35]. Klabjan defined a class of functions called subadditive generator functions which are sufficient for getting strong duality in the case of pure integer programs with non-negative entries. These functions create very good (we will define later what good means) certificates for pure integer programming problems but there are restrictions on the input data that make it impossible to apply the method to general MILPs.

In this thesis we generate certificates of optimality for different families of mixed integer programming problems. Previously all certificates generated for these problems consisted of a branch and bound tree and some cutting planes. So in order to certify the optimality of an instance, one had to go over nodes of the tree and prove that the cutting planes are valid for the problem. We use subadditive duality for this purpose and our certificates are much easier to be verified.

Other than optimality certificates, subadditive generator functions are useful tools for sensitivity analysis in mixed integer optimization. Notions such as reduced cost will carry over from linear programming using these functions.

In this thesis, we generalize Klabjan's functions for mixed integer programs with absolutely no restriction on the input data so that they can be used as optimality certificates for general mixed integer linear programs.

Now the question that arises is how to find an optimal subadditive generator function. To find such a function, we need a family of cutting planes which, when added to an LP relaxation of the original problem, reveals an optimal solution. Suppose that an MILP has been solved to optimality using the branch and bound method. We try to extract all the necessary cuts from inside the branch and bound tree. We show that with the branch and bound tree and optimal solution of the MILP at hand, we have enough information to generate a certificate and there is no need to optimize from scratch. These cuts that are extracted from the tree will not be valid for the original MILP until we lift them to become valid. Lifting valid inequalities is an important part of the procedure. After this is done, the cuts are used to generate an optimal subadditive dual function which represents an equivalent dual vector with an objective value equal to the optimal value of the MILP but easier to check compared with the original problem because of the structure of these functions.

We have also used these functions to find a finite representation of the convex hull of feasible points to a mixed integer program which is an extension to Wolsey's results in [45] and Bachem and Schrader's results in [2] on finite representation of convex hull of the set of feasible points of a pure integer programming problem. Each dual feasible subadditive function gives us a valid inequality for the original MILP. Using a finite number of these valid inequalities, we can describe the convex hull as a system of linear inequalities.

Klabjan's work and the definition of subadditive generator functions have their roots in the work of Burdet and Johnson in [15]. Reviewing their work in [16], we discovered some errors in their algorithm and we made attempts to correct them. Burdet and Johnson's work heavily relies on the group relaxation of a mixed integer programming. Since the structure of valid inequalities for arbitrary integer programs is too complex to be studied, researchers realized early on the necessity of understanding more restricted programs [10]. This would be very useful in branch and bound and cutting plane methods. The idea of group (corner) relaxation was first proposed by Gomory in his well known paper [24]. In spite of being NP-hard [23], the group problem has always been of great interest and in recent years, various versions have been defined and studied in depth; most of them used to generate more powerful cutting planes and unravel the structure of these polyhedra. For example, see Burdet and Johnson [15, 16, 32], Cornuéjols [11, 19], Wolsey [46], Balas [3], Klabjan [35], Shapiro [42, 43], Jeroslow [30, 31], Blair [13] and Basu, Molinaro and Köppe [7–11]. For a recent survey, see Dey and Richard [39].

Many different algorithms were designed for solving the group relaxation among which some use lifting as the main tool (see [15,32]). A subadditive function is defined and lifted at each iteration along with parameter optimization to avoid enumeration of the entire feasible set. Although there are no numerical reports on the efficiency of these algorithms, they form some of the standard methods for solving the group problem. Johnson's algorithm [32] for solving the group problem of a single constrained mixed integer program was published in 1973. Burdet and Johnson [15] extended this method for the group problem of a pure integer program. This algorithm does not use the group structure. However, the running time and the theory behind the algorithm are dependent on the group structure. The key point was the use of a finite set called the Generator Set to keep the function which is being lifted subadditive. They extended this algorithm in 1977 [16] to solve integer programs. This algorithm has some deficiencies and flaws which make it fail to work in general. We present examples that show that the algorithm fails to solve some pure integer programming problems. Ways of correcting these errors are given in some cases. However the corrected algorithm may not be very useful in general.

In 1994, Klabjan [35] introduced a method very similar to Burdet and Johnson's method to solve a pure integer program with non-negative constraint coefficients and right-hand side values. This algorithm has some errors as well. We present examples which show that Klabjan's algorithm fails to solve some integer programs and we also correct his algorithm.

We correct and improve two algorithms in this thesis for solving special families of integer programming problems. We provides examples of failure and ways of correcting the errors for both algorithms. To our knowledge, previously nobody has discussed the efficiency nor the correctness of these algorithms.

This thesis is organized as follows:

Chapter 2 gives an introduction on the corner (group) relaxation of (mixed) integer programming and subadditive duality for mixed integer programming. These two concepts form most of the underlying theory for this thesis. They will be used frequently in all chapters. We also mention some recent advances in group theoretic approach in mixed integer programming.

In Chapter 3, we describe the algorithm by Burdet and Johnson for integer programming along with two older algorithms as the basis of that work, and present some examples for the failure of the algorithm. In some cases, ways of correcting these errors are given as well. Efficiency and geometric interpretation of the algorithm will also be discussed.

Chapter 4 focuses on Klabjan's work, its errors and ways of correcting them. We give a thorough theoretic proof for Klabjan's algorithm for solving pure integer programs with non-negative entries with examples demonstrating the failure in specific cases. At the end, we present a corrected version of the algorithm which will work for any pure integer program with non-negative entries.

Chapter 5 consists of three main parts: In the first part, we generalize Klabjan's subadditive generator functions to general mixed integer programming problems. In the second part, we use these functions to find a finite representation for the convex hull of feasible solutions to a mixed integer program which is a generalization of results of Bachem and Schrader [2], and Wolsey [45]. Finally, in the last section, we use generalized subadditive functions as certificates of optimality for mixed integer programming problems and also as a tool for sensitivity analysis for MILPs. This is done by extracting enough cutting planes from the branch and bound tree that has been used for solving the optimization problem and using generalized subadditive generator functions and subadditive duality. We give a new algorithm for this purpose and present some computational results of our work at the end.

In Chapter 6, we discuss some unanswered questions and future research directions.

Chapter 2

Preliminaries

In this chapter, we will give some preliminaries on the group problem of (mixed) integer programming and subadditive duality. These two concepts are used frequently throughout the thesis and form the underlying theory of different chapters.

Throughout this thesis, we use the following notations: if a and b are two vectors in $\mathbb{R}^n$, ab will denote their inner product. Also if A is a matrix and I is a set of indices, A_I will denote the submatrix of A with columns indexed from I. Similarly, x_I will mean the same for vector x.

2.1 Group Problem of Integer Programming

Take positive integers m and n, and let $A \in \mathbb{Q}^{m\times n}$ where $\mathbb{Q}^{m\times n}$ denote the set of all $m \times n$ matrices with rational entries.[1] Let $b \in \mathbb{Q}^m$ and $c \in \mathbb{Q}^n$. Define the Pure

[1]In this thesis we always consider (mixed) integer programs with rational entries. It is a result of Meyer [37] that if a (mixed) integer program has rational entries, then the convex hull of the feasible points is a polyhedron (finitely generated). If we drop this condition, the structure of the latter set is not trivial.

Integer Programming Problem (IP) as:

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \text{ integer.} \end{aligned} \tag{2.1}$$

For example

$$\begin{aligned} \min \quad & x_1 - x_3 \\ \text{s.t.} \quad & x_1 + \tfrac{2}{3}x_2 - x_3 = 10 \\ & -x_2 + x_3 = 3 \\ & x_1, x_2, x_3 \geq 0, \text{ integer} \end{aligned}$$

is an IP with $c = [1, 0, -1]^T, x = [x_1, x_2, x_3]^T, A = \begin{bmatrix} 1 & \frac{2}{3} & -1 \\ 0 & -1 & 1 \end{bmatrix}$ and $b = [10, 3]^T$.

Note that other forms of integer programming problems can be transformed into form of IP (2.1) using slack/surplus variables and/or defining new non-negative variables.

Choose m linearly independent columns from matrix A, denote their index set by B and the index set of remaining columns by N. (B is usually the optimal linear programming basis, but any other basis could be used.) Now we can write A as $A = (A_B, A_N)$ and we have $A_B x_B + A_N x_N = b$. If we relax the non-negativity of x_B, then the following is a relaxation of IP(2.1):

$$\min \{z^* + \bar{c}_N x_N : A_B^{-1} A_N x_N \equiv A_B^{-1} b \ (mod\ 1)\} \tag{2.2}$$

where $z^* = c_B A_B^{-1} b$ and $\bar{c}_N = c_N - c_B A_B^{-1} A_N$. (Addition modulo 1 means that we consider only the fractional part of $a + b$. $\overline{a + b}$ is the fractional part of $a + b$. We know that $\overline{a+b} \equiv a+b \pmod 1$. Addition modulo 1 means to map $a+b$ to $\overline{a+b}$.) To see that this is indeed a relaxation, note that

$$\begin{aligned} G &= \{(x_B, x_N) : A_B x_B + A_N x_N = b,\ x_N \geq 0, x_B, x_N \text{ integer } \} \\ &= \{(x_B, x_N) : x_B + A_B^{-1} A_N x_N = A_B^{-1} b,\ x_N \geq 0, x_B, x_N \text{ integer } \} \end{aligned}$$

can be written in x_N-space as:

$$\{x_N \in \mathbb{Z}_+^N : A_B^{-1} A_N x_N \equiv A_B^{-1} b \ (mod\ 1)\}.$$

Since the non-negativity constraints are removed for basic variables in (2.1), the set of feasible solutions to (2.2) is a subset of the set of feasible solutions to (2.1). The rest of the operations performed above do not change the set of feasible solutions.

Problem (2.2) is called the *Corner Relaxation* or sometimes *The Group Problem* of IP (2.1). This idea was first proposed by Gomory in [24].

Theorem 1. [23] *The corner relaxation (2.2) is strongly NP-hard.*[2]

Although the group problem is NP-hard, it is useful for generating valid inequalities for the corresponding (mixed) integer program [39] and the bound can be used in any branch and bound based algorithm. Also algorithms for solving the group problem can be generalized to solve a (mixed) integer program as we will see later in Section 3.3.

For mixed integer programs, the corner relaxation is defined similarly. Consider

[2] A problem is said to be strongly NP-hard if it remains NP-hard when its input data are polynomially bounded.

the following Mixed Integer Linear Program (MILP):

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x_j \geq 0 \qquad j = 1, ..., n \\ & x_j \text{ integer} \quad \text{for all } j \in I \end{aligned} \tag{2.3}$$

where $I \subseteq \{1, ..., n\}$ denotes the index set of integer variables. Let $J = \{1, ..., n\} \backslash I$. Given an LP basis B, the set G can be reformulated as

$$G_M = \{(x_B, x_{NI}, x_{NJ}) \in \mathbb{Z}^B \times \mathbb{Z}_+^{NI} \times \mathbb{R}_+^{NJ} \mid x_B + A_B^{-1} A_N x_N \equiv A_B^{-1} b \ (mod\ 1)\}$$

where $NI = N \cap I$ and $NJ = N \cap J$. Now the lower bound on the basic variables may be relaxed. For basic variables that are continuous, relaxing the lower bound is equivalent to relaxing the corresponding constraint. So one may assume that $B \cap J = \emptyset$ by discarding constraints with continuous basic variables if necessary. The corner polyhedron associated with A_B then takes the form:

$$\{(x_{NI}, x_{NJ}) \in \mathbb{Z}_+^{NI} \times \mathbb{R}_+^{NJ} \mid A_B^{-1} A_N x_N \equiv A_B^{-1} b \ (mod\ 1)\}.$$

To solve the group problem, Gomory suggested a network model [24]. Shapiro [42] represented a dynamic programming algorithm to solve the group problem of an integer program based on such a network.

Although there are cases that the solutions of IP (2.1) and the group problem (2.2) are the same (this is called asymptotic theory for mixed integer programming), this does not happen very often. In this case, one may use the valid inequalities from group relaxation to trim down the search space or use Extended Group Relaxation

[46] or Infinite Group Relaxation [39].

2.1.1 Recent Advances Related to Group Problem

We will state a few theorems to make the reader familiar with the most recent advances in this area. Although these theorems and results will not be used in the rest of the thesis, they will give the reader a profound understanding of the concept. The reader can skip this section without any problem.

Any valid inequality for the group relaxation, is also valid for the original (mixed) integer program. So these valid inequalities can always be used in algorithms for solving the original problem such as branch and bound.

Also subadditivity is a very important property used to describe group polyhedra as system of inequalities and/or equalities. The oldest theorem of this kind belongs to Gomory in [24].

Let $\mathcal{G}$ be an abelian group with zero element o and $g_0 \in \mathcal{G}\backslash\{o\}$. *The Master Corner Polyhedron* $P(\mathcal{G}, g_0)$ is defined as the convex hull of all non-negative integer valued functions[3] $t : \mathcal{G} \to \mathbb{Z}_+$ satisfying

$$\sum_{g \in \mathcal{G}} g t(g) = g_0. \tag{2.4}$$

It is known from [24] that if a function $\pi : \mathcal{G} \to \mathbb{R}$ defines a valid inequality for master group polyhedron, then $\pi(g) \geq 0$ for all $g \in \mathcal{G}$ and $\pi(o) = 0$. The valid inequality is in the form

$$\sum_{g \in \mathcal{G}} \pi(g) t(g) \geq 1. \tag{2.5}$$

[3]Consider all functions $t : \mathcal{G} \to \mathbb{R}_+$ as the original space. Then the convex hull of all functions $t : \mathcal{G} \to \mathbb{Z}_+$ will be well defined.

Actually all faces of the $P(\mathcal{G}, g_0)$ are either in the form (2.5) or in the form $t(g) \geq 0$. Gomory shows the following theorem:

Theorem 2. *π defines a face of $P(\mathcal{G}, g_0)$ with $g_0 \neq 0$ if and only if it is a basic feasible solution to the following system:*

$$\begin{aligned}
&\pi(g_0) = 1 && \\
&\pi(g) + \pi(1-g) = 1 && g \in \mathcal{G}\backslash\{o\}, g \neq g_0 \\
&\pi(g) + \pi(g') \geq \pi(g+g') && g, g' \in \mathcal{G}\backslash\{o\} \\
&\pi(g) \geq 0. &&
\end{aligned}$$

Johnson has generalized the above theorem for the master group polyhedron of a mixed integer program in [34].

A valid inequality π is called *minimal* if there is no other valid inequality μ different from π with $\mu(g) \leq \pi(g)$ for all $g \in \mathcal{G}$. Also a valid inequality π is called *extreme* if $\pi = \frac{1}{2}\pi_1 + \frac{1}{2}\pi_2$ implies $\pi = \pi_1 = \pi_2$.

Johnson proved the following two theorems in [34] in 1974 that we state them without proof.

Theorem 3. *Minimal valid inequalities for the master group problem are subadditive valid inequalities.*

Theorem 4. *Extreme valid inequalities for the master group problem are minimal valid inequalities.*

If we replace $\mathcal{G}$ with $\mathbb{R}^N$, infinite group of real N-dimensional vectors with addition modulo 1 component wise, the group polyhedron will be the convex hull of all non-negative integer points t satisfying

$$\sum_{g \in \mathbb{R}^N} g t(g) = g_0$$

where t should have a finite support. This is called *Infinite Group Polyhedron* and the corresponding optimization problem is called *Infinite Group Problem (Relaxation).* This problem can also be written in the form

$$\begin{aligned} & -g_0 + \sum_{g \in \mathbb{R}^N} g t(g) \in \mathbb{Z}^N \\ & t(g) \in \mathbb{Z}_+ \text{ for all } g \in \mathbb{R}^N \\ & t \text{ has finite support} \end{aligned} \tag{2.6}$$

for dimension N. Valid inequalities arising from infinite group relaxation are of great interest even for the one dimensional case. Let $IR(g_0)$ denote the infinite relaxation (2.6) for $N = 1$.

Since the structure of valid inequalities for arbitrary integer programs is too complex to be studied, researchers realized early on the necessity of understanding more restricted programs [10].

Gomory proved Theorem 2 in 1972. For a long time, the optimization research community was not interested in the master group relaxation because its structure relies on $\mathcal{G}$, making it difficult to analyze. The infinite group relaxation reduces the complexity of the system by considering all possible elements in $\mathbb{R}^N$. Therefore, it is completely specified by the choice of g_0.

A function $\pi : \mathbb{R}^N \to \mathbb{R}$ is periodic if $\pi(x) = \pi(x + w)$ for all $x \in [0, 1]^N$ and $w \in \mathbb{Z}^N$. Also, π is said to satisfy the symmetry condition if $\pi(g) + \pi(g_0 - g) = 1$ for all $g \in \mathbb{R}^N$.

Theorem 5. *(Gomory and Johnson [25]) Let $\pi : \mathbb{R}^N \to \mathbb{R}$ be a non-negative function. Then π is a minimal valid function for infinite group relaxation if and only if $\pi(0) = 0$, π is periodic, subadditive and satisfies the symmetry condition.*

For the single constraint case, the following theorem was given by Gomory and Johnson [26] and is called the 2-Slope theorem.

Theorem 6. *Let $\pi : \mathbb{R} \to \mathbb{R}$ be a minimal valid function. If π is a continuous piecewise linear function with only two slopes, then π is extreme.*

Cornuéjols has generalized this theorem to the two dimensional case.

Theorem 7. *(Cornuéjols' 3-Slope Theorem [19]) Let $\pi : \mathbb{R}^2 \to \mathbb{R}$ be a minimal valid function. If π is a continuous 3-slope function with 3 directions, then π is extreme.*

Recently Köppe et al [10] generalized this for any dimension.

Definition 8. *A function $\theta : \mathbb{R}^k \to \mathbb{R}$ is genuinely k-dimensional if there does not exist a function $\varphi : \mathbb{R}^{k-1} \to \mathbb{R}$ and a linear map $T : \mathbb{R}^k \to \mathbb{R}^{k-1}$ such that $\theta = \varphi \circ T$.*

Theorem 9. *Let $\pi : \mathbb{R}^k \to \mathbb{R}$ be a minimal valid function that is piecewise linear with a locally finite cell complex and genuinely k-dimensional with at most $k + 1$ slopes. Then π is a facet (and therefore extreme) and has exactly $k + 1$ slopes.*

In recent years, relation between the facets of master group polyhedron and infinite relaxation polyhedron has been of great interest. Let $\mathcal{G} = C_n$ be a finite cyclic group of order n which can be considered as a subgroup of the group used in $IR(g_0)$. The following two theorems by Gomory and Johnson [26] go back to 1972.

Theorem 10. *If π is extreme for $IR(g_0)$, then $\pi|_{\mathcal{G}}$ is extreme for $P(\mathcal{G}, g_0)$.*

Theorem 11. *If $\pi|_{\mathcal{G}}$ is minimal for $P(\mathcal{G}, g_0)$, then π is minimal for $IR(g_0)$.*

However Dey, Li, Richard and Miller [21] proved the following in 2009:

Theorem 12. *If π is minimal and $\pi|_{C_{2^k n}}$ is extreme for $P(C_{2^k n}, g_0)$ for all $k \in \mathbb{N}$, then π is extreme for $IR(g_0)$.*

In 2012, Basu, Hildebrand and Köppe showed this last theorem in [8].

Theorem 13. *If π is minimal and $\pi|_{C_{4n}}$ is extreme for $P(C_{4n}, g_0)$, then π is extreme for $IR(g_0)$.*

The most important question is if these results are also true in higher dimensions (see [9]). For more details on the group problem and its different variations, see the survey by Richard and Dey [39].

2.2 Subadditive Duality

Definition 14. *A function f is called subadditive if for any x and y we have:*

$$f(x+y) \leq f(x) + f(y).$$

For MILP (2.3), the subadditive dual is defined as:

$$\begin{aligned} \max \quad & F(b) \\ \text{s.t.} \quad & F(a_j) \leq c_j \quad \text{for all } j \in I \\ & \overline{F}(a_j) \leq c_j \quad \text{for all } j \in J \\ & F \in \Gamma^m \end{aligned} \tag{2.7}$$

where

$$\Gamma^m = \{F : \mathbb{R}^m \to \mathbb{R} | F \text{ subadditive and } F(0) = 0\}$$

and $\overline{F}(d) = \limsup_{\delta \to 0^+} \frac{F(\delta d)}{\delta}$ and a_j is the j-th column of A. The maximum is taken over all functions $F \in \Gamma^m$.

Theorem 15. [28] (Weak Duality) *Suppose x is a feasible solution to MILP and F is a feasible solution to the subadditive dual (2.7). Then $F(b) \leq cx$.*

Theorem 16. [28] (Strong Duality) *If the primal problem (2.3) has a finite optimum, then so does the dual problem (2.7) and they are equal.*

Theorem 17. [28] (Complementary Slackness) *Suppose x^* is a feasible solution to MILP (2.3) and F^* is a feasible solution to the subadditive dual (2.7). Then x^* and F^* are feasible if and only if*

$$\begin{aligned} & x_j^*(c_j - F^*(a_j)) = 0 && \textit{for all } j \in I \\ & x_j^*(c_j - \overline{F}^*(a_j)) = 0 && \textit{for all } j \in J \\ & F^*(b) = \sum_{j \in I} F^*(a_j)x_j^* + \sum_{j \in J} \overline{F}^*(a_j)x_j^*. \end{aligned} \tag{2.8}$$

In the case of linear programming, it is easy to see that the function $F_{LP}(d) = \max_{v \in \mathbb{R}^m}\{vd : vA \leq c\}$ is a feasible subadditive function and using this function, the subadditive dual will reduce to the LP duality.

The Subadditive dual plays a very important role in the study of mixed integer programming. Any feasible solution to the dual gives a lower bound to the MILP(2.3). A dual feasible function F with $F(b) = z_{IP}$ is a certificate of optimality for the MILP(2.3). It will be equivalent to the dual vector in LP and the reduced cost of a column i can be defined as $c_i - F(a_i)$ and most of the other properties from LP can be extended to MILP, for example complementary slackness, and the fact that all optimal solutions can be found only among the columns i with $c_i = F(a_i)$, if F is optimal.

Chapter 3

Burdet and Johnson's Algorithm: Deficiencies and Fixes

After the introduction of the group problem by Gomory in 1972, a few methods were introduced by different people to solve the group problem. Gomory himself introduced a network model for the problem. The work of Shapiro [43] suggest a dynamic programming algorithm for optimization on corner polyhedron, however Bell [12] embedded this problem in a finer group and tried to solve this problem algebraically. Other than these methods, there was a way to understand the structure of these polyhedra and that was to make a table of facets. In higher dimensions it was an arduous task. Alain Burdet suggested [33] that instead of using tables, one could find an optimal solution to the corner relaxation via lifting subadditive functions and using subadditive duality. The first algorithm of this kind belongs to Johnson [33]. He introduced a new algorithm in 1973 [32] to solve the group problem of a mixed integer program with only one constraint. The main idea was to use the subadditive duality, lift (increase) a dual feasible function (starting from zero) until either duality constraints or subadditivity of the function are violated. At this step the function is fixed for some points and the algorithm repeats until it finds the optimal value for the problem. Subadditive duality (which is a strong duality for mixed integer programming) plays the most important role in this algorithm. However this algorithm is only for a single constraint group problem of mixed integer programming.

One year later Johnson and Burdet [15] extended this method for the group problem of a pure integer program with more than one constraint. The main property of this algorithm is that it never uses the group structure. However, the running time and the theory behind the algorithm are dependent on the group structure. The key point was the use of a subadditive function along with a finite set called the *Generator Set*. This set helps us maintain the subadditivity of the function that we are using plus the duality constraints.

In 1977, Burdet and Johnson [16] extended this algorithm to solve integer programs. The algorithm uses the same idea of lifting a subadditive dual feasible function. Although this method works for a large family of problems, it has some deficiencies and flaws which make it fail to work in general. We represent examples that show that the algorithm fails to solve some pure integer programming problems. Ways of fixing these problems are given in some cases, however the corrected algorithm may not be very useful in general.

In this chapter we first explain Johnson's algorithm for the group problem of an MILP with a single constraint. Then in the second part, Burdet and Johnson's algorithm is explained in detail. In the third section, we describe the algorithm by Burdet and Johnson for integer programming, and present some examples of the failure of the algorithm. Ways to fix these errors are given. During this chapter, the congruence symbol ($\equiv$) means congruence mod 1.

3.1 Johnson's Algorithm for the Group Problem of an MILP with a Single Constraint

3.1.1 Definitions

Ellis L. Johnson [32] proposed a new algorithm in 1973 to solve the group problem of a mixed integer programming problem with only one constraint. This algorithm uses subadditive duality as its main tool. All the theorems and proofs in this section are from [32]. Consider MILP (2.3). Suppose that its LP relaxation has been solved to optimality. Now among the basic variables, there are some with non-integer values. Choose one such variable and the corresponding constraint from its Canonical Form (see [20]). So we will have:

$$x_k + \sum_{j \in N} \overline{a}_{ij} x_j = b_0.$$

If $N \subset I$ i.e. every nonbasic variable is supposed to be integer valued, then the group relaxation for a single constraint can be written as:

$$\sum_{j \in N} f_j x_j \equiv f_0$$

where f_j is the fractional part of $\overline{a}_{ij}$ and f_0 is the fractional part of b_0. But when some x_j, $j \in N$ are not required to be integer, it changes to:

$$\sum_{j \in J_1} f_j x_j + \sum_{j \in J_2} \overline{a}_{ij} x_j \equiv f_0$$

where $J_1 = N \cap I$ and $J_2 = N \backslash I$.

Define T to be:

$$\{x \in \mathbb{Z}_+^{J_1} \times \mathbb{R}_+^{J_2} | \sum_{j \in J_1} f_j x_j + \sum_{j \in J_2} \overline{a_{ij}} x_j \equiv f_0\}.$$

The goal is to define $Conv(T)$ as a system of linear inequalities. For any $f_0 \in (0,1)$ define the family of functions $\Pi(f_0)$ on $[0,1]$, consisting of functions $\pi(u)$ having the following properties:

1. $\pi(u) \geq 0$ for all $0 \leq u \leq 1$ and $\pi(0) = \pi(1) = 0$;
2. $\pi(u) + \pi(v) \geq \pi(u+v)$ for all $0 \leq u, v \leq 1$ and $u+v$ is taken modulo 1 (subadditivity);
3. $\pi(u) + \pi(f_0 - u) = \pi(f_0)$ for all $0 \leq u \leq 1$ and $f_0 - u$ is taken modulo 1 (complementary linearity);
4. $\pi^+ = \lim_{u \to 0^+} \dfrac{\pi(u)}{u}$ and $\pi^- = \lim_{u \to 1^-} \dfrac{\pi(u)}{1-u}$ both exist and are finite.

There are two known results that we state here without proofs. The first one shows that we can represent the convex hull of the group problem of a single constrained mixed integer program, with valid inequalities derived from all these subadditive functions in the family $\Pi(f_0)$. However this representation may not be finite.

Theorem 18. [25]

$$conv(T) = \bigcap_{\pi \in \Pi(f_0)} \{x \in \mathbb{R}_+^N | \ \pi(f_0) \leq \sum_{j \in J_1} \pi(f_j) x_j + \sum_{j \in J_2^+} \pi^+ \bar{a}_{ij} x_j - \sum_{j \in J_2^-} \pi^- \bar{a}_{ij} x_j\}$$

where $J_2^+ = \{j \in J_2 : \bar{a}_{ij} > 0\}$ *and* $J_2^- = \{j \in J_2 : \bar{a}_{ij} < 0\}$.

The next theorem is actually the subadditive duality restricted to $\Pi(f_0)$. Note that optimization is on T i.e. the problem gets restricted to optimizing some objective function with non-negative coefficients subject to one constraint.

Theorem 19. [32] *For any objective function* $z = \sum \bar{c}_j x_j$ *with* $\bar{c}_j \geq 0$ *for all* $j \in N$, $x^* \in T$ *minimizes* z *over* T *provided*

$$\sum_{j \in J_N} \bar{c}_j x_j^* = \pi(f_0) \tag{3.1}$$

for some $\pi \in \Pi(f_0)$ *satisfying*

$$\pi(f_j) \leq \bar{c}_j \quad j \in J_1; \tag{3.2}$$

$$\pi^+ \overline{a_{ij}} \leq \bar{c}_j \quad j \in J_2^+; \tag{3.3}$$

$$\pi^- \overline{a_{ij}} \leq \bar{c}_j \quad j \in J_2^-. \tag{3.4}$$

In the next section, we will discuss the algorithm proposed by Johnson to solve this problem. The idea is to find a function $\pi \in \Pi(f_0)$ that satisfies the above constraints.

3.1.2 Algorithm

The idea is to generate a function $\pi \in \Pi(f_0)$ and x^* satisfying (3.1),(3.3),(3.4) and (3.4). In order to build such a function, the algorithm will start from the all zero function i.e. $\pi(u) = 0$ for $0 \leq u \leq 1$. This function will be increased until one of the constraints (3.3),(3.4) or (3.4) is violated. Violation of (3.4) or (3.4) will cause termination of the algorithm. But if a constraint of type (3.3) is violated at some point, the algorithm will fix the value of π at that point and continue to increase the function. (It is possible that two or even more constraints of type (3.3) get violated at the same time. This means that the value of π should be fixed for two or more points.) This procedure is repeated until it finds the optimal solution.

Let's see how π is defined and being increased. Function π is defined with its disjoint break points $0 = e_0 < e_1 < ... < e_L = 1$. For all other points $u \in [0, 1]$, if $e_i < u < e_{i+1}$, then linear interpolation is used from the two neighbor break points to determine $\pi(u)$. There are also two labels for break points: *increasing* and *fixed*. At the beginning, the only break points are $0, f_0$ and 1. The points 0 and 1 are fixed and f_0 is increasing.

When the algorithm increases π, it will increase the value of $\pi(e)$ for all increasing break points e. This step is called LIFT. This will continue until a point is hit i.e. one or more constraints of type (3.3) are violated. For example we hit the point $(f_j, \bar{c}_j)$ which means that further increase of π will violate $\pi(f_j) \leq \bar{c}_j$. This step is called HIT and the point $(f_j, \bar{c}_j)$ is called a hit point. Figure (3.1) shows the first step.

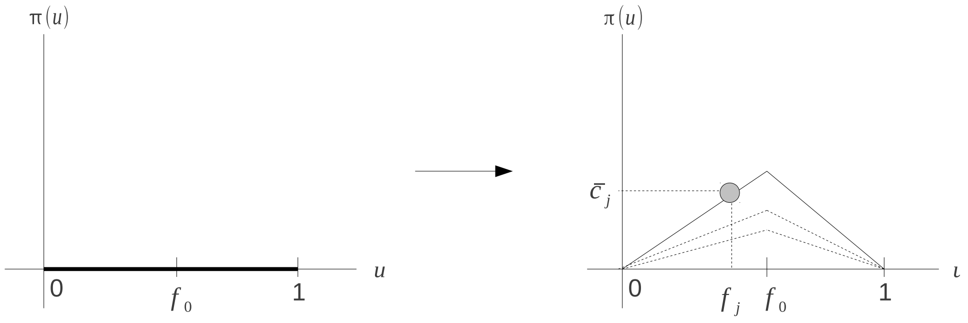


Figure 3.1: First step of the algorithm.

Then f_j will be changed to a fixed break point and $f_0 - f_j$ (modulo 1) will become an increasing break point. Again π will increase from increasing break points until it hits another pair $(f_k, \bar{c}_k)$. See Figure(3.2).

It is now obvious that the algorithm maintains the following property:

> If e_i is a fixed break point, then $f_0 - e_i (mod\ 1)$ is an increasing break point and vice versa.

The only remaining issue is to guarantee that $\pi \in \Pi(f_0)$. Clearly $\pi(u) + \pi(f_0 - u) =$

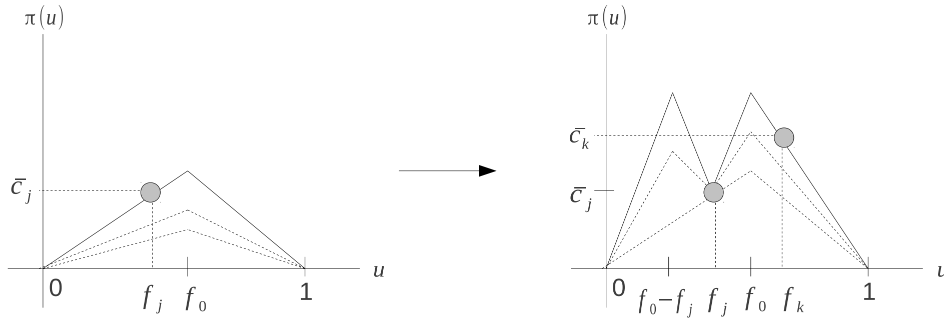


Figure 3.2: Second step of the algorithm

$\pi(f_0)$ remains satisfied since if u is a fixed (increasing) breakpoint, then $f_0 - u$ is increasing (fixed) breakpoint. So if one of the points u or $f_0 - u$ increases, the other one stays fixed and $\pi(f_0)$ will increase the same amount (f_0 is always an increasing break point). The next theorem will show that π will be subadditive if it is subadditive only on a small subset of $[0, 1]$ instead of the whole interval.

A *convex breakpoint* of a piecewise linear function is a breakpoint such that the left slope is less than the right slope. A *concave breakpoint* is defined in the same way.

Theorem 20. [32] *If π is a piecewise linear function satisfying the following:*

1. $\pi(u) \geq 0$, $0 \leq u \leq 1$ *and* $\pi(0) = \pi(1) = 0$*;*

2. $\pi(u) + \pi(v) \geq \pi(u + v)$, *for convex breakpoints* u, v*;*

3. $\pi(u) + \pi(f_0 - u) = \pi(f_0)$, $0 \leq u \leq 1$*;*

then π is subadditive i.e. $\pi(u) + \pi(v) \geq \pi(u + v)$, *for all* $0 \leq u, v \leq 1$.

Points 0 and 1 are considered as convex break points. In the algorithm, fixed break points become convex and increasing break points become concave break points.

It is enough to check if π is subadditive at convex break points. This is done by using a set called *Candidate Set*. This set contains all the breakpoints at which subadditivity may become violated later.

All points $(f_j, \bar{c}_j)$ with $j \in J_1$ are called *original*. All other points are called *generated*. At the beginning the candidate set contains only original points.

All non-negative integer combinations of $f_1, ..., f_N$ could become break points. In case of pure integer programs, since any feasible solution to the group problem is such a combination, break points are simply candidates for the the optimal solution. The algorithm systematically goes through these break points to search for one that is feasible and of minimum cost. For the mixed integer case, the algorithm may also terminate because of the violation of (3.4) and/or (3.4).

The function π will increase until we hit a point in the candidate set. Suppose that we hit two original points f_{j_1} and f_{j_2} at the same time. This means that two constraints of type (3.3) will be violated in case of further increase, namely $\pi(f_{j_1}) \leq \bar{c}_{j_1}$ and $\pi(f_{j_2}) \leq \bar{c}_{j_2}$. So these points become fixed points. Now the only points in which subadditivity may get violated later, are $2f_{j_1}$,$2f_{j_2}$ and $f_{j_1} + f_{j_2}$ i.e. the only possibility is that one of the following constraints get violated:

$$\begin{aligned}
&\pi(2f_{j_1}) \leq \pi(f_{j_1}) + \pi(f_{j_1}) = 2\bar{c}_{j_1} \\
&\pi(2f_{j_2}) \leq \pi(f_{j_2}) + \pi(f_{j_2}) = 2\bar{c}_{j_2} \\
&\pi(f_{j_1} + f_{j_2}) \leq \pi(f_{j_1}) + \pi(f_{j_2}) = \bar{c}_{j_1} + \bar{c}_{j_2}.
\end{aligned}$$

This is because subadditivity of π on the set of convex break points will imply subadditivity of π on $[0, 1]$. So we enter these new points $(2f_{j_1}, 2\bar{c}_{j_1}), (2f_{j_2}, 2\bar{c}_{j_2})$ and $(f_{j_1} + f_{j_2}, \bar{c}_{j_1} + \bar{c}_{j_2})$ to the candidate set. This step is called GENERATE. Since any of these new points are non-negative linear combinations of the original points, checking subadditivity becomes equivalent to finding the first hit point in the candidate set.

So the candidate set does two things: first it contains all the original points and hence it will avoid any violation of constraints of type (3.3) and second, it contains all critical points at which the function π may lose its subadditivity later.

The algorithm will terminate if any increasing break point is hit. If hit g is an increasing breakpoint then its complimentary point $f_0 - g$ is fixed and $(f_0, \pi(f_0))$ can be generated as the sum of two generated hit points:

$$(f_0, \pi(f_0)) = (g, \pi(g)) + (f_0 - g, \pi(f_0 - g)).$$

So if for example

$$(f_0, \pi(f_0)) = (\sum_{j \in J_1} \lambda_j f_j, \sum_{j \in J_1} \lambda_j c_j)$$

for (f_j, c_j) original, then $x_j = \lambda_j$ for $j \in J_1$ will be a solution satisfying (3.1).

The algorithm also terminates when $\pi^+ = c^+$ or $\pi^- = c^-$. If $\pi^+ = c^+$, then by property (3) of $\Pi(f_0)$, the slope at f_0 from below should be π^+ as well. So the first breakpoint below f_0 must be a fixed one since otherwise π^+ would not be increasing. This fixed break point $(e, \pi(e))$ can be reached using non-negative integer combination of original points (f_j, c_j). From this breakpoint c^+ can be used to reach $(f_0, \pi(f_0))$ where

$$\pi(f_0) = c^+(f_0 - e) + \pi(e).$$

So a solution $x^* \in T$ can be generated satisfying (3.1).

For example if

$$(e, \pi(e)) = (\sum_{j \in J_1} \lambda_j f_j, \sum_{j \in J_1} \lambda_j c_j)$$

for (f_j, c_j) original, let $x_j = \lambda_j$ for $j \in J_1$ and then we can adjust one of the continuous variables x_j in J_2^+ where $\pi^+ = c^+$ occurs to take the value $\dfrac{c^+(u_0 - e)}{c_j}$. The case $\pi^- = c^-$ is treated in a similar way.

Now we can write the steps of the algorithm. B denotes the set of all break points.

We suppose that elements in B, are ordered in the following form:

$$0 = e_0 < e_1 < e_2 < ... < e_{L-1} < e_L = 1.$$

The set F will keep track of fixed break points. If $y \in F$, then Gy is a fixed break point. So the set of fixed break points will be $\{Gy|y \in F\}$ and the set of increasing break points will be $B \backslash \{Gy|y \in F\}$. H is the set of hit points. C will keep track of the points in candidate set i.e. if y is in C, then Gy is in the candidate set. So the candidate set can be written as $\{Gy|y \in C\}$.

Let

$$c^+ = \min\{\frac{\bar{c}_j}{a_{ij}} : \ j \in J_2^+\} \text{ and } c^- = \min\{\frac{\bar{c}_j}{-a_{ij}} : \ j \in J_2^-\}.$$

Then (3.4) and (3.4) become $\pi^+ \leq c^+$ and $\pi^- \leq c^-$.

Since π always has the same value at 0 and 1 and computation is modulo 1, we consider 0 and 1 as one point and use 0 for both. δ^i will denote the unit vector with 1 at the i-th entry and zero at other entries.

Theorem 21. [32] *LIFT-HIT-GENERATE algorithm will find* $\pi \in \Pi(f_0)$ *and* $x^* \in T$ *with*

$$\sum_{j \in J_N} c_j x_j = \pi(f_0).$$

This theorem is proved in [32]. When function π is increasing, we get to a point at which further increase will violate subadditivity of π. The main idea of the proof is to show that the algorithm will stop increasing π here because some point in the candidate set is already hit.

```
**Data**: Group problem of an MILP with only one constraint
**Result**: Optimal solution to the group problem
    **Initialization:** Let
        $$B = \{0, f_0\}, H = \emptyset, F = \{[0,0,...,0]^T\}, C = \{\delta^1, ..., \delta^N\}, G = [f_1, ..., f_N];$$
        $$\pi(0) = \pi(f_0) = 0.$$
    **LIFT:** For each $y \in C$ with $\pi(Gy) < \bar{c}y$, evaluate $\delta_y$ in the following way:
    Let $e_i \leq Gy \leq e_{i+1}$ where $e_i$ and $e_{i+1}$ are breakpoints in $B$. Let
    $\theta_y = \alpha\theta_l + (1-\alpha)\theta_u$ where $\theta_l(\theta_u)$ is 0 or 1 depending on whether $e_i(e_{i+1})$
    is fixed or increasing respectively and $\alpha = \frac{e_{i+1} - Gy}{e_{i+1} - e_i}$. Let
    $\pi_y = \alpha\pi(e_i) + (1-\alpha)\pi(e_{i+1})$ and $\delta_y = \frac{\bar{c}y - \pi_y}{\theta_y}$. If $\bar{c}y - \pi_y = 0$ then
    consider the fraction to be 0 even if $\theta_y = 0$. If $\bar{c}y - \pi_y > 0$ and $\theta_y = 0$,
    then the ratio is $+\infty$.
    Now define $\delta = \min\{\delta_y : \pi(Gy) < \bar{c}y, y \in C\}, \delta^+ = c^+e_1 - \pi(e_1)$ and
    $\delta^- = c^-(1 - e_{L-1}) - \pi(e_{L-1})$.
    **if** $\delta^+ < \delta$ *or* $\delta^- < \delta$ **then**
        change $\pi(e)$ to $\pi(e) + \min\{\delta^+, \delta^-\}$ for all points in $B \backslash \{Gy | y \in F\}$ and
        terminate.
    **end**
    **else**
        Increase $\pi$ for all increasing break points by $\delta$:
            $$\pi(e) = \pi(e) + \delta, \text{ for all } e \in B \backslash \{Gy | y \in F\}.$$
    **end**
    **HIT:** Update $H, C$ and $F$:
        $$H = \{y \in C | \pi(Gy) = cy\}, C \leftarrow C \backslash H, F \leftarrow F \cup H$$
    **if** $H \cap B \backslash \{Gy | y \in F\} \neq \emptyset$ **then**
        Terminate. Optimal solution found.
    **end**
    **GENERATE:** For each $y \in H$ let $B = B \cup \{Gy, f_0 - Gy\}$.
    **if** $y \neq \delta^i$ *for any* $i$ **then**
        let $F' = \{y + f | f \in F\}$
    **end**
    **else**
        $F' = \{y + \delta^i | \delta^i \in F\}$
    **end**
    Set $C \leftarrow C \cup F'$ and go to LIFT.
```

Algorithm 3.1: LIFT-HIT-GENERATE Algorithm

3.2 Burdet and Johnson's Algorithm for the Group Problem of Integer Programming

We saw the lifting for a single constraint group problem in the previous section. Since in the case of one constraint, the corresponding subadditive function is a one-variable function, dealing with this case is easier than the multiple constraint case. In fact if we want to use a similar algorithm for the group problem, we have to lift a multi-variable function while trying to keep it subadditive and satisfy other duality constraints. This task seems nontrivial.

The idea for making the latter case easier was to define a subadditive function with a parameter as a degree of freedom. These parameters will be used as a tool for increasing (lifting) the subadditive function while maintaining some properties and keep some constraints satisfied. Burdet and Johnson [15] proposed this algorithm in 1974 to solve the group problem of an integer program however the case for a mixed integer program was left open. The basic idea of the algorithm is very similar to the previous algorithm in 3.1 by Johnson. However, the problem is transferred to a hypercube from the [0,1] interval in one dimension. The lifting and keeping the dual function subadditive are done using *Candidate Set* and a function π which will be introduced in next section. All the theorems and proofs in this section are from [15].

3.2.1 Definitions

Suppose that we have solved the LP relaxation of integer program IP (2.1) to optimality. Let D denote the index set of basic variables that do not satisfy integrality constraints. Without loss of generality, assume that $D = \{1, ..., d\}$. Define $f^0 \in \mathbb{R}^D$ by

$$f_i^0 = \bar{b}_i - \lfloor \bar{b}_i \rfloor \geq 0 \qquad i \in D \subseteq B$$

and vectors $f^j \in \mathbb{R}^D$ for $j \in N$ by

$$f_i^j = \begin{cases} \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor & \text{if } 0 \leq \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor \leq f_i^0, \\ \bar{a}_{ij} - \lceil \bar{a}_{ij} \rceil & \text{if } f_i^0 \leq \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor \leq 1. \end{cases}$$

The group problem of the integer program IP (2.1) can be written:

$$\begin{aligned} \min \quad & (z - z_B) = \sum_{j \in N} \bar{c}_j x_j \\ \text{s.t.} \quad & \sum_{j \in N} f_i^j x_j \equiv f_i^0 && i \in D \\ & x_j \geq 0, \text{integer} && j \in N. \end{aligned} \tag{3.5}$$

Subadditive functions will be defined on the unit hypercube $U \subseteq \mathbb{R}^D$, which is defined by

$$U = \{u : \ f_i^0 - 1 \leq u_i \leq f_i^0 \text{ for all } i \in D\}$$

and contains the vectors f^0 and f^j for $j \in N$. For the unit hypercube, we know that:

1. $0 \in U \subset \mathbb{R}^D$,

2. for any vertex V of U we have

$$V_i = f_i^0 \text{ or } f_i^0 - 1 \text{ for all } i \in D.$$

Theorem 22. [15] *Let π be a function defined on the d-dimensional unit hypercube U satisfying*

1. $\pi(u) \geq 0$ *for all* $u \in U$;

2. $\pi(0) = 0$;

3. $\pi(u) + \pi(v) \geq \pi(u + v)$;

for all u and v in U where $u + v$ is taken modulo 1 so that $u + v \in U$. Then the inequality

$$\sum_{j \in N} \pi(f^j) x_j \geq \pi(f^0) \tag{3.6}$$

is valid for integer program IP (2.1).

Suppose that we have

$$\pi(f^j) \leq \bar{c}_j \text{ for all } j \in N.$$

Using Theorem 22 we will get:

$$\pi(f^0) \leq \sum_{j \in N} \pi(f^j) x_j \leq \sum_{j \in N} \bar{c}_j x_j.$$

This means that $\pi(f^0)$ provides a lower bound on the optimal value for the group problem. The goal is to find such a π so that $\pi(f^0)$ is as large as possible. In fact, there is always a π with $\pi(f^0)$ equal to the optimal value of the group problem.

Let G be a matrix with columns f^j for $j \in N$. The basic tool in this algorithm is the function π:

$$\pi_{\alpha,E}(u) = \min_{y \in E} \{\bar{c}y + D_\alpha(u - Gy)\} \text{ for all } u \in U$$

where $D_\alpha(u)$ is a function defined on U with parameter $\alpha \in \mathbb{R}^D$ satisfying three properties:

1. D_α is subadditive;

2. $D_\alpha(0) = 0$;

3. $D_\alpha(u) \geq 0$ for all $u \in U$;

4. for each $y \in E$ such that $Gy \neq f^0$, $D_\alpha(f^0 - Gy) \to \infty$ as $\alpha \to \infty$.

5. for each $u \in U$, $D_\alpha(u) \leq D_\beta(u)$ for all $0 \leq \alpha \leq \beta$.

E is a finite subinclusive[1] set in $\mathbb{Z}_+^N$ called the *Generator Set* which contains only zero vector at the beginning, but will be expanded later by the algorithm.

For instance D_α can be defined as

$$D_\alpha(u) = \begin{cases} \alpha & \text{if } f^0 - 1 \leq u < 0 \text{ or } 0 < u \leq f^0 \\ 0 & \text{if } u = 0. \end{cases} \tag{3.7}$$

We will see later that this is not the best function that may be defined (actually it is the worst).

The point in defining $\pi_{\alpha,E}$ is that we do not need to check the subadditivity of $\pi_{\alpha,E}$ for all points. Indeed if $\pi_{\alpha,E}$ is subadditive on the set $\{Gy|y \in E\}$, then it is subadditive everywhere on U.

For fixed α and E, we can use π instead of $\pi_{\alpha,E}$ for simplicity.

Lemma 23. [15] *For fixed α and E, if $u, v \in U$ are such that $\pi(u)+\pi(v) < \pi(u+v)$, then*

$$\pi(G\bar{y}) + \pi(v) < \pi(G\bar{y} + v)$$

where $\bar{y} \in E$ is such that $\pi(u) = \bar{c}\bar{y} + D(u - G\bar{y})$.

Theorem 24. [15] *π is subadditive on U if and only if it is subadditive on the set $\{Gy|y \in E\}$.*

[1]A set E of integer vectors is called subinclusive if $x \in E$ and $0 \leq y \leq x$ imply that $y \in E$.

3.2.2 Algorithm

Let M denote the set of non-negative integer N-vectors. When we write Gy with $y \in M$, we mean $Gy \pmod 1$.

Suppose that π is subadditive on $\{Gy|y \in E\}$ (therefore, subadditive on U by Theorem 24) and f^0 cannot be written as Gy for $y \in E$. Also suppose that

$$\pi(f^j) \leq \bar{c}^j \text{ for all } j \in N \text{ and for each } y \in E, \pi(Gy) = \bar{c}y. \tag{3.8}$$

Then $\pi(f^0) \to \infty$ as $\alpha \to \infty$. In addition, (3.8) continues to hold if α is increased by property 4 of D_α. Hence, using Theorem 22, to have $\pi(f^0)$ to give as good a lower bound as possible with the given E, we increase α as much as possible without violating subadditivity and $\pi(f^j) \leq \bar{c}_j$ for $j \in N$.

Observe that having π subadditive on the set $\{Gy|y \in E\}$ and $\pi(f^j) \leq \bar{c}_j$ for all $j \in N$ implies that

$$\pi(Gy) \leq \bar{c}y \text{ for all } y \in M \backslash E. \tag{3.9}$$

So, α cannot be further increased as soon as a constraint in (3.9) becomes active. When that occurs, move all such y to E. We will show that after moving such points to E, π still remains subadditive on E and (3.8) continues to hold. Thus, one can then further increase α again.

The above process continues until f^0 is moved to E. At the moment this happens, we have a $y \in M$ such that $f^0 = Gy$ and $\pi(f) = \bar{c}y$. So y is an optimal solution to the group problem. The process can be made finite since $\{Gy|y \in M\}$ is finite as G is rational and there is no need to consider $z \in M \backslash E$ such that $Gy \equiv Gz$.

Checking (3.9) seems like an arduous task. Instead one simply check (3.9) for points in a smaller set called the *Candidate Set* of E. The candidate set is defined by

$$C = \{y \in \mathbb{Z}_+^N | y \notin E,\ S(y) \backslash \{y\} \subseteq E\}$$

where $S(y) = \{x \in \mathbb{Z}_+^N | x \leq y\}$.

Steps of the algorithm are stated below:

```
Data: Group problem of an IP
Result: Optimal solution to the group problem
    INITIALIZATION: Set $E = \{0\}$ and $C = \{\delta^j : j \in N\}$.
    while $f^0 \neq Gy$ for some hit point $y$ do
        LIFT: Increase $\pi(f^0)$ by increasing $\alpha$ until one of the following
        constraints become active:
                $\pi(Gy) \leq \bar{c}y$ for all $y \in C$.        (3.10)
        Note that $\alpha$ could be zero. Every $y \in C$ with $\pi(Gy) = \bar{c}y$ is a new
        hit point.
        HIT: Move all hit points $y$ from the previous step from $C$ to $E$. If
        $f^0 = Gy$ for some hit point $y$, terminate.
        GENERATE: For new hit points $y$, adjoin $y + \delta^j$ to $C$ for every
        $\delta^j \in E$. Go to LIFT.
    end
```

Algorithm 3.2: LIFT-HIT-GENERATE Algorithm for the general group problem of IP

Theorem 25. [15] *Suppose that (3.5) has a finite optimal solution. LIFT-HIT-GENERATE algorithm by Burdet and Johnson* [15] *for solving the group problem of an integer program will solve this problem in a finite number of iterations.*

Choosing D_α

For each choice of $\alpha_1, ..., \alpha_d$, one obtains a function D (of course, one should choose $\alpha_1, ..., \alpha_d$ such that all 5 properties hold. In particular, one wouldn't choose 0 for all of them even though Burdet and Johnson allow this). In other words, each set of $\alpha_1, ..., \alpha_d$ defines a family of D_α. Burdet and Johnson define D_α in their paper [15] in the following way:

Definition 26. *Given a set of $(d+1)$ parameters $\alpha_0, \alpha_1, ..., \alpha_d$ satisfying*

$$\alpha_i \geq 0 \textit{ for all } i = 0, 1, ..., d \textit{ and} \tag{3.11}$$

$$\sum_{i=1}^{d} \alpha_i f_i^0 \bar{f}_i^0 = \alpha_0 \textit{ with } \bar{f}_i^0 = 1 - f_i^0 \tag{3.12}$$

define the diamond gauge function D by

$$D(u) = \sum_{i=1}^{d} \alpha_i \sigma_i(u_i) u_i$$

with

$$\sigma_i(u_i) = \begin{cases} \bar{f}_i^0 & \textit{if } 0 \leq u_i, \\ -f_i^0 & \textit{if } u_i < 0. \end{cases}$$

Properties of $D(u)$

Proofs for these properties are give in [15].

Property 1. For any vertex V of U, we have

$$D(V) = \sum_{i=1}^{d} \alpha_i f_i^0 \bar{f}_i^0 = \alpha_0.$$

Since f^0 is a vertex of U, $D(f^0) = \alpha_0$. Also $D(0) = 0$ and $0 \leq D(u) \leq \alpha_0$ for all $u \in U$.

Property 2. Consider one of the 2^d truncated orthants Q

$$Q = \begin{cases} 0 \le u_i \le f_i^0 & \text{for all } i \in \Delta^+ \\ -\bar{f}_i^0 \le u_i \le 0 & \text{for all } i \in \Delta^- \end{cases} \subset U$$

where Δ^+ and Δ^- are given disjoint index sets such that $\Delta^+ \cup \Delta^- = N$. For any $u \in Q$ we have

$$D(u) = \sum_{i \in \Delta^+} \alpha_i \bar{f}_i^0 u_i - \sum_{i \in \Delta^-} \alpha_i f_i^0 u_i = \sum_{i=1}^{n} \delta_i u_i$$

with

$$\delta_i = \begin{cases} \alpha_i \bar{f}_i^0 & \text{for } i \in \Delta^+, \\ -\alpha_i f_i^0 & \text{for } i \in \Delta^-. \end{cases}$$

Property 3. The Diamond function D is linear in Q. (This is obvious from property 2.)

Property 4. The level set $lev_\alpha D = \{u : \; D(u) \le \alpha\}$ is convex.

Definition 27. *A Gauge is a convex function which is positively homogeneous*[2].

Property 5. D is a gauge on U.

Property 6. D is subadditive on U.

3.3 Burdet and Johnson's Algorithm for Integer Programming

In this section we describe Burdet and Johnson's algorithm to solve integer programming problems [16] which extends the ideas in [15] and [32] (Sections 3.2 and 3.1 respectively).

[2]Function f is called positively homogeneous if for any $\lambda \ge 0$, $f(\lambda x) \le \lambda f(x)$ for all x in domain of f.

Obviously the solution to the group relaxation might not be the same for integer programming. The reason is that we have relaxed the non-negativity constraints on the basic variables. Now if we add these constraints back into the problem in terms of non-basic variables, the problem will be equivalent to IP. It will be in the following form:

$$\begin{aligned} \min \quad & \sum_{j \in N} \bar{c}_j x_j \\ \text{s.t.} \quad & Gx \equiv g_0 \\ & Hx \geq h_0 \\ & x_j \geq 0, \text{ integer for all } j \in N. \end{aligned} \tag{3.13}$$

For $x \in \mathbb{Z}_+^N$ Define

$$S_I(x) = \{y \in \mathbb{Z}_+^N |\ y \geq x \text{ and } Gy \equiv g_0,\ Hy \geq h_0\},$$
$$S_L(x) = \{y \in \mathbb{Z}_+^N |\ y \geq x \text{ and }\ Hy \geq h_0\},$$

and let $X_I(x) = \{x \in \mathbb{Z}_+^N | S_I(x) \neq \emptyset\}$. If π is a subadditive function on X_I i.e.

$$\pi(x) + \pi(y) \geq \pi(x+y) \text{ for all } x, y \in X_I \text{ and } x + y \in X_I,$$

then the inequality

$$\sum_{j \in N} \pi_j x_j \geq \pi_0 \tag{3.14}$$

is valid for (3.13) where

$$\pi_j = \pi(\delta^j)$$

$$\pi_0 \leq \min\{\pi(x) | x \in S_I(0)\}.$$

Based on the idea from previous algorithms in [15] and [32], $\pi(x)$ is defined from a subadditive function Δ on $\mathbb{R}_+^N$ with a finite generator set E by

$$\pi(x) = \min_{y \in I(x)} \{\bar{c}y + \Delta(x-y)\}$$

where $I(x) = E \cap S(x)$ and $S(x) = \{y \in \mathbb{Z}_+^N | y \leq x\}$. If $\pi(y_1 + y_2) \leq \bar{c}y_1 + \bar{c}y_2$, for all $y_1, y_2 \in E$ and $y_1 + y_2 \in C$, where C is the candidate set for E, then π is subadditive. In other words, if for every $x \in C$, $\Delta(x) \leq \bar{c}x$, then

$$\sum_{j \in N} \pi_j x_j \geq \pi_0$$

is a valid inequality for IP (2.1) when $\pi_0 \leq \min\{\pi(x) | x \in S_I(0)\}$ holds. This π_0 will be a lower bound on the optimum objective value.

Let σ be defined as

$$\sigma(q, \sigma^+, \sigma^-) = \begin{cases} \sigma^+ & q > 0 \\ 0 & q = 0 \\ -\sigma^- & q < 0. \end{cases}$$

Assume that G and H have m_1 and m_2 rows respectively. For $2m_1 + 2m_2$ real numbers

$$\gamma_1^+, ..., \gamma_{m_1}^+, \gamma_1^-, ..., \gamma_{m_1}^-, \alpha_1^+, ..., \alpha_{m_2}^+, \alpha_1^-, ..., \alpha_{m_2}^-$$

Burdet and Johnson defined the generalized diamond gauge[3] function D from $\mathbb{R}^n$ to $\mathbb{R}$ as

$$D(x) = \max_{\alpha, \gamma} \{\gamma G x + \alpha H x\}$$

[3]A function which is non-negative, convex and positively homogeneous is called a gauge [40]. A generalized gauge doesn't have the non-negativity constraint. Also D has been named after its level set which is a diamond polyhedron (see [14]) centered at the origin.

where the minimum is taken over $2m_1 + 2m_2$ possible values: $\gamma_i = \gamma_i^+$ or γ_i^- and $\alpha_i = \alpha_i^+$ or α_i^-. This function was first defined in [15] and used to solve the group problem of integer programs. Burdet and Johnson defined the gauge function Δ as any of the following functions:

$$\begin{aligned}
\Delta_0(x) &= \min_z\{D(z)|z \text{ satisfies } z \geq 0, z \text{ integer }, Gz \equiv Gx, Hz \geq Hx\}\\
\Delta_1(x) &= \min_z\{\gamma(u)\cdot u|u = Gz \text{ for } z \geq 0, z \text{ integer }, Gz \equiv Gx\}\\
&\quad + \min_z\{\alpha(\xi)\cdot\xi|\xi = Hz \text{ for } z \geq 0, z \text{ integer, and } Hz \geq Hx\}\\
\Delta_2(x) &= \min_u\{\gamma(u)\cdot u|u \equiv Gx\} + \min_z\{\alpha(\xi)\cdot\xi|\xi = Hz \text{ for } z \geq 0 \text{ and } Hz \geq Hx\}\\
\Delta_3(x) &= \min_u\{\gamma(u)\cdot u|u \equiv Gx\} + \min_\xi\{\alpha(\xi)\cdot\xi|\xi \geq Hx\}
\end{aligned}$$

where $\gamma_i(u) = \sigma(u_i, \gamma_i^+, \gamma_i^-)$ and $\alpha_i(u) = \sigma(\xi_i, \alpha_i^+, \alpha_i^-)$. Note that

$$\Delta_0(x) \geq \Delta_1(x) \geq \Delta_2(x) \geq \Delta_3(x).$$

Among these functions, Δ_0 gives the best lower bound however it is difficult to be evaluated at one point. In the other end Δ_3 is much easier to use computationally, but the bound it generates is not as good as the one by Δ_0.

Burdet and Johnson's algorithm may be written in the form of Algorithm 3.

Theorem 28. [16] *At termination, Algorithm 3 returns an optimal solution for IP (2.1).*

3.3.1 Geometric Interpretation of the Algorithm

From a geometric point of view, Algorithm 3 *lifts* a function (namely π) until it *hits* a point where extra lifting will cause the violation of subadditivity of π. In other words the algorithm is trying to increase the value of a dual feasible subadditive function

```
**Data**: Pure Integer Program
**Result**: Optimal solution to the IP
Initialization: Assume that $\min_{x \in S}\{\pi(x)\} > 0$ for some $S \supseteq S_I(0)$. Scale $\pi$
such that this minimum is equal to one. Let
$\alpha_0 = \min_{j=1,\ldots,n}\{\frac{\bar{c}_j}{\Delta(\delta^j)}|\Delta(\delta^j) > 0\}$. The initial subadditive function is
$\pi(x) = \alpha_0\Delta(x)$ and the initial bound is $\alpha_0$, $E = \{0\}$ and
$C = \{\delta^i : i = 1, ..., n\}$;
while $cx \neq \pi_0$ for some $x$ in $C$ and feasible to IP (2.1) do
    1. Calculate $x^* = \operatorname{argmin}_{j=1,\ldots,n}\{\frac{\bar{c}x}{\Delta(x)}|\Delta(x) > 0, x \in C\}$.
    2. Let $\alpha_0 = \frac{\bar{c}x^*}{\Delta(x^*)}$.
    3. Move $x^*$ from $C$ to $E$ and update $C$: adjoin to $C$ all the
       points $x > x^*$ and$x \in \mathbb{Z}^n_+$ with the property that for every
       $y < x$ and $y \in \mathbb{Z}^n_+$, we have $y \in E$.
    4. Evaluate $\pi_0 = \min_{y \in E \cap S(x)}\{\bar{c}y + \min_{x \in S}\alpha_0\Delta(x - y)\}$.
    5. (optional) Solve the following LP and update $\pi_0$ accordingly:
       $\max \ \pi_0$
       s.t. $\pi_0 \leq \bar{c}y + \Delta(x - y) \quad y \in E, x \in S$
            $\Delta(x) \leq \bar{c}x \quad x \in C.$
end
```

Algorithm 3.3: Burdet and Johnson's algorithm to solve IP

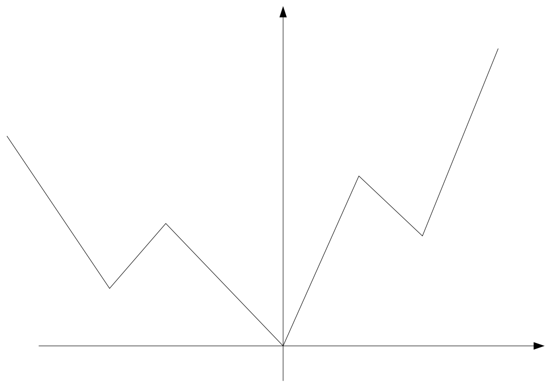
until it reaches strong duality and zero gap. The generator and candidate sets help us keep track of such points. The lifting is done by increasing the norming factor α_0 as much as possible. The parameter optimization adjusts the slopes and directions of the function π, so that the best lower bound results from lifting. See Figure (3.3).

The algorithm is a combination of enumeration and cutting plane methods. Iterating without parameter adjustment will be pure enumeration through the set of feasible points. However parameter adjustment will be equivalent to using the deepest cutting plane of the form (3.14). In the worst case, the algorithm may have to enumerate all feasible points until it finds an optimal solution hence the algorithm can have exponential running time. However, parameter optimization can reduce the running time significantly.

3.3.2 Deficiencies and Fixes

Burdet and Johnson claimed that any subadditive function could be used in this algorithm for Δ (they recommended the use of Δ_2 for practical reasons but generally any subadditive function was allowed in their paper). This is not correct since if we use $\Delta = 0$, the algorithm will not work since π_0 will never increase from zero and the parameter adjustment would be impossible.

Second, they claimed that one could use the algorithm without using the parameter optimization. It is stated that enumeration must be done to some extent in order to proceed (which is correct), but completion of parameter adjustment is not necessary. In the following example, we consider two cases. One without using parameter adjustment and one with parameter adjustment. In both cases, the algorithm fails to solve the IP. Δ_0 is used as the subadditive function however for this particular example all functions $\Delta_0, ..., \Delta_3$ will be equivalent and will give the same result.



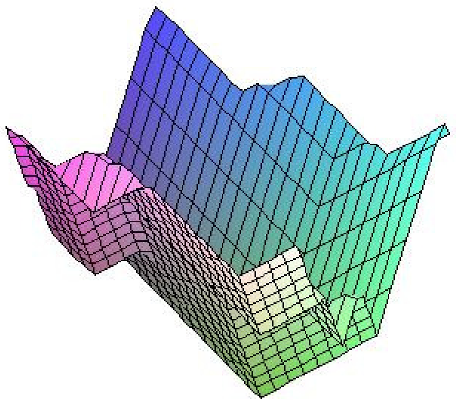


Figure 3.3: Lifting π in 2 and 3 dimensions

Example 29. *Consider the following pure integer program:*

$$\begin{array}{lllll} min & x_1 + \ 3x_2 & & & \\ s.t. & & -\frac{1}{2}x_2 + x_3 & & = \frac{1}{2} \\ & -x_1 + x_2 & & + x_4 & = -1 \\ & x_1, x_2, x_3, x_4 \geq 0, \ \textit{integer}. & & & \end{array}$$

The optimal solution is $x^* = (2, 1, 1, 0)$ with $z^* = 5$. To see this, note that $x_2 \geq 1$ is a valid Gomory cut. Adding this cut to the LP relaxation will give the optimal solution.

Using the optimal LPR basis {3,4}, the group problem becomes:

$$\begin{aligned} min \quad & x_1 + \ 3x_2 \\ s.t. \quad & \tfrac{1}{2}x_2 \equiv \tfrac{1}{2} \\ & x_1, x_2 \geq 0 \ \textit{integer.} \end{aligned}$$

Equivalent form for the IP is

$$\begin{aligned} min \quad & x_1 + \ 3x_2 \\ s.t. \quad & \tfrac{1}{2}x_2 \equiv \tfrac{1}{2} \\ & x_1 - x_2 \geq 1 \\ & x_1, x_2 \geq 0 \ \textit{integer.} \end{aligned}$$

Note that one constraint ($\frac{1}{2}x_2 \geq -\frac{1}{2}$) *has been removed since it is redundant. With our previous notation, we have:*

$$G = \begin{bmatrix} 0 & \frac{1}{2} \end{bmatrix}, H = \begin{bmatrix} 1 & -1 \end{bmatrix}, g_0 = \tfrac{1}{2} \ \textit{and} \ h_0 = 1.$$

Choose Δ_0 *with* $S = S_L(0)$.

Scale γ *and* α *so that* $\min_{x \in S}\{\pi(x)\} = 1$. *Since* $E = \{(0,0)\}$, *this becomes equivalent to*

$$1 = \min_z\{D(z)|z \ \textit{satisfies} \ z \geq 0, z \ \textit{integer} \ , Gz \equiv Gx, Hz \geq Hx\}.$$

So we have:

$$\begin{aligned}1 &= \min_{x\in S}\min_{z}\{\max_{\gamma,\alpha}\{\gamma Gz+\alpha Az\}|z\geq 0, z \textit{ integer }, Gz\equiv Gx, Hz\geq Hx\}\\ &= \min_{x\in S}\min_{z}\{\max\left\{\begin{array}{c}\frac{1}{2}\gamma^+ z_2\\ -\frac{1}{2}\gamma^- z_2\end{array}\right\}+\max\left\{\begin{array}{c}\alpha^+ z_1-\alpha^+ z_2\\ -\alpha^- z_1+\alpha^- z_2\end{array}\right\}:\\ &\quad z\geq 0, z \textit{ integer }, \tfrac{1}{2}z_2\equiv\tfrac{1}{2}x_2, z_1-z_2\geq x_1-x_2\}\end{aligned}$$

Note that $x=(1,0)$ will minimize Δ_0 on S since as shown in Figure (3.4), any point

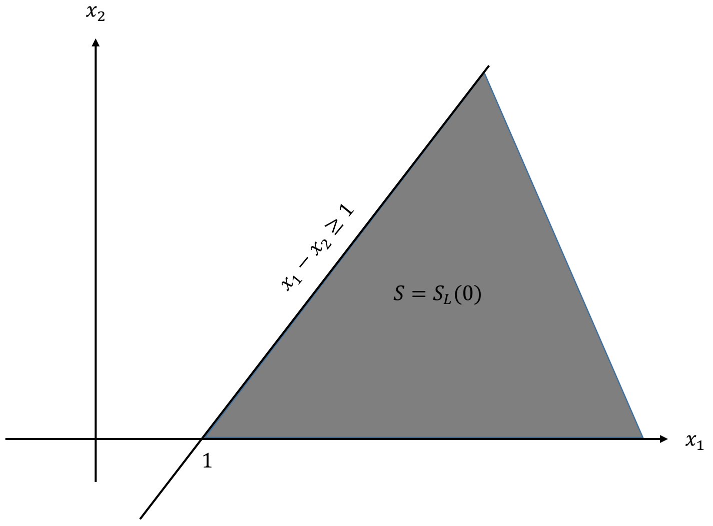


Figure 3.4: The set $S = S_L(0)$

x that we choose from S, will have $x_1 - x_2 \geq 1$. So we have:

$$1=\min_{z}\{\max\left\{\begin{array}{c}\frac{1}{2}\gamma^+ z_2\\ -\frac{1}{2}\gamma^- z_2\end{array}\right\} + \max\left\{\begin{array}{c}\alpha^+ z_1-\alpha^+ z_2\\ -\alpha^- z_1+\alpha^- z_2\end{array}\right\}: \\ z\geq 0, integer, \tfrac{1}{2}z_2\equiv 0, z_1-z_2\geq 1\}.$$

z_2 can take values $2k$ for $k=0,1,2,\ldots$ and $z_1=1+2k$. Obviously $k=0$, will give

the optimal value and this gives us $\alpha^+ = \alpha^- = 1$. γ could be any value. We choose $\gamma^+ = \gamma^- = 1$.

Proposition 30. *For $k \geq 0$, $\alpha^+ = \alpha^- = \alpha$ and $\gamma^+ = \gamma^- = \gamma$ for some α and γ, $\Delta_0(k, 0) = k\alpha$.*

Now $E = \{(0,0)\}$ and $C = \{(1,0),(0,1)\}$. We have $\Delta_0(1,0) = 1$ and $\Delta_0(0,1) = \frac{1}{2}$:

$$\begin{aligned}
\Delta_0(0,1) &= \min_z\{\max\left\{\begin{array}{c}\frac{1}{2}\gamma^+ z_2\\ -\frac{1}{2}\gamma^- z_2\end{array}\right\} + \max\left\{\begin{array}{c}\alpha^+ z_1 - \alpha^+ z_2\\ -\alpha^- z_1 + \alpha^- z_2\end{array}\right\}: \\
&\quad z \geq 0, \textit{integer}, \tfrac{1}{2}z_2 \equiv \tfrac{1}{2}, z_1 - z_2 \geq -1\} \\
&= \max\left\{\begin{array}{c}\frac{1}{2}\gamma^+\\ -\frac{1}{2}\gamma^-\end{array}\right\} + 0 \\
&= \tfrac{1}{2}
\end{aligned}$$

so

$$\alpha_0 = \min_{j=1,\dots,n}\{\frac{\bar{c}_j}{\Delta_0(\delta^j)}|\Delta_0(\delta^j) > 0\} = \min\{\frac{1}{\Delta_0(1,0)}, \frac{3}{\Delta_0(0,1)}\} = \min\{\tfrac{1}{1}, \frac{3}{\frac{1}{2}}\} = 1,$$

so initial $\pi(x)$ is $\Delta_0(x)$.

$$x^* = \operatorname*{argmin}_{j=1,\dots,n}\{\frac{\bar{c}x}{\Delta_0(x)}|\Delta_0(x) > 0, x \in C\} = \operatorname{argmin}\{\frac{1}{\Delta_0(1,0)}, \frac{3}{\Delta_0(0,1)}\} = (1,0).$$

Now the new sets are $E = \{(0,0),(1,0)\}$ and $C = \{(2,0),(0,1)\}$.

$$\pi_0 = \min_{y \in E \cap S(x)}\{\bar{c}y + \min_{x \in S}\alpha_0\Delta_0(x - y)\} = 1.$$

At this point if we continue without parameter optimization, π_0 will always re-

main at 1 *and the points* $(k,0)$ *for* $k \geq 2$ *will enter* C *and then* E *one by one. As result,* $(2,1)$ *which corresponds to the optimal solution will never enter* C *and hence the algorithm will never find the optimal solution. This can be seen by noting that* $\Delta_0(k,0) = k$ *and* $\bar{c} \cdot (k,0) = k$ *for* $k \geq 2$*, while* $\Delta_0(0,1) = \frac{1}{2}$ *and* $\bar{c} \cdot (0,1) = 3$.

Otherwise if we use parameter adjustment, the LP will have the form

$$\begin{array}{lll} max & \pi_0 & \\ s.t. & \pi_0 \leq \Delta_0(x) & x \in S_L(0) \\ & \pi_0 \leq 1 + \Delta_0(x - (1,0)) & x \in S_L(0) \\ & \Delta_0(0,1) \leq 3 & \\ & \Delta_0(2,0) \leq 2 & \end{array}$$

which is equivalent to

$$\begin{array}{lll} max & \pi_0 & \\ s.t. & \pi_0 \leq \min\limits_z \{ \max\limits_{\gamma,\alpha} \{\gamma G z + \alpha A z\} | z \geq 0, & \\ & \quad z \text{ } integer \text{ }, Gz \equiv Gx, Hz \geq Hx\} & x \in S \\ & \pi_0 \leq 1 + \min\limits_z \{ \max\limits_{\gamma,\alpha} \{\gamma G z + \alpha A z\} | z \geq 0, & \\ & \quad z \text{ } integer \text{ }, Gz \equiv Gx, Hz \geq Hx - 1\} & x \in S \\ & \max \left\{ \begin{array}{c} \frac{1}{2}\gamma^+ \\ -\frac{1}{2}\gamma^- \end{array} \right\} \leq 3 & \\ & 2\alpha^+ \leq 2. & \end{array}$$

Note that the first constraint can be written as $\pi_0 \leq \alpha^+$ *since*

$$\begin{aligned}
\pi_0 &\leq \min_z\{\max_{\gamma,\alpha}\{\gamma Gz + \alpha Az\} | z \geq 0, z \text{ integer }, Gz \equiv Gx, Hz \geq Hx\} \\
&= \min_z\{\max\left\{\begin{array}{c} \frac{1}{2}\gamma^+ z_2 \\ -\frac{1}{2}\gamma^- z_2 \end{array}\right\} + \max\left\{\begin{array}{c} \alpha^+ z_1 - \alpha^+ z_2 \\ -\alpha^- z_1 + \alpha^- z_2 \end{array}\right\} : \\
&\quad z \geq 0, z \text{ integer }, \tfrac{1}{2}z_2 \equiv \tfrac{1}{2}x_2, z_1 - z_2 \geq x_1 - x_2\} \\
&= \min_z\{\max\{\alpha^+ z_1 - \alpha^+ z_2\} : z \geq 0, z \text{ integer }, \tfrac{1}{2}z_2 \equiv 0, z_1 - z_2 \geq 1\} \\
&= \alpha^+.
\end{aligned}$$

Also we have

$$\begin{aligned}
\Delta_0(0,1) &= \min_z\{\max\left\{\begin{array}{c} \frac{1}{2}\gamma^+ z_2 \\ -\frac{1}{2}\gamma^- z_2 \end{array}\right\} + \max\left\{\begin{array}{c} \alpha^+ z_1 - \alpha^+ z_2 \\ -\alpha^- z_1 + \alpha^- z_2 \end{array}\right\} : \\
&\quad z \geq 0, z \text{ integer }, \tfrac{1}{2}z_2 \equiv \tfrac{1}{2}, z_1 - z_2 \geq -1\} \\
&= \max\left\{\begin{array}{c} \frac{1}{2}\gamma^+ \\ -\frac{1}{2}\gamma^- \end{array}\right\}
\end{aligned}$$

and

$$\begin{aligned}
\Delta_0(2,0) = &= \min_z\{\max\left\{\begin{array}{c} \frac{1}{2}\gamma^+ z_2 \\ -\frac{1}{2}\gamma^- z_2 \end{array}\right\} + \max\left\{\begin{array}{c} \alpha^+ z_1 - \alpha^+ z_2 \\ -\alpha^- z_1 + \alpha^- z_2 \end{array}\right\} : \\
&\quad z \geq 0, z \text{ integer }, \tfrac{1}{2}z_2 \equiv 0, z_1 - z_2 \geq 2\} \\
&= 2\alpha^+
\end{aligned}$$

This LP gives $\alpha^+ = \alpha^- = 1, \gamma^+ = \gamma^- = 6$ *and* $\pi_0 = 1$. *With these new parameters, we will have:*

$$\Delta_0(0,1) = 3 \text{ and } \Delta_0(2,0) = 2.$$

$$x^* = \operatorname*{argmin}_{j=1,\dots,n}\{\frac{\bar{c}x}{\Delta_0(x)} | \Delta_0(x) > 0, x \in C\} = (0,1).$$

New $E = \{(0,0),(1,0),(0,1)\}$ *and* $C = \{(2,0),(1,1),(0,2)\}$. $\alpha_0 = 1$, *and*

$$\pi_0 = \min_{y \in E \cap S(x)} \{\bar{c}y + \min_{x \in S} \alpha_0 \Delta_0(x-y)\} = 1$$

At this point if one continues without parameter optimization, since $\Delta_0(0,2) = 0$, *the points* $(k,0)$ *for* $k \geq 2$ *will enter* C *and then* E *one by one and* $(2,1)$ *will never enter* C. *So again we continue with parameter optimization. We have;*

$$\begin{array}{lll} max & \pi_0 & \\ s.t. & \pi_0 \leq \Delta_0(x) & x \in S_L(0) \\ & \pi_0 \leq 1 + \Delta_0(x-(1,0)) & x \in S_L(0) \\ & \pi_0 \leq 3 + \Delta_0(x-(0,1)) & x \in S_L(0) \\ & \Delta_0(1,1) \leq 4 & \\ & \Delta_0(2,0) \leq 2 & \\ & \Delta_0(0,2) \leq 6. & \end{array}$$

which is equivalent to

$$\begin{aligned}
max \quad & \pi_0 \\
s.t. \quad & \pi_0 \leq \min_z \{ \max_{\gamma,\alpha} \{\gamma G z + \alpha A z\} | z \geq 0, \\
& \qquad z \textit{ integer }, Gz \equiv Gx, Hz \geq Hx\} \qquad\qquad x \in S \\
& \pi_0 \leq 1 + \pi_0 \leq \min_z \{ \max_{\gamma,\alpha} \{\gamma G z + \alpha A z\} | z \geq 0, \\
& \qquad z \textit{ integer }, Gz \equiv Gx, Hz \geq Hx - 1\} \qquad x \in S \\
& \pi_0 \leq 3 + \min_z \{ \max_{\gamma,\alpha} \{\gamma G z + \alpha A z\} | z \geq 0, \\
& \qquad z \textit{ integer }, Gz \equiv Gx - \tfrac{1}{2}, Hz \geq Hx + 1\} \quad x \in S \\
& \max \left\{ \begin{array}{c} \frac{1}{2}\gamma^+ \\ -\frac{1}{2}\gamma^- \end{array} \right\} \leq 4 \\
& 2\alpha^+ \leq 2 \\
& 0 \leq 2
\end{aligned}$$

This LP gives $\alpha^+ = \alpha^- = 1, \gamma^+ = \gamma^- = 8$ *and* $\pi_0 = 1$. *With these new parameters, we will have:*

$$\Delta_0(1,1) = 4, \Delta_0(0,2) = 0 \textit{ and } \Delta_0(2,0) = 2.$$

$$x^* = \operatorname*{argmin}_{j=1,\dots,n} \{ \frac{\bar{c}x}{\Delta_0(x)} | \Delta_0(x) > 0, x \in C\} = (1,1).$$

New $E = \{(0,0),(1,0),(0,1),(1,1)\}$ *and* $C = \{(2,0),(0,2)\}$. $\alpha_0 = 1$, *and*

$$\pi_0 = \min_{y \in E \cap S(x)} \{\bar{c}y + \min_{x \in S} \alpha_0 \Delta(x - y)\} = 1.$$

First Note that for any α *and* $\gamma, \Delta_0(0,2) = 0$. *From this iteration on, since* $(2,0)$ *is in* C *and* $\Delta_0(0,2) = 0$, *again the points* $(k,0)$ *for* $k \geq 3$ *will enter* C *and then* E *one by one and* $(2,1)$ *will never enter* C *and this algorithm will never terminate. Since we have in the constraints of parameter adjustment LP that* $\Delta_0(k,0) \leq k$ *which gives*

$k\alpha^+ \leq k$ [4] and $\alpha^+ \leq 1$, α^+ will never change. Consequently, π_0 will never increase to more than 1 with these settings because of the first constraint in parameter adjustment LP namely $\pi_0 \leq \alpha^+$ which will remain in the LP in all iterations since $(0,0) \in E$ in all iterations.

This problem can be fixed by imposing an upper bound for each variable. In this case, eventually π_0 and α_0 will increase. For instance in the example above, if we have $x_1 \leq K$, then $(0,2)$ will enter C if we get to a point that we have

$$E = \{(0,0),(1,0),...,(K,0)\} \text{ and } C = \{(K+1,0),(0,2)\}.$$

This means that entering $(K+1,0)$ to E will not help since it does not belong to X_I. See [41] for imposing bounds on variables.

The number of steps for the enumeration part of the algorithm is proportional to the maximum coordinate of the optimal solution. For example if x^* is an optimal solution for some IP with $x_i^* = 1000$ for some i, it will take at least 1000 iterations to find the optimal solution. The reason is that x^* must enter C at some iteration and π_0 has to increase to $\bar{c}x^*$. Even if we move multiple points from C to E at each iteration, the x_i for the points in C will increase only by one unit at most. However this algorithm may work better for binary programs.

The following example shows that in order to use Burdet and Johnson's algorithm, one has to do some preprocessing first. If $\delta_i \notin X_I$ then x_i must be eliminated from the IP by letting it be zero. Without this step, later in the algorithm strong duality will not hold.

[4] Note that $\Delta_0(k,0) = k\alpha^+$.

Example 31. *Consider the following pure integer program:*

$$\begin{aligned} \textit{min} \quad & x_1 + \ c_2 x_2 \\ \textit{s.t.} \quad & x_1 + \frac{1}{2} x_2 + x_3 = \frac{1}{2} \\ & x_1, x_2, x_3 \geq 0 \ \textit{integer.} \end{aligned}$$

Equivalent form for the IP is

$$\begin{aligned} \textit{min} \quad & x_1 + \ c_2 x_2 \\ \textit{s.t.} \quad & \frac{1}{2} x_2 \equiv \frac{1}{2} \\ & -x_1 - \frac{1}{2} x_2 \geq -\frac{1}{2} \\ & x_1, x_2 \geq 0 \ \textit{integer.} \end{aligned}$$

Choose any subadditive function with $\Delta(0) = 0$ *and* $\Delta(\delta^1) > 0$*. Here,* $S_I(0) = \{\delta^2\}$ *and* $z_{IP} = c_2$*. For any* $\alpha \geq 0$ *we have*

$$\pi_0 = \min_{x \in S_I(0)} \pi(x) = \min_{y \in E, y \leq \delta^2} \{\bar{c} y + \alpha \Delta(\delta^2 - y)\} = \min\{\alpha \Delta(\delta^2), c_2\}.$$

Since $\alpha \Delta(\delta^1) \leq c_1$*, we get that* $\alpha \leq \dfrac{1}{\Delta(\delta^1)}$*. If* $c_2 > \dfrac{\Delta(\delta^2)}{\Delta(\delta^1)}$*, then* $\pi_0 = \dfrac{\Delta(\delta^2)}{\Delta(\delta^1)} < c_2 = z_{IP}$*. So strong duality does not hold.*

Chapter 4

Klabjan's Algorithm

In this chapter we will describe Klabjan's method for solving IP(2.1) from [35]. Klabjan published this algorithm in 2007 [35]. The underlying theory of the method is based on the algorithms of Burdet and Johnson described in the previous chapter. However, this method is not as complicated as the previous algorithms since the group problem is never used. This algorithm applies only to pure integer programs with non-negative entries in the input data i.e. A and b. Most of the instances that Klabjan has used for numerical experiments are set partitioning problems.

Klabjan defines a family of functions called Subadditive Generator Functions. These functions are feasible to the subadditive dual (2.7) and one can show that it suffices to consider only these functions to achieve a strong duality. So if one rewrites the subadditive dual using these functions (restrict Γ^m to this family), then solving IP (2.1) will become equivalent to optimizing over this family of function.

The main idea is to search for a feasible point (and hence create an upper bound) and at the same time increase the value of the dual function (better lower bound) and gradually close the duality gap. The search is done using the concept of candidate set introduced in the previous chapter and the optimization is done using a linear program similar to adjusting parameters in Section 3.3.

We have discovered some errors in the method as stated in [35] that make it fail to work in general. We will discuss the errors at the end along with a corrected version of the algorithm.

In this chapter and also in the next chapter, instead of A_I (columns of A indexed from I), we will write A^I in accordance with the original works of Klabjan. The same will be valid for x^I instead of x_I. Also, we will denote the set of feasible solutions of IP (2.1) by $\mathcal{F}$.

4.1 Subadditive Generator Functions and Klabjan's Algorithm for pure IPs with Nonnegative Entries

In this chapter we assume that the matrix A and b of integer program (2.1) have non-negative entries and that N denotes the set $\{1, ..., n\}$.

Definition 32. *For integer program (2.1) with non-negative A, a generator subadditive function is defined for a given $\alpha \in \mathbb{R}^m$ as:*

$$F_\alpha(d) = \alpha d - \; max \; \{\sum_{i \in E} (\alpha a_i - c_i) x_i : A^E x \leq d, x \in \mathbb{Z}_+^E\}$$

where $E = \{i \in N \; : \; \alpha a_i > c_i\}$.

Also a ray generator subadditive function is defined for some given $\beta \in \mathbb{R}^m$ as:

$$F_\beta(d) = \beta d - \; max \; \{\sum_{i \in E} (\beta a_i) x_i \; : \; A^E x \leq d \; , \; x \in \mathbb{Z}_+^E\}$$

where $E = \{i \in N \ : \ \beta a_i > 0\}$.

Theorem 33. [35] *Let* $H = N \backslash E$. *For any* α *the following statements are true:*

1. F_α *is subadditive and* $F_\alpha(0) = 0$;

2. $F_\alpha(a_i) \leq \alpha a_i \leq c_i$ *for all* $i \in H$;

3. $F_\alpha(a_i) \leq c_i$ *for all* $i \in E$.

Lemma 34. [35] *Let IP (2.1) be feasible and let* $\pi^j x \leq \pi_0^j, \ j \in V$ *be valid inequalities for the set* $\{x \in \mathbb{Z}_+^n : Ax \leq b\}$ *where* V *is a finite index set. Let*

$$\begin{aligned} z^* = \quad & min \quad cx \\ & s.t. \quad Ax = b \\ & \qquad\ \ \pi^j x \leq \pi_0^j \quad j \in V \\ & \qquad\ \ x \geq 0 \end{aligned} \tag{4.1}$$

and let (α, γ) *be an optimal dual vector for LP(4.1) where* γ *corresponds to constraints* $\pi^j x \leq \pi_0^j, \ j \in V$. *Then* $F_\alpha(b) \geq z^*$.

Theorem 35. [35] *If IP (2.1) is feasible, then there exists an* α *such that* F_α *is an optimal generator function; i.e.* $F_\alpha(b) = cx^*$. *If it is infeasible, then there exists a ray generator function* F_β *such that* $F_\beta(b) > 0$.

It is enough to restrict the class of functions used in the subadditive dual to a subset of generator functions called *Basic Generator Functions.* These functions are enough to get strong duality and facets of IP(2.1). The problem is to find an α with best $F_\alpha(b)$. It is observed that the optimum value of IP(2.1) is equal to

$$\max\{\eta : (\eta, \alpha) \in Q_b(E)\} \tag{4.2}$$

where

$$Q_b(E) = \begin{aligned} &\{(\eta,\alpha) \in (\mathbb{R}\times\mathbb{R}^m) | \alpha a_i \le c_i \text{ for } i \in H \text{ and} \\ &\eta + \alpha(A^E x - b) \le c^E x \text{ for all } x \in \mathbb{Z}^E \text{ with } A^E x \le b\} \end{aligned} \tag{4.3}$$

for some E. It can be shown that $Q_b(E)$ is a polyhedron. See [35] for details.

Extreme points and extreme rays of these polyhedra correspond to basic generator functions in the following sense:

Definition 36. *A generator function F_α is called basic if $(F_\alpha(b), \alpha)$ is an extreme point of (4.3). Also a ray generator function F_β is called a basic ray generator function if $(F_\beta(b), \beta)$ is an extreme ray of (4.3).*

In order to find the optimal value of IP (2.1), one can solve

$$\max_{\alpha\in\mathbb{R}^m} F_\alpha(b).$$

Since this is as hard as the original problem IP(2.1) to solve[1], Klabjan relaxes the problem to another one using function π that he defines. Let

$$\pi(x) = \alpha A x - \max_{y\in U\cap S(x)} \{(\alpha A - c)y\} \tag{4.4}$$

where $E = \cup_{x\in U} supp(x)$ and U is any subinclusive subset of $\{A^E x \le b : x \in \mathbb{Z}^n_+\}$ and $S(x) = \{y \in \mathbb{Z}^n_+ : \ y \le x\}$.

If $\pi(e_i) \le c_i$ for all i, then π is said to be dual feasible. If π is dual feasible and

[1]Solving this problem directly, is equivalent to solving a number of IPs which are comparable in size to the original IP.

subadditive and x is feasible to IP(2.1), we will have:

$$\pi(x) = \pi(\sum_{j=1}^{n} x_j e_j) \leq \sum_{j=1}^{n} \pi(e_j) x_j \leq \sum_{j=1}^{n} c_j x_j = cx. \tag{4.5}$$

Klabjan claims that π gives a lower bound for the optimal value of IP for all x feasible to IP(2.1). This may not be true because $\pi(x)$ is not necessarily a lower bound on cx^*. For example if $E = \emptyset$, then $\pi(x) = \alpha Ax = \alpha b$ which obviously doesn't give a lower bound unless for specific α. However, $\min\limits_{x \in \mathbb{Z}_+^n, Ax=b} \pi(x)$ gives a lower bound on cx^*.

So the problem is relaxed to

$$\begin{aligned} \max_{\alpha} \quad & \min_{x \in \mathbb{Z}_+^n, Ax=b} \pi(x) \\ & \pi(e_i) \leq c_i \\ & \pi \text{ subadditive} \end{aligned} \tag{4.6}$$

Klabjan writes maximization instead of minimization ($\max\limits_{\alpha} \max\limits_{x \in \mathbb{Z}_+^n, Ax=b} \pi(x)$) which must be a typo since this problem may not be a strong dual if one uses maximization.

To solve (4.6), Klabjan uses a method similar to Burdet and Johnson's [16]: Given the set U define the candidate set $V = \{x : \ x \notin U, \ S(x) \backslash \{x\} \subseteq U\}$. Now solve

$$\begin{array}{llll} \max & \pi_0 & & \\ \text{s.t.} & \pi_0 + \alpha(Ay - b) \leq cy & y \in U & \qquad D(U,V) \\ & \alpha Ax \leq cx & x \in V & \\ & \alpha, \pi_0 \text{ unrestricted} & & \end{array}$$

with $U = \{0\}$ at the beginning. V will be the set $\{\delta^i, \ i = 1, ..., n\}$ at the first step. The first set of constraints capture the objective value and the second set keeps π

subadditive.

At each step an integer point is added from V to U and the LP is updated and solved again until an optimal point enters V. So Klabjan's algorithm will be in the form of Algorithm 4.1.

```
Data: An integer program and optimal dual vector for LP relaxation
Result: Optimal solution and optimal value to IP
    Initialization: $U = \{0\}$ and $V = \{\delta^i : i = 1, ..., n\}$ and set $\alpha$ to be
    the optimal dual vector for LP relaxation of IP (2.1). $z^{IP} = -\infty$
    while $z^{IP} \neq min\{cx : \ Ax = b, \ x \in V\}$ do
        choose a point $\bar{x}$ from $V$.
        update $U$ and $V$: $U = U \cup \{\bar{x}\}, V = V \cup \{\bar{x} + \delta^i : i \in N\}$
        update $\alpha$ by solving $D(U, V)$ and get $\pi_0^*$.
        $z^{IP} = \max\{z^{IP}, \pi_0^*\}$.
    end
```

Algorithm 4.1: Klabjan's Algorithm to solve IP

Once the optimal value to the IP (2.1) has been found, and one may want to find α and corresponding $E(\alpha)$ as well. The α found in the first algorithm is not optimal since π is just a relaxation of F_α. A generator subadditive function is called optimal over $S \subseteq N$, if it is optimal for the IP $\min\{c^S x : A^S x = b, x \geq 0 \text{ integer }\}$.

Instead of finding α for $S = N$ at once, the algorithm tries to find optimal α over $S \subseteq N$ and expands gradually. Starting with $E = E(\alpha)$, let H be a subset of columns with lowest and negative $\alpha a_i - c_i$. This choice of H is based on the fact that these columns are likely candidates for H in the final optimal subadditive function. Set $S = E \cup H$.

The algorithm will either

1. greedily expand S by a new column from $N \backslash S$; or
2. not change S but expand E by a new column from $S \backslash H$.

Klabjan's second algorithm for finding an optimal F_α is stated in Algorithm 4.2.

```
**Data**: An integer program, best available $\alpha$, $x^*$ and $z^*_{IP}$
**Result**: Optimal $F_\alpha$
    Initialization: Let $j$ be the column where
    $\max\{\alpha a_i - c_i : i \in N \backslash S\}$ is attained and set $S = S \cup \{j\}$ and
    $E = E \cup \{j\}$.
    while $S \neq N$ do
        solve (4.2) with $A = A^S$ ans $S = E \cup H$ and let $\alpha$ be the
        optimal solution and $\eta^*$ the objective value.
        if $\eta^* = z^{IP}$ then
            if $S = N$ then
                $F_\alpha$ is the optimal and exit.
                else
                    $\alpha$ is optimal over $S$, expand $S$.
                    let $j$ be the column where
                    $\max\{\alpha a_i - c_i : i \in N \backslash S\}$ is attained
                    set $S = S \cup \{j\}$ and $E = E \cup \{j\}$
                end
            end
            else
                expand $E$. Select a subset $\tilde{E}$ of columns from $S \backslash E$
                and set $E = E \cup \tilde{E}$.
                $S = S \cup \{i \in N \backslash S : \alpha a_i \leq c_i\}$ and $H = S \backslash E$.
            end
        end
    end
```

Algorithm 4.2: Klabjan's Second Algorithm to Find Optimal F_α

4.2 Deficiencies of Klabjan's Algorithm

There are four main issues (errors) with Klabjan's algorithm:

1. Each point added from V to U should be removed from V; otherwise, the set V will not satisfy the definition of the candidate set and the subadditivity of π is violated.

2. Any point in V that is not feasible to the original IP should be removed from V or else the bound π_0 will never increase and will never close the gap.

3. The search is not directed: it is possible that we add points from V to U in a direction that never lets the optimal solution enter U. This, however, will not happen in bounded problems if we use the corrected version of the algorithm which we will see below. For unbounded problems, bounding strategies from the previous chapter would resolve the error.

4. To choose which point enters U first, we have to choose one that violates the constraint
$$\alpha Ax \leq cx \text{ for } x \in V$$
first. Burdet and Johnson use a parameter α_0 in 3.3 to do this. However, Klabjan doesn't use any rules for this and the points are chosen arbitrarily. This may cause π lose subadditivity.

Klabjan's Algorithm fails to solve the following example:

Example 37.

$$\begin{aligned} min \quad & -3x_1 - x_2 \\ s.t. \quad & 2x_1 + x_2 = 3 \\ & x_1, x_2 \geq 0, integer. \end{aligned}$$

For this problem, $z^*_{IP} = -4$ and $z^*_{LPR} = -4.5$. At the second step of the algorithm, $U = \{(0,0),(1,0)\}$ and $V = \{(2,0),(0,1)\}$ and $D(U,V)$ reads:

$$\begin{aligned} max \quad & \pi_0 \\ s.t. \quad & \pi_0 - 3\alpha \leq 0 \\ & \pi_0 - \alpha \leq -3 \\ & \alpha \leq -1 \\ & 4\alpha \leq -6. \end{aligned}$$

The optimal value of $D(U,V)$ is -4.5 and it will never increase because of two reasons:

1. $\pi_0 \leq 3\alpha$ *will always be a constraint;*
2. *we will always have a point like $(k,0)$ in V with $k \geq 0$, which will result in having the constraint $2k\alpha \leq -3k$ so this will give us $\alpha \leq -1.5$. This in turn causes $\pi_0 \leq -4.5$ because of the constraint $\pi_0 \leq 3\alpha$.*

1 and 2 together show that π_0 will never increase from -4.5. The reason that this happens is that the α that the algorithm finds is a linear programming dual feasible vector.

Klabjan gives enhancements to the algorithm. These enhancements help to reduce the number of constraints in $D(U,V)$. See [35]. Also, much preprocessing was applied to the problems. Using these enhancements and preprocessing, Klabjan solves the set partitioning problems [35].

4.3 Correcting Klabjan's Algorithm

Given $U \subseteq \{x \in \mathbb{Z}_+^N : Ax \leq b\}$ and $\alpha \in \mathbb{R}^m$, define

$$\pi_{\alpha,U}(x) = \alpha Ax - \max_{y \in U, y \leq x} \{\sum_{i \in E} (\alpha a_i - c_i) y_i\}$$

where $E = \bigcup_{x \in U} supp(x)$.

Theorem 38. *If $\pi_{\alpha,U}(x)$ is subadditive and $\pi_{\alpha,U}(\delta^i) \leq c_i$ for all $i \in N$, then $\min_{x \in \mathcal{F}} \pi_{\alpha,U}(x)$ is a lower bound for the optimum value of IP (2.1).*

Proof.

$$\pi_{\alpha,U}(x) = \pi_{\alpha,U}(\sum_{i=1}^{n} \delta^i x_i) \leq \sum_{i=1}^{n} \pi_{\alpha,U}(\delta^i) x_i \leq \sum_{i=1}^{n} c_i x_i = cx. \tag{4.7}$$

So we have

$$\min_{x \in \mathcal{F}} \pi_{\alpha,U}(x) \leq \min_{x \in \mathcal{F}} cx = z_{IP}^*.$$

$\square$

Theorem 39. *There exists a subinclusive set $U \subseteq \{x \in \mathbb{Z}_+^N : Ax \leq b\}$ and $\alpha \in \mathbb{R}^m$ such that*

1. *$\pi_{\alpha,U}$ is subadditive;*
2. *$\pi_{\alpha,U}(\delta^i) \leq c_i$ for all $i \in N$;*
3. *$\min_{x \in \mathcal{F}} \pi_{\alpha,U}(x) = z_{IP}^*$.*

Proof. Let $F_\alpha(x)$ be the subadditive generator function defined by Klabjan in [35] i.e.

$$F_\alpha(d) = \alpha d - \max \{\sum_{i \in E} (\alpha a_i - c_i) x_i : A^E x \leq d, x \in \mathbb{Z}_+^E\}$$

where $E = \{i \in N \ : \ \alpha a_i > c_i\}$.

Let α be such that $F_\alpha(b) = z^*_{IP}$. According to Klabjan [35] there always exists such α. Also let

$$U = \{x \in \mathbb{Z}^N_+ : Ax \leq b, x_i = 0 \text{ if } i \notin E\}$$

where $E = \{i \in N \ : \ \alpha a_i > c_i\}$. Note that $E = \bigcup_{x \in U} supp(x)$. Then subadditivity of $\pi_{\alpha,U}$ becomes equivalent to

$$\begin{aligned} &\max \{\sum_{i \in E} (\alpha a_i - c_i) y_i : A^E y \leq b, y \leq x_1, y \in \mathbb{Z}^E_+\} \\ &+ \max \{\sum_{i \in E} (\alpha a_i - c_i) y_i : A^E y \leq b, y \leq x_2, y \in \mathbb{Z}^E_+\} \\ &\leq \ \max \{\sum_{i \in E} (\alpha a_i - c_i) y_i : A^E y \leq b, y \leq x_1 + x_2, y \in \mathbb{Z}^E_+\} \end{aligned} \tag{4.8}$$

But it is enough that $\pi_{\alpha,U}$ is subadditive on $\mathcal{F}$. Suppose that $x_, x_2$ and $x_1 + x_2$ are all in $\mathcal{F}$. Now let y^*_1 and y^*_2 be the optimal solutions to the two maximization problems on the left. Then $y^*_1 + y^*_2 \leq x_1 + x_2$ so

$$A^E(y^*_1 + y^*_2) \leq A^E(x_1 + x_2) \leq A(x_1 + x_2) = b,$$

which proves the desired result.

For the second part, we have

$$\pi_{\alpha,U}(\delta^i) = \alpha a_i - \max_{A^E y \leq b, y \leq \delta^i} \{\sum_{i \in E} (\alpha a_i - c_i) y_i\}.$$

Since in the maximization problem, y will take δ^i if $i \in E$, we have $\pi_{\alpha,U}(\delta^i) \leq c_i$. If $i \notin E$, then

$$\pi_{\alpha,U}(\delta^i) = \alpha a_i \leq c_i.$$

Now suppose that $x \in \mathcal{F}$. Then we have:

$$\pi_{\alpha,U}(x) = \alpha b - \max\{\sum_{i \in E}(\alpha a_i - c_i)y_i, A^E y \le b, y \le x, y \in \mathbb{Z}_+^n\}.$$

With the choice of U and α, and since x is in $\mathcal{F}$ we have

$$\min_{x \in \mathcal{F}} \pi_{\alpha,U}(x) = F_\alpha(b) = z_{IP}^*.$$

□

Finding the optimal solution of IP (2.1) becomes equivalent to the problem

$$\max_{U,\alpha} \min_{x \in F} \pi(x) \tag{4.9}$$

where maximum is taken over all $U \subseteq \{x \in \mathbb{Z}_+^N : Ax \le b\}$ and $\alpha \in \mathbb{R}^m$. This problem is equivalent to

$$\begin{array}{lll} \max_U & \min & \pi_0 \\ & \text{s.t.} & \pi_0 \le \alpha Ax - (\alpha A - c)y \quad \text{for all } x \in F, y \in U, y \le x \\ & & \pi_{\alpha,U} \text{ subadditive.} \end{array} \tag{4.10}$$

Let V be the candidate set for $U \subseteq \{x \in \mathbb{Z}_+^N : Ax \le b\}$.

Lemma 40. *If $\alpha Ax \le cx$ for every $x \in V$, then $\pi_{\alpha,U}$ is subadditive.*

See [35] for proof. One lower bound for (4.10) is given by

$$\begin{array}{llll} \max_U & \min & \pi_0 & \\ & \text{s.t.} & \pi_0 + \alpha(Ay - b) \le cy & \text{for every } y \in U \\ & & \alpha Ax \le cx & \text{for every } x \in V. \end{array} \tag{P}$$

The reason is that if $\alpha Ax \leq cx$ for every $x \in V$ is satisfied, then π is subadditive, but the converse is not true. So the feasible region of (P) is actually smaller than the feasible region of (4.10). However, this LP would still catch the value $\pi_0 = cx^*$ by Theorem 39 part 3, since π is still subadditive.

We now give an algorithm that computes a U and an α with $F_\alpha(b) = z^*_{IP}$. Let $D(U, V)$ denote the following linear program

$$\begin{array}{lll} \min & \pi_0 & \\ \text{s.t.} & \pi_0 + \alpha(Ay - b) \leq cy & \text{for every } y \in U \\ & \alpha Ax \leq cx & \text{for every } x \in V. \end{array}$$

Steps of the algorithm are stated in Algorithm 4.3.

```
Data: An integer program and optimal dual vector for LP
      relaxation
Result: Optimal solution to IP
    Initialization: $U = \{0\}$ and $V = \{\delta^i : i = 1, ..., n\}$ and set $\alpha$
    to be the optimal dual vector for LP relaxation of IP (2.1).
    $z^{IP} = -\infty$.
    while $z^{IP} \neq min\{cx : \ Ax = b, \ x \in V\}$ do
        Let $\bar{x} = \text{argmin}\{cx - \alpha Ax : x \in V\}$.
        Update $U$ and $V$:
        $U = U \cup \{\bar{x}\}, V = V \cup \{\bar{x} + \delta^i : i \in N\} \backslash \{\bar{x}\}$.
        Remove from $V$ all the points $x$ with $Ax \leq b$ not
        satisfied.
        if $V = \emptyset$ then
            solve $D(U, V)$ and stop. $\pi_0^*$ is equal to the the
            optimal value of IP(2.1).
        end
        Update $\alpha$ by solving $D(U, V)$ and get $\pi_0^*$.
        $z^{IP} = \max\{z^{IP}, \pi_0^*\}$.
    end
```

Algorithm 4.3: Corrected Version of Klabjan's Algorithm

All the four errors mentioned above have been corrected in the new version. However to overcome the third error, we still have to use bounds on variables. Klabjan's algorithm may have to enumerate all feasible points to find an optimal solution. So in the worst case, it can have exponential running time, however imposing bounds on variables can reduce the size of the feasible set and number of iterations as well.

Theorem 41. *Algorithm 4.3 will find the optimal value of IP(2.1) in a finite number of iterations.*

Proof. It is obvious that since the feasible set is finite, eventually (in the worst case) an optimal solution will enter the candidate set. On the other hand, π_0 will increase since we now are removing points from C and by Theorem 39 we know that there exists a U (C) with $\pi_0 = \min_{x \in \mathcal{F}} \pi_{\alpha,U}(x) = z^*_{IP}$. This will guarantee that the algorithm will terminate with an optimal solution of the IP. However, the corrected algorithm is faster due to the directed search we have added to the algorithm. $\square$

Chapter 5

Optimality Certificates and Sensitivity Analysis using Branch and Bound Tree and Generator Functions

An important aspect of computational research is the verifiability of reported benchmark results [44]. Generating optimality certificates for mixed integer programming problems is getting more important every day. Especially when a very hard problem of large size is solved, checking the optimality of the solution at hand might not be easy. For example, see [1] where the authors provide an optimality certificate for an optimal TSP tour through 85,900 cities. This certificate is mostly going over the branch and bound tree and checking optimality at each node, and also checking the validity of each cutting plane that has been added. The time needed to check the optimality using this certificate might take up to 568.9 hours (24 days). The actual time for solving the problem is 286.2 days so this is a good certificate (we will define what "good" means later in this chapter).

Today most of the optimization problems in science and industry are solved using major commercial solvers such as Gurobi and CPLEX. They are reliable to some extent, however there are instances of mixed integer programming problems that both these solvers fail to solve correctly. See [18] for such examples and related discussions.

Subadditive Generator Functions defined by Klabjan can serve as optimality certificates for specific families of integer programs such as knapsack or set covering problems. However they are restricted to pure integer programs with A and b non-negative. In this chapter, we first generalize these functions to any mixed integer linear program and then we will show that these functions are feasible in the subadditive dual and actually they are enough to get strong duality; i.e. if we restrict Γ^m to this family, strong duality still holds. Then we will make use of these functions for generating optimality certificates.

We have generalized the subadditive functions defined by Klabjan. However, we have not extended his algorithms to MILP since they would not be very useful due to the undirected search in the first algorithm and also expensive running times in the second. We will talk more about this in Chapter 6 of the thesis.

One application of this extended class of functions is in finite representation of the convex hull of feasible points to a mixed integer program. In 1981, Wolsey [45] showed that the convex hull of feasible points to a pure IP can be described with a finite number of valid inequalities corresponding to subadditive dual feasible functions. In 1980, Bachem and Schrader [2] had shown that the convex hull of feasible points to an MILP can be described with subadditive dual feasible functions and the corresponding cutting planes but this representation might not be finite. We will show that a finite representation is possible using generalized subadditive generator functions.

Sensitivity analysis is another important topic in MILP studies. It has been studied in the work of Wolsey [47] using Chvátal functions and value functions of integer programs and also by John Hooker in [29] using branch and bound tree and inference duality (also see [28] for a survey). However, in both cases, for a large-size problem, it is not easy to perform sensitivity analysis. In this chapter, we will see that if we have an optimal dual feasible function, we can generalize the tools for sensitivity analysis in linear programming to mixed integer programming.

In the first section of this chapter, we generalize Klabjan's functions. We prove a finite representation of MILP using these functions in Section 5.2. In Section 5.3, we describe our algorithm to generate an optimality certificate for MILP using the branch and bound tree as well as the tools for sensitivity analysis. In the last section, we present some of our computational experiments and numerical results.

Here we state a result by Meyer [37] that we will use later in this chapter.

Theorem 42. *(Meyer* [37]*) Given a rational matrix A, G and a rational vector b, let $P := \{(x, y) : Ax + Gy \leq b\}$ and let $S := \{(x, y) \in P : x$ integral $\}$.*

1. *There exist rational matrices A', G' and a rational vector b' such that conv$(S) = \{(x, y) : A'x + G'y \leq b'\}$.*

2. *If S is nonempty, the recession cones of conv(S) and P coincide.*

Proof. See [37]. $\square$

5.1 Generator Subadditive Functions: Generalized

Throughout this chapter $N = \{1, 2, ..., n\}$ and $C = N \backslash I$. Consider MILP (2.3) and its subadditive dual. Let $E(\alpha) = \{i \in N : \alpha a_i > c_i\} \cup \{i \in N : a_i \ngeq 0\}$ for $\alpha \in \mathbb{R}^m$.

Also let $S_\alpha(\ell) = \{x \in \mathbb{Z}_+^I \times \mathbb{R}_+^C : \sum_{i \in E(\alpha)} a_i x_i \leq \ell\}$ and define

$$\Delta_\alpha = \{\ell \in \mathbb{R}^m : S_\alpha(\ell) \neq \emptyset\}.$$

For $\alpha \in \mathbb{R}^m$ define the Subadditive Generator Function $F_\alpha(\ell) : \Delta_\alpha \to \mathbb{R} \cup \{-\infty\}$ by

$$\begin{aligned} F_\alpha(\ell) &= \alpha\ell - \max\{\sum_{i \in E(\alpha)} (\alpha a_i - c_i)x_i : \\ &\qquad \sum_{i \in E(\alpha)} a_i x_i \leq \ell, x \in \mathbb{Z}_+^{I \cap E} \times \mathbb{R}_+^{C \cap E}\}. \end{aligned}$$

We will write E instead of $E(\alpha)$ if α is fixed. The definition of generator functions has been extended from work of Klabjan [35]. They were previously defined for a restricted family of pure integer programs. The extended version applies to any MILP in general.

Theorem 43. *$F_\alpha(a_i) \leq c_i$ for all $i \in I$ and $\overline{F}_\alpha(a_i) \leq c_i$ for all $i \in C$.*

Proof. First we will show that $F_\alpha(a_i) \leq c_i$ for all $i \in I$. We have

$$\begin{aligned} F_\alpha(a_i) &= \alpha a_i - \max\{\sum_{i \in E(\alpha)} (\alpha a_i - c_i)x_i : \\ &\qquad \sum_{i \in E(\alpha)} a_i x_i \leq a_i, x \in \mathbb{Z}_+^{I \cap E} \times \mathbb{R}_+^{C \cap E}\}. \end{aligned}$$

There are two cases: for $i \in I$, i is either in E or not. If $i \in E$, then $x = \delta^i$ is a feasible solution to the maximization problem where δ^i is the unit vector with 1 as the i-th component and zero otherwise. This gives us $F_\alpha(a_i) \leq c_i$. If $i \notin E$, then $x = 0$ is a feasible solution to the maximization problem and gives us $F_\alpha(a_i) \leq \alpha a_i \leq c_i$.

To show that $\overline{F}_\alpha(a_i) \leq c_i$ for all $i \in C$, first note that Gomory and Johnson [24] show that if $\overline{F}_\alpha(\ell)$ is finite, then the limsup and the ordinary limit coincide. Now we have:

$$\begin{aligned}\overline{F}_\alpha(a_i) &= \lim_{h\to 0^+} \frac{F_\alpha(ha_i)}{h} \\ &= \alpha a_i - \lim_{h\to 0^+} \frac{1}{h}\max\{\sum_{i\in E(\alpha)} (\alpha a_i - c_i)x_i : \\ &\qquad\qquad \sum_{i\in E(\alpha)} a_i x_i \le h a_i, x \in \mathbb{Z}_+^{I\cap E} \times \mathbb{R}_+^{C\cap E}\}.\end{aligned}$$

If i is in E, then $x = h\delta^i$ is a feasible solution to the maximization problem where δ^i is the unit vector. Then the maximum will be greater than or equal to $h(\alpha a_i - c_i)$, so the limit will be greater than or equal to $\alpha a_i - c_i$ since $h > 0$. This gives us $\overline{F}_\alpha(a_i) \le c_i$. If $i \notin E$, then $x = 0$ is a feasible solution to the maximization problem and a similar argument gives us $\overline{F}_\alpha(a_i) \le \alpha a_i \le c_i$. □

Theorem 44. *$F_\alpha(\ell)$ is subadditive on Δ_α.*

Proof. The subadditivity of F_α, i.e. $F_\alpha(\ell_1) + F_\alpha(\ell_2) \ge F_\alpha(\ell_1 + \ell_2)$, is equivalent to

$$\begin{aligned}&\max\{\sum_{i\in E(\alpha)} (\alpha a_i - c_i)x_i : \sum_{i\in E(\alpha)} a_i x_i \le \ell_1, x \in \mathbb{Z}_+^{I\cap E} \times \mathbb{R}_+^{C\cap E}\} + \\ &\max\{\sum_{i\in E(\alpha)} (\alpha a_i - c_i)x_i : \sum_{i\in E(\alpha)} a_i x_i \le \ell_2, x \in \mathbb{Z}_+^{I\cap E} \times \mathbb{R}_+^{C\cap E}\} \le \\ &\max\{\sum_{i\in E(\alpha)} (\alpha a_i - c_i)x_i : \sum_{i\in E(\alpha)} a_i x_i \le \ell_1 + \ell_2, x \in \mathbb{Z}_+^{I\cap E} \times \mathbb{R}_+^{C\cap E}\}.\end{aligned}$$

If ℓ_1^* and ℓ_2^* are optimal solutions to the two maximums on the left hand side of this inequality, then $\ell_1^* + \ell_2^*$ is a feasible solution to the maximization problem on the right side. The only case that remains is when one of the maxzimization problems on the left hand side is unbounded. In this case obviously the maximization problem on the right would be unbounded as well. □

The following lemma shows that $F_\alpha(0) = 0$ for any $\alpha \in \mathbb{R}^m$.

Lemma 45. *Let F be a subadditive function defined on a monoid*[1] *M to $\mathbb{R} \cup \{-\infty\}$. If $F(0) < 0$, then F is identically $-\infty$.*

Proof. Suppose that $F(0) < 0$ and there exists $\ell \in M$ with $F(\ell) > -\infty$. Then we have

$$-\infty < F(\ell + 0) \leq F(\ell) + F(0) \Longrightarrow -\infty < F(\ell) \leq F(\ell) + F(0) \Longrightarrow F(0) \geq 0$$

which is a contradiction. So for all $\ell \in M$, $F(\ell) = -\infty$. □

It is easily observed that Δ_α is a monoid for any α and so we can state the following theorem:

Theorem 46. *(Strong Duality) If MILP (2.3) is feasible, then there exists $\alpha \in \mathbb{R}^m$ with $F_\alpha(b) = z^*_{MILP}$ and $F_\alpha(0) = 0$.*

Proof. Let MILP (2.3) be feasible and let $\pi^j x \leq \pi_0^j$, $j \in J$ be valid inequalities for the set

$$V = \{x \in \mathbb{Z}_+^I \times \mathbb{R}_+^C : Ax \leq b\}$$

such that

$$\begin{aligned} z^*_{MILP} = \quad \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & \pi^j x \leq \pi_0^j \quad j \in J \\ & x \geq 0. \end{aligned}$$

This is possible by Theorem 42 where it is shown that if A, b and c in MILP (2.3) are rational, then the convex hull of the feasible points is a polyhedron (finitely

[1] A set with a binary operation is called a monoid if it is closed under that operation, it satisfies the associativity property and has an identity element.

generated). Let (α, γ) be an optimal dual vector where γ corresponds to constraints $\pi^j x \leq \pi_0^j,\ j \in J$. We show that $F_\alpha(b) \geq z^*_{MILP}$.

The dual program of the above LP is

$$\begin{array}{ll} \max & b\alpha - \sum_{j \in J} \pi_0^j \gamma_j \\ \text{s.t.} & \alpha a_i - \sum_{j \in J} \pi_i^j \gamma_j \leq c_i \qquad i \in N \\ & \alpha \text{ unrestricted}, \gamma \geq 0. \end{array}$$

The optimal value of this problem is z^*_{MILP}. Let $x \in \mathbb{Z}_+^I \times \mathbb{R}_+^C$ be a vector such that $\sum_{i \in E(\alpha)} a_i x_i \leq b$. Such a vector exists since we have assumed that MILP (2.3) is feasible and we know that E contains all the columns which are not entirely non-negative. So any column not in E will be entirely non-negative. We have

$$\begin{aligned} \sum_{i \in E(\alpha)} (\alpha a_i - c_i) x_i &\leq \sum_{i \in E(\alpha)} \sum_{j \in J} x_i \pi_i^j \gamma_j \\ &= \sum_{j \in J} \gamma_j \Big(\sum_{i \in E(\alpha)} x_i \pi_i^j \Big) \\ &\leq \sum_{j \in J} \gamma_j \pi_0^j = b\alpha - z^*_{MILP}. \end{aligned}$$

The last inequality holds because $\gamma \geq 0$ and the fact that $\pi^j x \leq \pi_0^j$ is a valid inequality for the set V. So we get that

$$\max \Big\{ \sum_{i \in E(\alpha)} (\alpha a_i - c_i) x_i : \sum_{i \in E(\alpha)} a_i x_i \leq b, x \in \mathbb{Z}_+^{I \cap E} \times \mathbb{R}_+^{C \cap E} \Big\} \leq b\alpha - z^*_{MILP}.$$

So we have $F_\alpha(b) \geq z^*_{MILP}$. Also, we know that $F_\alpha(b) \leq z^*_{MILP}$ since F_α is feasible to subadditive dual problem for MILP (2.3). (See [28] for examples of subadditive duality.) So $F_\alpha(b) = z^*_{MILP}$. Also, by Lemma 45, $F_\alpha(0) \geq 0$ and we have $F_\alpha(0) \leq 0$

since for $F_\alpha(0)$, zero is a feasible solution to the maximization problem. This shows that $F_\alpha(0) = 0$ and the proof is complete. $\square$

5.2 Finite Representation of Mixed Integer Polyhedra: An Extension of Wolsey's Result

Wolsey proved [45] that for pure integer programming, there is a finite representation, i.e. if $P = \{x \in \mathbb{R}^\ell_+ : \ Ax = b\}$ for some ℓ, then there exist $F^1, ..., F^k \in \Gamma^m$ for some integer $k > 0$ such that

$$conv(P \cap \mathbb{Z}^\ell_+) = \{x : \sum_{j \in I} F^i(a_j)x_j \geq F^i(b), \ i = 1, ..., k, x \geq 0\}.$$

Bachem and Schrader [2] showed that if $P = \{x : Ax = b\}$, then $conv(P \cap \mathbb{Z}^E_+ \times \mathbb{R}^C_+)$ can be written as

$$\{x : \sum_{j \in I} F(a_j)x_j + \sum_{j \in C} \overline{F}(a_j)x_j \geq F(b), F \in \Gamma^m, x \geq 0\}.$$

This means that there is a representation for any mixed integer programming problem using subadditive functions.

Here we will prove a more general case for mixed integer programs using basic generator functions.

If F is any subadditive function with $F(0) = 0$ and dual feasible, then

$$\sum_{j \in I} F(a_j)x_j + \sum_{j \in C} \overline{F}(a_j)x_j \geq F(b)$$

is a valid inequality for MILP (2.3).

It is enough to restrict our attention to a subset of generator functions called basic generator functions. These functions are enough to describe the convex hull of an MILP finitely.

Theorem 47. *The optimum value of MILP (2.3) is equal to* $\max\{\eta : (\eta, \alpha) \in Q_b(E)\}$ *where*

$$
\begin{aligned}
Q_b(E) = \{ \quad & (\eta, \alpha) \in (\mathbb{R} \times \mathbb{R}^m) : \qquad \alpha a_i \leq c_i \text{ for } i \in N \backslash E \\
& \eta + \alpha(A^E x - b) \leq c^E x, \quad A^E x \leq b, \quad x \in \mathbb{Z}_+^{E \cap I} \times \mathbb{R}_+^{E \cap C} \}
\end{aligned}
\tag{5.1}
$$

for some E.

Proof. By Theorem 46 there exists $\alpha \in \mathbb{R}^m$ with $F_\alpha(b) = z^*_{MILP}$. Choose $E = E(\alpha)$ and consider $Q_b(E)$. Then

$$
\begin{aligned}
\eta^* &= \max\{\eta : (\eta, \alpha) \in Q_b(E)\} \\
&= \max\{\eta : \eta \leq c^E x - \alpha(A^E x - b) : A^E x \leq b, x \in \mathbb{Z}_+^{E \cap I} \times \mathbb{R}_+^{E \cap C}\} \\
&= \alpha b + \max\{(c^E - \alpha A^E) x : A^E x \leq b, x \in \mathbb{Z}_+^{E \cap I} \times \mathbb{R}_+^{E \cap C}\} \\
&= F_\alpha(b) = z^*_{MILP}.
\end{aligned}
$$

□

Theorem 48. $Q_b(E)$ *is a polyhedron.*

Proof. Obviously, the set $K = conv(\{x : A^E x \leq b, x \in (\mathbb{Z}_+^{E \cap I} \times \mathbb{R}_+^{E \cap C})\})$ is a polyhedron by Theorem 42. So it has a finite number of extreme points and extreme rays. Let M and N denote the set of extreme points and extreme rays of K respectively.

Then since M and N are finite, we have

$$Q_b(E) = \{(\eta, \alpha) \in (\mathbb{R} \times \mathbb{R}^m) : \begin{array}{l} \alpha a_i \le c_i \text{ for } i \in N \backslash E \\ \eta + \alpha(A^E x - b) \le c^E x \text{ for } x \in M \\ \eta + \alpha(A^E y - b) \le c^E y \text{ for } y \in N\} \end{array}$$

which is obviously a polyhedron (finitely generated). □

Definition 49. *A generator function F_α is called basic if $(F_\alpha(b), \alpha)$ is an extreme point of (5.1).*

Since there are finite choices for E and for each E, $Q_b(E)$ has a finite number of extreme points, there are only a finite number of basic generator functions. We denote their index set by $K(b)$.

Suppose that z^* is the optimal solution for MILP (2.3) and $K(b)$ is the set of indices of all basic generator functions. Then

$$\begin{aligned} z^* = \quad & \min \quad cx \\ & \text{s.t.} \quad Ax = b \\ & \qquad \sum_{j \in I} F_{\alpha_k}(a_j) x_j + \sum_{j \in C} \overline{F_{\alpha_k}}(a_j) x_j \ge F_{\alpha_k}(b) \quad k \in K(b) \\ & \qquad x \ge 0. \end{aligned} \tag{5.2}$$

It is obvious that we only need basic generator subadditive functions. Since there are only a finite number of them, the following theorem holds.

Theorem 50. *Given A, b and c, there exists a finite set of subadditive generator functions such that the linear program (5.2) has the following properties:*

1. *LP (5.2) is infeasible if and only if MILP (2.3) is infeasible.*

2. *LP (5.2) has unbounded optimum value if and only if MILP (2.3) has unbounded optimum value.*

3. *Otherwise LP (5.2) has an optimal extreme point solution which is also optimal for MILP (2.3).*

Also, since we know that the convex hull of feasible solutions to MILP (2.3) is polyhedral, i.e. it can be described by a finite set of facet defining valid inequalities, we have the following corollary.

Corollary 51. *There exists a finite set of subadditive generator functions* $\{F_{\alpha_i}\}_{i=1}^{R}$ *such that*

$$\begin{aligned} & Ax = b \\ & \sum_{j \in I} F_{\alpha_i}(a_j)x_j + \sum_{j \in C} \overline{F_{\alpha_i}}(a_j)x_j \geq F_{\alpha_i}(b) \quad && i = 1, ..., R \\ & x_j \geq 0 && \forall j \in N \end{aligned} \tag{5.3}$$

is the convex hull of solutions to MILP (2.3) with right hand side b.

The above corollary shows that there is a finite representation for MILPs using subadditive functions which is an extension to the works of Wolsey and Bachem and Schrader.

Theorem 46 will be the base of our algorithm in the next sections of this chapter. It is a generalized versions of Klabjan's work and it shows how subadditive generator functions can be used as a certificate of optimality in practice.

5.3 Generating Optimality Certificates and Sensitivity Analysis

In this section we consider 0-1 MILPs only. Suppose that MILP (2.3) is solved to optimality using a branch and bound method. Now we have the tree and the optimal solution available. We have two goals: generate a certificate of optimality and generate a tool for sensitivity analysis. We use the generalized subadditive generator functions for both purposes.

The goal is to generate enough cuts from the branch and bound tree to get the desired α using Theorem 46. The algorithm that we will present in this chapter, extracts cutting planes from different disjunctions in the branch and bound tree. Then these cuts are lifted to become valid for the entire tree. This is explained in Section 5.3.2. However another lifting is needed if we want to use Theorem 46 which is explained in Section 5.3.1. Then the dual vector of the LPR with all the cuts added, is the α that we need. F_α can be used as a certificate of optimality and for sensitivity analysis. The algorithm we will present in Section 5.3.4 iteratively extracts cuts from different levels of the tree.

Definition 52. *A Certificate of Optimality for an MILP is information that can be used to check optimality in a shorter time than solving MILP itself.*

Definition 53. *For MILP (2.3) with optimal solution x^*, α is a certificate of optimality if $F_\alpha(b) = cx^*$. We call $\alpha \in \mathbb{R}^m$ a "good" certificate if $|E| << n$. α^* is called optimal if we have:*

$$\alpha^* \in \underset{\alpha \in \mathbb{R}^m}{\operatorname{argmin}} \{|E(\alpha)| : F_\alpha(b) = z^*_{MILP}\}.$$

Note that if the size of E is much smaller than N, then the certificate that we have is very much easier to check since the number of variables is reduced.

Remark 54. [27] *For MILP (2.3) with optimal solution x^*, α with $F_\alpha(b) = cx^*$ can be used for sensitivity analysis.*

Theorem 46 tells us that if we add a family of cutting planes $\pi^j x \leq \pi_0^j,\ j \in J$ to the LP relaxation of MILP (2.3), such that we have

$$\begin{aligned} z^* = \quad & \min \quad cx \\ & \text{s.t.} \quad Ax = b \\ & \qquad \pi^j x \leq \pi_0^j \quad j \in J \\ & \qquad x \geq 0, \end{aligned} \tag{5.4}$$

then if (α, γ) is an optimal dual vector where γ corresponding to constraints $\pi^j x \leq \pi_0^j,\ j \in J,\ F_\alpha(b) \geq z^*$. In other words, the existence of such α is guaranteed by Theorem 46.

Note that if we add the cut $cx \geq z^*_{MILP}$ to MILP(2.3), then obviously we can use the α calculated using Theorem 46 since for that α, we have $F_\alpha(b) = cx^* = z^*_{MILP}$. However based on our empirical observations through computational experiments, the size of $E(\alpha)$ in this case is usually equal to or comparable to n. We have observed that the more cuts that we add to the LP relaxation of MILP(2.3), the better α we can get using Theorem 46. We have also observed that if we add enough cuts so that the optimal solution of the LP relaxation is x^*, then the size of E is much less than N, so we can consider it as a good certificate.

On the other hand, small E does not necessarily mean that $F_\alpha(b)$ is easier to be evaluated. However again based on our observations, for specific families of problems such as set covering and knapsack problems, it takes relatively short CPU time to calculate $F_\alpha(b)$ whenever α is a good certificate.

We should note that the cutting planes $\pi^j x \leq \pi_0^j,\ j \in J$ must be valid inequalities for the set $\{x \in \mathbb{Z}_+^I \times \mathbb{R}_+^C : Ax \leq b\}$ by Theorem 46. This leads us to the lifting problem: each cutting plane that we add to the LP relaxation of MILP (2.3) must be lifted first to become valid for the set $\{x \in \mathbb{Z}_+^I \times \mathbb{R}_+^C : Ax \leq b\}$.

In the next section, we show that the set $\{(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By = b\} \neq \emptyset$ is a face of $\{(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By \leq b\}^2$. So the problem of lifting is well defined. We also will state an algorithm by Espinoza et al. [22] to do this.

5.3.1 Lifting Valid Inequalities valid for a Facet

Consider a mixed integer set M and a proper face Q of $conv(M)$ defined by $Q = \{x \in conv(M); c^T x = d\}$ where $c^T x \geq d$ is a valid inequality for $conv(M)$. Assume that $a^T x \geq b$ is valid for Q, and that $Q \neq \emptyset$. The lifting problem consists in finding an inequality $\bar{a}^T x \geq \bar{b}$ that is valid for $conv(M)$ and such that $\bar{a}^T x - \bar{b} = a^T x - b$ for all $x \in Q$.

Lifting is equivalent to finding a $\lambda \in \mathbb{R}$ with $(a^T x - b) - \lambda(c^T x - d) \geq 0$ valid for $conv(M)$. Espinoza et al [22] use the following algorithm to solve the mentioned lifting problem. Let $M = \{x \in \mathbb{R}^n : Ax \geq h, x_i \in \mathbb{Z}\ \ \forall i \in I\}$ where $A \in \mathbb{Q}^{m \times n}, h \in \mathbb{Q}^m$ and $I \subseteq \{1, ..., n\}$. Assume that $cx \geq d$ for all $x \in M$. Let $RM = \{r \in \mathbb{R}^n : Ar \geq 0\}$.

The algorithm for this kind of lifting is given in Algorithm 7.

We will now show that the set $\{(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By = b\} \neq \emptyset$ is a face of $\{(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By \leq b\}$ which means that this type of lifting can be used to retrieve α.

Lemma 55. *Let $P = \{x : Ax \leq b\}$ be a polyhedron and $T = \{x : \alpha x \leq \alpha_0\}$ be a supporting hyperplane. Then $P \cap \{x : \alpha x = \alpha_0\}$ is a face of P.*

[2]Note that we prove this for $(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n$. However, the theorem can be proved for $(x, y) \in \mathbb{Z}_+^I \times \mathbb{R}_+^C$ in the same way.

```
Data: A nonempty mixed integer linear set $M$, vectors
      $a, c \in \mathbb{Q}^n$, and scalars $b, d \in \mathbb{Q}$.
Result: The solution $\lambda^*$ to
        $\max\{\lambda : (a^T x - b) - \lambda(c^T x - d) \geq 0, \forall x \in M\}$, or
        proof of infeasibility or unboundedness.
  $i \leftarrow 1$, $z_1 \leftarrow \min\{-(c^T x - d) : x \in M\}$.
  Let $x^1$ be a minimizer or $r^1 \in RM : -c^T r^1 < 0$.
  if $z_1 = 0$ then
      $z \leftarrow \min\{(a^T x - b) : x \in M\}$.
      if $z \geq 0$ then
          Problem is unbounded. Stop.
      end
      Problem is infeasible. Stop.
  end
  if $z_1 = -\infty$ then
      while $z_i < 0$ do
          if $c^T r^i = 0$ then
              Problem is infeasible. Stop.
          end
          $\lambda_{i+1} \leftarrow \frac{a^T r^i}{c^T r^i}, i \leftarrow i + 1$.
          $z_i \leftarrow \min\{(a^T r) - \lambda_i(c^T r) : r \in RM, r \leq 1\}$.
          Let $r^i$ be a minimizer having value $z_i$.
      end
      $z_i \leftarrow \min\{(a^T x - b) - \lambda_i(c^T x - d) : x \in M\}$.
      Let $x^i$ be a minimizer having value $z_i$.
  end
  while $z_i < 0$ do
      if $(c^T x^i - d) = 0$ then
          Problem is infeasible. Stop.
      end
      $\lambda_{i+1} \leftarrow \frac{a^T x^i - b}{c^T x^i - d}, i \leftarrow i + 1$.
      $z_i \leftarrow \min\{(a^T x - b) - \lambda_i(c^T x - d) : x \in M\}$.
      Let $x^i$ be a minimizer having value $z_i$.
  end
  $\lambda^* \leftarrow \lambda_i$.
```

Algorithm 5.1: Lifting Valid Inequalities Using Espinoza's Algorithm

Let $P_1 = \{(x,y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By = b\} \neq \emptyset$ and $P_2 = \{(x,y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By \leq b\}$. Also let $P_1' = conv(P_1)$ and $P_2' = conv(P_2)$. Let $\pi^1 = \mathbb{1}A$ and $\pi^2 = \mathbb{1}B$ and $\pi_0 = \mathbb{1}b$ where $\mathbb{1}$ is vector of ones. Obviously $\pi^1 x + \pi^2 y \leq \pi_0$ is a supporting hyperplane of P_2'. (Let (x^*, y^*) be a feasible point of P_1. This point is on the hyperplane $\pi^1 x + \pi^2 y = \pi_0$ and is also in P_2 and so in P_2'. Also any point in P_2' will satisfy $Ax + By \leq b$ [3] and hence $\pi^1 x + \pi^2 y \leq \pi_0$.)

Lemma 56. *$ext(P_1') \subseteq P_2' \cap \{(x,y) : \pi^1 x + \pi^2 y = \pi_0\}$ where ext is the set of extreme points.*

Proof. Suppose that $(x,y) \in ext(P_1')$. This means that $(x,y) \in P_1'$ and x is integral. So we have $Ax + By = b$. Now we conclude that $\pi^1 x + \pi^2 y = \pi_0$ and $Ax + By \leq b$. Since x is integral, $(x,y) \in P_2$ and so in P_2' and we are done. $\square$

Lemma 57. *$ext(P_2') \cap \{(x,y) : \pi^1 x + \pi^2 y = \pi_0\} \subseteq P_1'$.*

Proof. Let (x,y) be an extreme point of P_2' with $\pi^1 x + \pi^2 y = \pi_0$. Since $(x,y) \in P_2'$, we have $Ax + By \leq b$ and also x is integral. Since $\pi^1 x + \pi^2 y = \pi_0$ and $Ax + By \leq b$, we conclude that $Ax + By = b$[4]. Since x is integral, $(x,y) \in P_1$ and so $(x,y) \in P_1'$. $\square$

Theorem 58. *Let $P_1 = \{(x,y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By = b\} \neq \emptyset$ and $P_2 = \{(x,y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By \leq b\}$ and A, B and b have rational entries. Then $conv(P_1)$ is a face of $conv(P_2)$.*

Proof. From Theorem 42 we know that

$$rec(P_1) = rec(P_1') \text{ and } rec(P_2) = rec(P_2').$$

[3]Note that $P_1 \subseteq P_1' \subseteq \{(x,y) \in \mathbb{R}_+^{n+p} : Ax + By = b\}$ and $P_2 \subseteq P_2' \subseteq \{(x,y) \in \mathbb{R}_+^{n+p} : Ax + By \leq b\}$

[4]We can take proper combinations of the rows of the two systems.

Also it is obvious that since $P_1 \subseteq P_2$, $rec(P_1) \subseteq rec(P_2)$ so

$$rec(P_1') \subseteq rec(P_2'). \tag{5.5}$$

Moreover we know that

$$rec(P_1) = \{(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By = 0\}$$

and

$$rec(P_2) = \{(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^n : Ax + By \leq 0\}.$$

Now let $(x, y) \in P_1'$. Then by theorem of Minkowski and Weyl (see [41] page 88), we can write

$$(x, y) = \sum \lambda_i(x_i, y_i) + \sum \mu_i(z_i, w_i)$$

where $\sum \lambda_i = 1$, (x_i, y_i) are extreme points and (z_i, w_i) are extreme rays of P_1'. By Lemma 56, $(x_i, y_i) \in P_2' \cap \{(x, y) : \pi^1 x + \pi^2 y = \pi_0\}$. Also we have:

$$\pi^1 x + \pi^2 y = \sum \lambda_i(\pi^1 x_i + \pi^2 y_i) + \sum \mu_i(\pi^1 z_i + \pi^2 w_i).$$

But $\pi^1 z_i + \pi^2 w_i = 0$ since (z_i, w_i) are extreme rays of P_1' and $\pi^1 = \mathbb{1}A$ and $\pi^2 = \mathbb{1}B$ and $\sum \lambda_i(\pi^1 x_i + \pi^2 y_i) = \sum \lambda_i \pi_0 = \pi_0$. This shows that $\pi^1 x + \pi^2 y = \pi_0$. Now using (5.5) and the fact that P_2' and $\{(x, y) : \pi^1 x + \pi^2 y = \pi_0\}$ are both convex sets, we can conclude that $(x, y) \in P_2' \cap \{(x, y) : \pi^1 x + \pi^2 y = \pi_0\}$.

Conversely let $(x, y) \in P_2' \cap \{(x, y) : \pi^1 x + \pi^2 y = \pi_0\}$. We can write

$$(x, y) = \sum \lambda_i(x_i, y_i) + \sum \mu_i(z_i, w_i)$$

where $\sum \lambda_i = 1$, (x_i, y_i) are extreme points and (z_i, w_i) are extreme rays of P'_2. By Lemma 55 $(x_i, y_i) \in \{(x, y) : \pi^1 x + \pi^2 y = \pi_0\}$ and by Lemma 57 $(x_i, y_i) \in P'_1$.

Claim 59. *$(z_i, w_i) \in rec(P'_1)$ for every i.*

Proof of the claim: We know that $(z_i, w_i) \in rec(P'_2)$ so we have $Az_i + Bw_i \le 0$. Also we know that

$$\pi^1 x + \pi^2 y = \sum \lambda_i(\pi^1 x_i + \pi^2 y_i) + \sum \mu_i(\pi^1 z_i + \pi^2 w_i) = \pi_0.$$

But we know that $\sum \lambda_i(\pi^1 x_i + \pi^2 y_i) = \pi_0$. So we must have $\sum \mu_i(\pi^1 z_i + \pi^2 w_i) = 0$. Since $\mu_i \ge 0$ and $\pi^1 z_i + \pi^2 w_i \le 0$, we conclude that $\pi^1 z_i + \pi^2 w_i = 0$ for all i and so $(z_i, w_i) \in rec(P'_1)$ for every i.

Now By Claim 59 and the fact that P'_1 is a convex set, we get the desired result.

□

In order to generate optimality certificates and sensitivity analysis tools, we will extract cutting planes from the branch and bound tree. However, these cutting planes are valid for a specific node inside the tree not for the entire tree i.e. MILP (2.3). So we have to lift these cuts to become valid for the entire tree. In the next section, we will explain a method by Balas, Ceria and Cornuéjols from [5] for this purpose.

5.3.2 Lifting Inequalities from a Node in the Branch and Bound Tree

Consider MILP (2.3). Obviously if we generate a cutting plane for a subproblem of MILP (2.3), it wouldn't be valid for the original problem since we have projected the problem into a lower dimension space. However, these cuts can be lifted to become valid for the original problem as follows:

Suppose that we have branched on several binary variables in order to solve the problem in a branch and bound framework. Let $\bar{x}$ be the optimal solution for the node $\mathcal{N}$ with $0 < \bar{x}_j < 1$ for some j. We are now branching on x_j. Let (α^R, β) be an optimal solution for the following LP where $R \supseteq \{i \in I : 0 < \bar{x}_i < 1\} \cup \{i \in N : \bar{x}_i > 0\}$.

$$\begin{array}{lll}
\max & \beta - \alpha^R \bar{x}^R & \\
\text{s.t.} & \alpha^R - u^R A^R + u_0 e_j^R & \geq 0 \\
& \alpha^R - v^R A^R - v_0 e_j^R & \geq 0 \\
& u^R b^R & \geq \beta \\
& v^R b^R + v_0 & \geq \beta \\
& u^R, v^R \geq 0. &
\end{array}$$

Theorem 60. [5] *(α^R, β) is a valid inequality for the node $\mathcal{N}$ and can be lifted to become valid for MILP (2.3) with*

$$\alpha_i = \begin{cases} \alpha_i^R & i \in R \\ \max\{\alpha_i^1, \alpha_i^2\} & i \notin R \end{cases}$$

where $\alpha_i^1 = u^R A_i^R$ and $\alpha_i^2 = v^R A_i^R$ where A_i^R is a subvector of A_i (i-th column of A) with the same row set as A^R (It is assumed that A subsumes the inequalities $x_j \leq 1$ and $x_j \geq 0$ for all j, so restricting columns to R will also remove some rows of A as well). The lifted valid inequality (α, β) cuts off $\bar{x}$ from MILP(2.3).

5.3.3 From Branch and Bound to Cutting Planes

We consider the simplest form of the branch and bound algorithm here with depth first search. LP relaxation of IP (2.1) is solved. In the optimal solution, if x_j is fractional then two subproblems are created: one with $x_j \geq 1$ and one with $x_j \leq 0$. Fathoming can happen in three different cases:

1. fathoming by bound: when the optimal objective value of a node is greater than the best lower bound;
2. fathoming by integer: when the optimal solution of a node is integral;
3. fathoming by infeasibility: when the subproblem corresponding to a node is infeasible.

At the beginning, the best lower bound is set to infinity. Later it will be updated when a node is fathomed by integer. We don't use any specific branching techniques here. However this is something that could reduce the size of the tree.

One question that might rise here is why the branch and bound tree is being used for the purpose of obtaining a good α. The reason is that we assume that we have already solved the problem using a branch and bound method and we do not want to optimize from scratch. We have enough information at hand that can guide us to a reasonably good certificate (not necessarily optimal α) and we can extract the cuts used in Theorem 46 from that tree as we will show below.

Lemma 61. [38] *Let P denote the set of feasible points to the LP relaxation of MILP (2.3). Let*

$$P_j = (P \cap \{x : x_j \leq 0\}) \cup (P \cap \{x : x_j \geq 1\})$$

and $\bar{x}$ be a vertex of $conv(P_j)$ with $0 < x_j < 1$. Then there is a cutting plane (α, β)

valid for conv(P_j) that cuts off $\bar{x}$. This cut can be generated using the following LP:

$$\begin{aligned}
min\quad & \alpha\bar{x} - \alpha_0 \\
s.t.\quad & \alpha - uA + u_0 e_j \geq 0 \\
& \alpha - vA - v_0 e_j \geq 0 \\
& \alpha_0 - ub \leq 0 \\
& \alpha_0 - vb - v_0 \leq 0 \\
& (\alpha, \beta, u, v, u_0, v_0) \in S \\
& u, u_0, v, v_0 \geq 0
\end{aligned} \tag{5.6}$$

where S is a normalization. Any inequality valid for the conv(P_j) is feasible to LP (5.6). So any facet of conv(P_j) corresponds to a basic solution of LP (5.6). (See Figure 5.1.)

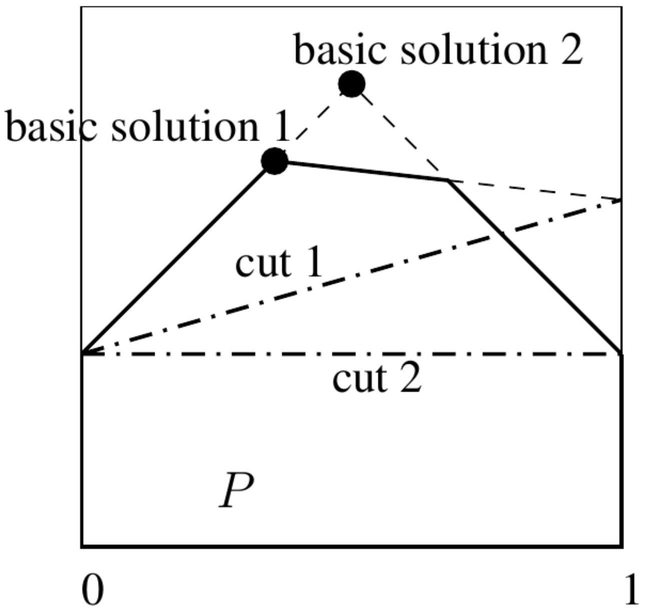


Figure 5.1: Correspondence of disjunctive cuts with basic solutions of LP (5.6)

In [6] Perregaard shows that these cuts can be obtained from the simplex tableau using a strengthening procedure. Using disjunctive cuts is a standard method to solve MILP problems and these cuts are frequently used in a branch and cut framework for mixed integer optimization [5].

Now we will state and prove a theorem that will form the basis of our algorithm. However, first we have to make two assumptions:

1. We will assume that the branch and bound algorithm will fathom a node by bound only if the optimal objective value of the corresponding node is strictly greater than the best lower bound available. If we get an objective value equal to the best available lower bound with a fractional solution, we will continue branching. (As if we are trying to find all integer optimal solutions)
2. We will assume that the LP corresponding to each fathomed node has a unique optimal solution.

Theorem 62. *Suppose that MILP (2.3) is solved to optimality using a branch and bound method. Given the branch and bound tree and an optimal solution x^*, one can extract a family $\mathcal{F}$ of valid inequalities such that optimal solution to*

$$\begin{array}{lll} min & cx & \\ s.t. & Ax = b & \\ & \alpha x \leq \beta & \text{for all } (\alpha, \beta) \in \mathcal{F} \\ & cx \geq z^*_{MILP} & \\ & x \geq 0 & \end{array} \tag{5.7}$$

is integral for indices corresponding to I.

Proof. We will prove this theorem in an algorithmic way.

First add the cutting plane $cx \geq cx^*$ to the LP relaxation of the original problem. Now solve this problem and find the optimal solution $\bar{x}$. If $\bar{x} = x^*$ or any other optimal solution (i.e. if $\bar{x}$ is integral for indices in I), then there is nothing to do. Otherwise locate this point $\bar{x}$ in the tree i.e. find a disjunction (branch) that this point falls in it. To be rigorous, lets assume that we show a middle node of the tree with n_i and the index of the variable that the branch and bound algorithm branches on within this node with b_{n_i}. Let r_0 denote the root and l_i be a leaf node in the tree. Then we can define a sequence in the branch and bound tree such as $b_{r_0}, b_{n_1}, b_{n_2}, ..., b_{n_k}$ which shows the sequence of indices branched on until we get to the leaf node l_i. By locating $\bar{x}$ inside the tree we mean finding a sequence inside the tree that coincides with the indices of integer components of $\bar{x}$. Now lets assume that $\bar{x}_j$ is fractional and we have found a sequence $s = b_{r_0}, b_{n_1}, b_{n_2}, ..., b_{n_k}$ with $b_{n_k} = j$ and $x_{b_{r_0}}, x_{b_{n_1}}, ..., x_{b_{n_{k-1}}}$ are all 0 or 1 based on the sequence. Then by Lemma 61, there exists a cutting plane $\alpha x \geq \beta$ that cuts off $\bar{x}$. Extract this cut either by solving the cut generating LP (5.6)[5] or using the simplex tableau and strengthening procedure from [6]. However this cut may not be valid for the entire tree. This is no problem since we can lift this cut to become valid for the tree (MILP(2.3)) in such a way that it still cuts off $\bar{x}$ after lifting using the method described in Section 5.3.2. Add (α, β) to $\mathcal{F}$ and continue this process until $\bar{x} = x^*$ or any optimal solution. Now we will show that we can always find such a sequence otherwise $\bar{x}_i$ is integral for $i \in I$.

Assume on contrary that we cannot locate $\bar{x}$ inside the tree where $\bar{x}$ is not integral for some indices in I. In this case there are four possibilities:

1. Let l_i be a leaf node fathomed by bound with the sequence $b_{r_0}, b_{n_1}, b_{n_2}, ..., b_{n_i}$. $\bar{x}$ is such that $x_{b_{r_0}}, x_{b_{n_1}}, ..., x_{b_{n_k}}$ are all integral and $0 < x_k < 1$ for some k not in $\{b_{r_0}, b_{n_1}, b_{n_2}, ..., b_{n_i}\}$. In this case we get a contradiction since obviously

[5]Optimal bases of the branch and bound nodes can be saved to be accessed later to extract cuts.

$c\bar{x} = cx^*$. (Because x^* is a feasible point to the LP relaxation and $c\bar{x}$ cannot be greater than cx^* since otherwise the LP would return x^* instead of $\bar{x}$ as optimal solution.) The tree wouldn't have stopped and fathomed this node since we have assumed that fathoming by bound only happens in the case of strict inequality. So this case is impossible.

2. $\bar{x}$ falls exactly inside one of the nodes fathomed by bound. This is impossible since we have assumed that each fathomed node has a unique solution.

3. Let l_i be a leaf node fathomed by integer with the sequence $b_{r_0}, b_{n_1}, b_{n_2}, ..., b_{n_i}$. $\bar{x}$ is such that $x_{b_{r_0}}, x_{b_{n_1}}, ..., x_{b_{n_k}}$ are all integral and $0 < x_k < 1$ for some k not in $\{b_{r_0}, b_{n_1}, b_{n_2}, ..., b_{n_i}\}$. In this case we get a contradiction since again $c\bar{x} = cx^*$ and this means that the node has multiple optimal solutions which is a contradiction. So this case is impossible as well.

4. $\bar{x}$ falls inside one of the nodes fathomed by integer. If $\bar{x}$ falls inside such a node and it is integral for indices in I, we obviously have more than one optimal solution which will cause the algorithm to terminate or otherwise the node has multiple optimal solutions which is a contradiction.

This show that there is always a disjunctive cut that cuts off $\bar{x}$. Now we have to show that this process will terminate in a finite number of iterations. This is very easy since we know that the branch and bound tree is finite[6] and also we know by Lemma 61 that there are finitely many cuts to describe the convex hull of each disjunction in the tree (See [4] for a theorem by Balas where he proves that $K_n(K_{n-1}(...(K_2(K_1)))) = conv(P \cap (\ \{0,1\}^I_+ \times \mathbb{R}^C_+)$ where $K_j = conv(P \cap \{x : x_j \leq 0\}) \cup (P \cap \{x : x_j \geq 1\}$.) So the total procedure will terminate in finitely many iterations and at some point the

[6]For 0-1 MILP we know that the tree would be finite. However for general MILP, there is no such result available that shows that it can be solved using a finite tree. This is why we concentrate on 0-1 MILP in this chapter.

optimal solution of the LP (5.7) will be equal to x^* or integral for indices $i \in I$.

This completes the proof. $\square$

Although the proof is stated algorithmically, we don't use exactly the same procedure in our algorithm. The reason is that in real world problems, usually one cut from each disjunction in the path from root to a node corresponding to an optimal solution in the tree suffices to get the desired integral optimal solution.

Note that if we drop assumption 2, we may need to use Gomory cuts within the nodes with multiple optimal solutions and then lift them to become valid for the tree and also lift them again to become usable in Theorem 46. However, this doesn't happen very often. In most of the instances we used for testing, there was no need for extra Gomory cuts or any other types of cutting planes.

We have observed in our experiments that if we lift all the valid inequalities of the LP relaxation which are active at x^* within the corresponding node in the tree, and then lift these valid inequalities, they form a very useful family of cutting planes and help the algorithm to terminate earlier.

Note that the cutting plane $cx \geq z^*_{MILP}$ is necessary in order to have finite termination. Without the use of this valid inequality, there might be a gap after adding all valid inequalities from all disjunctions in the tree. In this case again we may need to use different types of cutting planes to close this gap.

Example 63. *Consider the following IP:*

$$\begin{aligned} min \quad & 5x_1 + x_2 + x_3 + 4x_4 + 4x_5 \\ s.t. \quad & x_1 + x_2 + x_3 + x_5 \geq 1 \\ & x_4 + x_5 \geq 1 \\ & x_1 + x_2 + x_3 + x_4 \geq 1 \\ & x \in \{0,1\}^5. \end{aligned}$$

Optimal solution to this IP is $x^* = [0, 0, 1, 0, 1]$ with $z^*_{MILP} = 5$. Optimal solution to the LP relaxation is $[0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$ with objective value 4.5. The branch and bound algorithm will find the optimal solution to this problem by branching on x_3 and then x_2 with two disjunctions. If we add all cutting planes to describe the convex hulls of the two disjunctions in the tree, we will get the optimal solution $[\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, 0]$ with objective value 4. But if $cx \geq 5$ is added we get x^* as optimal solution. Note that adding this cut alone is also not enough where we get $x = [0, 0, \frac{3}{7}, \frac{4}{7}, \frac{4}{7}]$.

5.3.4 Our Algorithm to Generate α

Before we explain our algorithm, we will introduce a new family of cutting planes which can be extracted from the branch and bound tree.

It is possible that during the process of solving MILP (2.3) using branch and bound, some nodes of the tree are infeasible. In this case, we can use logical inference to generate cuts. These cuts will be very deep cuts and will help us get the desired α faster.

Definition 64. *Let n_k be a node in a branch and bound tree for solving MILP (2.3) and b_k be its branching variable. Let n_{k_0} and n_{k_1} be the left and right children respectively. Also assume that n_{k_1} is an infeasible node. Let R be the set of indices of variables in MILP (2.3) which has been set to zero in n_k and L be the set of indices of variables set to one. The infeasibility cut associated with n_k is*

$$\sum_{i \in R} x_i + \sum_{i \in L} (1 - x_i) - x_{b_i} \geq 0.$$

If n_{k_0} is infeasible, then the infeasibility cut is

$$\sum_{i \in R} x_i + \sum_{i \in L} (1 - x_i) + x_{b_i} \geq 1.$$

Theorem 65. *Infeasibility cuts are valid inequalities for MILP (2.3).*

Proof. Type 1 infeasibility cut can be written as

$$x_{b_i} \leq \sum_{i \in R} x_i + \sum_{i \in L} (1 - x_i)$$

which means that if $x_i = 0$ for all $i \in R$ and $x_i = 1$ for all $x \in L$, then x_{b_i} must be zero. Otherwise it is a redundant valid inequality of the form $x_{b_i} \leq 1$.

Type 2 infeasibility cut can be written as

$$1 - x_{b_i} \leq \sum_{i \in R} x_i + \sum_{i \in L} (1 - x_i)$$

which means that if $x_i = 0$ for all $i \in R$ and $x_i = 1$ for all $x \in L$, then x_{b_i} must be equal to one. Otherwise it is a redundant valid inequality of the form $x_{b_i} \geq 0$. □

Now we describe our algorithm. We denote a node by n_k, and its branching variable index with b_k. This algorithm uses the information from the branch and bound tree and also the optimal solution x^* found. First it adds the cutting plane $cx \geq cx^*$. Then from each disjunction (branch) that has both children feasible, it extracts disjunctive cuts to cut off the fractional solution of the current node. Any cut generated in this way needs to be lifted to become valid for the entire tree i.e. for the MILP. This is done using the method described in section 5.3.2. These cuts are added to LP relaxation and this process is repeated until the optimal solution of the LP relaxation is equal to the optimal solution of the MILP. Now we are ready to use Theorem 46 to generate α. But first, we have to lift all these generated cutting planes to become valid for the set $\{x \in \mathbb{Z}_+^I \times \mathbb{R}_+^C : Ax \leq b\}$. This is done using the method explained in section 5.3.1. Then the dual vector of the LP will give us the α.

Our algorithm is stated in Algorithm 5.2.

```
Data: Mixed integer linear program
      $\min\{cx : Ax = b, x \geq 0, x_i \in \{0,1\}$ for $i \in I\}$, Branch
      and bound tree that solves the MILP and an optimal
      solution $x^*$.
Result: A family of cutting planes $\mathcal{F}$.
    Find the leaf node corresponding to optimal solution of
    the MILP (2.3). If there are more than one nodes like
    that, choose the one that has the lowest level in the tree.
    Denote it by $n_0$ and its parent node by $n_1$. Let $i = 1$ and
    $\mathcal{F} = \emptyset$.
    while $n_i$ is not the root do
        if one of the child nodes of $n_i$ has infeasible status
        then
            Generate infeasibility cut for the node and add it
            to $\mathcal{F}$.
        end
        Within the node $n_i$, extract one disjunctive cut using
        LP (5.6) valid for the convex hull of the disjunction
        that cuts off the $\bar{x}_i$ the fractional optimal solution of
        the disjunction. Lift the valid inequality in to become
        valid for MILP (2.3). Lift it again to become valid for
        $\{x \in \mathbb{Z}_+^I \times \mathbb{R}_+^C : Ax \leq b\}$ and add it to $\mathcal{F}$.
        If $n_i$ is the root node of the tree, stop. Else $i = i + 1$
        and let $n_i$ be the parent node of $n_{i-1}$.
    end
    Lift and add the valid inequality $cx \geq cx^*$ to $\mathcal{F}$.
    if $\bar{x}$, the solution of LP relaxation with all cuts in $\mathcal{F}$
    added to it, is still not equal to $x^*$ or any other optimal
    solution then
        while $x \neq x^*$ or any other optimal solution do
            Find the branch in the tree that $\bar{x}$ falls in that
            disjunction. Find the disjunctive cut that cuts off
            $\bar{x}$. Lift it and add to $\mathcal{F}$.
        end
    end
```

Algorithm 5.2: Our Algorithm for Finding α.

Example 66. *Consider the following set covering problem:*

$$\begin{aligned}
min \quad & 2x_1 + 3x_2 + 2x_3 + 3x_4 \\
s.t. \quad & x_1 + x_2 + x_4 \geq 1 \\
& x_1 + x_3 + x_4 \geq 1 \\
& x_2 + x_3 \geq 1 \\
& x \in \{0,1\}^4.
\end{aligned}$$

The optimal solution of the problem is $x^* = [0,1,0,1]$ with $z^* = 4$. The LPR solution is $x = [\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0]$ with $\bar{z} = 3\frac{1}{2}$. Figure 5.2 shows the branch and bound tree that solves this problem.

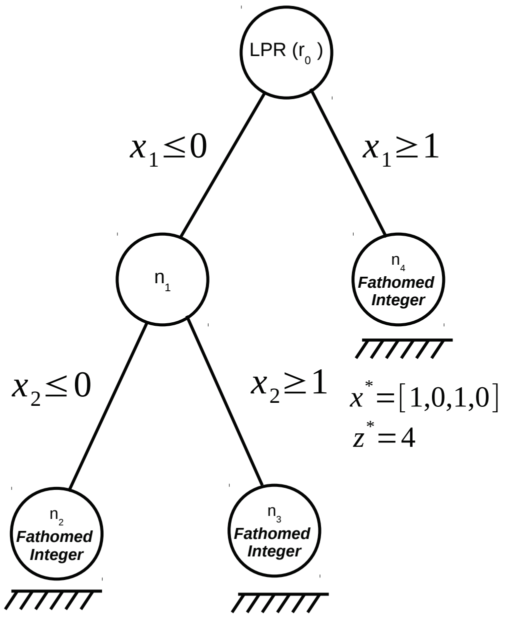


Figure 5.2: Branch and Bound Tree for Example 66

The first cut $x_1 + x_2 + x_3 + x_4 \geq 2$ is extracted from the first disjunction cutting off the LP relaxation solution and the second cut $x_1 + x_2 + x_3 + 2x_4 \geq 2$ is extracted from the second disjunction cutting off the solution of the node n_1 namely $\bar{x}_1 = [0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$. Also a cut of the form $cx \geq cx^*$ is added as needed in the algorithm i.e. $2x_1 + 3x_2 + 2x_3 + 3x_4 \geq 4$. With all these cuts added, the solution of the LP relaxation is the same as the optimal solution. □

This method can be combined with branch and cut to get a certificate faster. The bases in the tree nodes can be saved and accessed in each iteration to save time and computational expense.

We have tried to use the branch and bound tree and x^* as our resources to generate valid inequalities. However, if the original MILP has been solved using a cutting plane method or a combination of cutting planes and branching algorithm, we still can use this method to generate the desired α. Most commercial solvers solve MILPs using branch and cut algorithm but they won't release the cutting planes and/or branching information to the user. That's why we are trying to rely on our own solvers. There is no restriction in adding any type of cutting plane at any stage of the algorithm. This actually will help the algorithm terminate earlier. However, we have focused on branching for the reasons mentioned.

5.4 Computational Experiments

Our computational experiments show that in most 0-1 mixed integer programs, the size of E (see Section 5.1 for definitions) is significantly smaller than N. In lower dimensions size of E is usually about 50% of the size of N, but when $|N|$ is large, this ratio will decrease to 25% in average and even less depending on the problem type. In the best case, we had $|E|/|N| = 0.01$.

Also the results are even better when we are working with non-negative entries i.e. when A and b in MILP(2.3) are non-negative. This is obvious since all columns with at least one negative element should be put in E. However if we have a problem with lots of negative entries, we can multiply rows of A by -1 to get a better structure of the problem.

We have used two families of mixed integer programming problems for our numerical experiments: knapsack and set covering problems. A knapsack problem is an optimization problem of the form

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_i x_i \leq b \\ & x \in \{0,1\}^n. \end{aligned}$$

Also if we have continuous variables in the problem, then it is called a mixed integer knapsack problem. A set covering problem is of the form

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax \geq \mathbf{1} \\ & x \in \{0,1\}^n. \end{aligned}$$

where $\mathbf{1}$ is a vector of ones and elements of A are in $\{0,1\}$. All the instances were generated randomly. However, all the problems are transformed into form of IP (2.1) for experiments. In the instances that we generate, elements of c in both families vary between -10 and 10 and a_i and b in knapsack problems vary between 0 and 10 for smaller problems (less than 200 variables). For large size problems the size of elements increase accordingly.

The programs for the computational experiments were written in C++ and Python containing approximately 2,000 lines of code. The best results are from (mixed) integer knapsack problems. In the next sections, we represent our numerical experiments for each family of problems.

All the instances used for combinational experiments are available in MPS or LP format on `http://people.math.carleton.ca/∼bmoazzez/Thesis.html` along with the generated certificates.

5.4.1 Pure Integer Knapsack Problems with Non-negative Coefficients

For this family of problems, we get the best results. We have several examples with 5000 variables with $|E|/|N|$ less than 0.04.

Problem Name	Number of Variables	Size of E	Percentage E/N
ipknapos1.mps	5	2	40
ipknapos2.mps	20	18	90
ipknapos3.mps	100	46	46
ipknapos4.mps	100	19	19
ipknapos5.mps	500	7	1.4
ipknapos6.mps	1000	43	4.3
ipknapos7.mps	5000	178	3.65

Table 5.1: Pure Integer Knapsack Problems with Non-negative Coefficients

5.4.2 Mixed Integer Knapsack Problems with Non-negative Coefficients

Our results here are as good as the previous part. However, the best results still belong to the pure integer case.

Problem Name	Number of Variables	Size of E	Percentage E/N
mipknapos1.mps	50	6	12
mipknapos2.mps	100	8	8
mipknapos3.mps	200	22	11
mipknapos4.mps	500	50	10
mipknapos5.mps	1000	104	10.4
mipknapos6.mps	5000	362	7.24

Table 5.2: Mixed Integer Knapsack Problems with Non-negative Coefficients

5.4.3 Pure Integer Knapsack Problems

When we allow entries of the original matrix A be any number, we have to include in E all columns with at least one negative element. (In case of knapsack problems, a column has only one element.) This will increase the size of E compared to non-negative entries case.

Problem Name	Number of Variables	Size of E	Percentage E/N
ipkn1.mps	20	14	70
ipkn2.mps	20	12	60
ipkn3.mps	100	36	36
ipkn4.mps	100	42	42
ipkn5.mps	500	213	41.6
ipkn6.mps	1000	449	44.9
ipkn7.mps	5000	2175	43.5

Table 5.3: Pure Integer Knapsack Problems

5.4.4 Mixed Integer Knapsack Problems

Problem Name	Number of Variables	Size of E	Percentage E/N
mipkn1.mps	50	27	57
mipkn3.mps	100	51	51
mipkn4.mps	200	94	47
mipkn5.mps	500	242	48.4
mipkn6.mps	1000	462	46.2
mipkn7.mps	2000	941	47.05

Table 5.4: Mixed Integer Knapsack Problems

5.4.5 Set Covering Problems

The results for set covering problems are very satisfying. However, there are examples where adding all the cuts doesn't work. (See the example with * in the following table) But in average, for set covering problems, we are able to generate good certificates.

Problem Name	Number of Variables	Size of E	Percentage E/N
scp1.mps	10	3	30
scp2.mps	20	5	25
scp3.mps	30	20	66.6
scp4.mps	40	8	20
scp5.mps	50	25	50
scp6.mps	60	7	11.6
scp7.mps	70	17	24.3
scp8.mps	80	45	56.25
scp9.mps	90	51	56.6
scp10.mps	100	60	60
scp11.mps	110	84	76.36
scp12.mps	150	43	28.66
scp12-1.mps*	150	150	100
scp13.mps	200	94	47
scp14.mps	250	17	6.8
scp15.mps	300	7	2.3
scp16.mps	500	12	2.4
scp17.mps	750	339	45.2

Table 5.5: Set Covering Problems

Chapter 6

Future Research Directions

In this chapter we will discuss some problems that are still unanswered and are likely to provide a good basis for future research.

In Chapter 3, we corrected an algorithm given by Burdet and Johnson for solving integer programming problems using group theoretic approach and subadditivity. One possibility is to generalize this method to mixed integer programming problems. However we strongly believe that the generalized method would not be very efficient due to the results that we get from the corrected version of the algorithm for pure IPs. Especially when the MILP or IP has a right hand side with large elements. In this case, since the feasible region is very large, or at least the candidate set will get very large, it will take a long time for solving these problems using this algorithm. We mentioned before that this algorithm might be useful for binary optimization problems. However, since the search is not directed by any rules in these types of algorithms, they cannot compete with other methods such as cutting plane algorithm or branch and cut.

In Chapter 4, we corrected Klabjan's algorithm and proved that the corrected version works for general integer programming. In spite of Klabjan's numerical reports on different classes of problems, we still believe that this method is not comparable to other standard methods of solving pure IPs. However this method might be useful in the case of specific families of optimization problems such as set covering or set partitioning problems. Klabjan has reported computational results for set partitioning problems where his method is comparable to IBM ILOG CPLEX Optimizer. But he has also mentioned that he has used lots of preprocessing before starting to solve the problems using this algorithm. It is not clear if his method is still comparable to CPLEX without the preprocessing. Although it takes a long time to solve such problems using Klabjan's method, the method is still useful for generating certificates and sensitivity analysis.

An important point that should be mentioned here is that although it may take a long time (maybe more than solving the original problem) to find the optimal α, it is still worth it to find α. The reason is that it doesn't matter how long it takes to generate the certificate. What matters is how long it will take to check the certificate for optimality and we have shown in Chapter 5 that in some cases it is really beneficial to find the α since the number of variables in a problem will reduce to 1%!

In Chapter 5, we have generalized Wolsey's result from [45]. However Wolsey has stronger results in the same article. Wolsey shows that for a pure IP, the convex hull of the set of feasible points can be described using valid inequalities corresponding to subadditive dual feasible functions. However Wolsey shows this for all b i.e. he finds a family of subadditive functions that their corresponding valid inequalities describe the convex hull of the feasible set for all b. In our extension however, b is fixed. Is it possible to find a family of valid inequalities from subadditive functions to describe the convex hull of feasible set of an MILP for all b? I believe that it is possible.

Conjecture 67. *Consider the set $P = \{x \in \mathbb{R}^n_+ : Ax = b\}$ with A rational. There exist a finite number of subadditive generator functions $F_{\alpha_1}, F_{\alpha_2}, ..., F_{\alpha_k}$ such that for* *any* $b \in \mathbb{Q}^m$*:*

$$conv(P \cap \mathbb{Z}^I_+) = P \cap \{x \in \mathbb{R}^n_+ : \sum_{j \in I} F_{\alpha_i}(a_j)x_j + \sum_{j \in C} \overline{F_{\alpha_i}}(a_j)x_j \geq F_{\alpha_i}(b) \text{ for } i = 1, ..., k\}$$

In Chapter 5 we have defined the optimal α certificate. However one question is still unanswered: How can we find optimal α? It is obvious that optimal α is not unique. For example consider the following IP:

$$\begin{aligned} \min \quad & 6x_1 + 6x_2 + 9x_3 + 10x_4 + 7x_5 \\ \text{s.t.} \quad & x_4 + x_5 \geq 1 \\ & x_1 + x_2 + x_3 + x_4 \geq 1 \\ & x_1 + x_5 \geq 1 \\ & x \in \{0, 1\}^5. \end{aligned}$$

Any $[\alpha_1, \alpha_2, \alpha_3]^T$ with $\alpha_1 = 7$ and $\alpha_2 = 6$ and $\alpha_3 \in [3, 7]$ is optimal with $E = \{1, 4, 5\}$. Our observations show that if we extract all cutting planes necessary to describe x^*, we will get a good certificate. However we still do not know if this α is optimal. After finding a good α, we can improve it in several ways. Use of a greedy algorithm for finding optimal α is of great interest if it is less expensive than generating all cutting planes needed. Klabjan uses an algorithm for improving the found α. But note that this is not the same thing. That algorithm is trying to make α a certificate not necessarily optimal. I think that Klabjan's second algorithm can be generalized to find the optimal α.

Another question is when is optimal α unique? i.e. for which classes of problems could we find a unique optimal α?

For a mixed integer set M and a proper face Q of $conv(M)$ defined by $Q = \{x \in conv(M); c^T x = d\}$ where $c^T x = d$ is a valid inequality for $conv(M)$, assume that

$a^T x \geq b$ is valid for Q, and that $Q \neq \emptyset$.

Lifting $ax \geq b$ valid for Q to become valid for M is equivalent to solving the following MILP:

$$\max\{\lambda : (a^T x - b) - \lambda(c^T x - d) \geq 0, \forall x \in M\}.$$

As we have seen in Chapter 5, this is a very important part of our algorithm. In our computations, we recognized that for all cutting planes that we extract from branch and bound tree, $\lambda = -1$ if the cut is scaled in a specific way i.e. if we scale the cutting plane (α, β) to have integer components and $\gcd(\alpha_1, ..., \alpha_n, \beta) = 1$. This is definitely something that can be investigated more and if we can prove this, it will reduce the size of work in our algorithm.

Testing other families of problems and finding certificates for them is another direction for the future research.

Bibliography

[1] David L. Applegate, Robert E. Bixby, Vašek Chvátal, William Cook, Daniel G. Espinoza, Marcos Goycoolea, and Keld Helsgaun, *Certification of an optimal TSP tour through 85,900 cities*, Oper. Res. Lett. **37** (2009), no. 1, 11–15.

[2] Achim Bachem and Rainer Schrader, *Minimal inequalities and subadditive duality*, SIAM J. Control Optim. **18** (1980), no. 4, 437–443.

[3] Egon Balas, *A note on the group theoretic approach to integer programming and the* $0-1$ *case*, Operations Res. **21** (1973), 321–322. Mathematical programming and its applications.

[4] ———, *Disjunctive programming: properties of the convex hull of feasible points*, Discrete Appl. Math. **89** (1998), no. 1-3, 3–44.

[5] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols, *Mixed 0-1 Programming by Lift-and-Project in a Branch-and-Cut Framework*, Management Science **42** (1996), no. 9, 1229–1246.

[6] Egon Balas and Michael Perregaard, *A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming*, Math. Program. **94** (2003), no. 2-3, Ser. B, 221–245. The Aussois 2000 Workshop in Combinatorial Optimization.

[7] Amitabh Basu, Robert Hildebrand, and Matthias Köppe, *The Triangle Closure is a Polyhedron*, arXiv:1111.1780.

[8] ———, *Equivariant Perturbation in Gomory and Johnson's Infinite Group Problem. I. The One-Dimensional Case*, arXiv:1206.2079.

[9] ———, *Equivariant Perturbation in Gomory and Johnson's Infinite Group Problem. II. The Unimodular Two-Dimensional Case*, arXiv:1210.6732.

[10] Amitabh Basu, Robert Hildebrand, Matthias Köppe, and Marco Molinaro, *A $(k+1)$-slope theorem for the k-dimensional infinite group relaxation*, SIAM J. Optim. **23** (2013), no. 2, 1021–1040.

[11] Amitabh Basu, Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli, *Minimal inequalities for an infinite relaxation of integer programs*, SIAM J. Discrete Math. **24** (2010), no. 1, 158–168.

[12] David E. Bell and Jeremy F. Shapiro, *A convergent duality theory for integer programming*, Operations Res. **25** (1977), no. 3, 419–434.

[13] C. E. Blair and R. G. Jeroslow, *The value function of a mixed integer program. I*, Discrete Math. **19** (1977), no. 2, 121–138.

[14] Claude-Alain Burdet, *Enumerative cuts. I*, Operations Res. **21** (1973), 61–89. Mathematical programming and its applications.

[15] Claude-Alain Burdet and Ellis L. Johnson, *A subadditive approach to the group problem of integer programming*, Mathematical Programming Study **2** (1974), 51–71. Approaches to integer programming.

[16] ———, *A subadditive approach to solve linear integer programs*, Studies in integer programming (Proc. Workship, Bonn, 1975), 1977, pp. 117–143. Ann. of Discrete Math., Vol. 1.

[17] William Cook and Sanjeeb Dash, *On the matrix-cut rank of polyhedra*, Math. Oper. Res. **26** (2001), no. 1, 19–30.

[18] William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter, *A hybrid branch-and-bound approach for exact rational mixed-integer programming*, Math. Program. Comput. **5** (2013), no. 3, 305–344.

[19] Gérard Cornuéjols and Marco Molinaro, *A 3-Slope Theorem for the infinite relaxation in the plane*, Math. Program. **142** (2013), no. 1-2, Ser. A, 83–105.

[20] George B. Dantzig, *Linear programming and extensions*, Reprint of the 1968 corrected edition, Princeton Landmarks in Mathematics, Princeton University Press, Princeton, NJ, 1998.

[21] Santanu S. Dey, Jean-Philippe P. Richard, Yanjun Li, and Lisa A. Miller, *On the extreme inequalities of infinite group problems*, Math. Program. **121** (2010), no. 1, Ser. A, 145–170.

[22] Daniel Espinoza, Ricardo Fukasawa, and Marcos Goycoolea, *Lifting, tilting and fractional programming revisited*, Oper. Res. Lett. **38** (2010), no. 6, 559–563.

[23] Matteo Fischetti and Michele Monaci, *How tight is the corner relaxation?*, Discrete Optim. **5** (2008), no. 2, 262–269.

[24] Ralph E. Gomory, *Some polyhedra related to combinatorial problems*, Linear Algebra and Appl. **2** (1969), 451–558.

[25] Ralph E. Gomory and Ellis L. Johnson, *Some continuous functions related to corner polyhedra*, Math. Programming **3** (1972), 23–85.

[26] ______, *Some continuous functions related to corner polyhedra. II*, Math. Programming **3** (1972), 359–389.

[27] M. Guzelsoy, *Dual Methods in Mixed Integer Linear Programming*, 2010. Thesis (Ph.D.)–Lehigh University.

[28] M. Guzelsoy and T. K. Ralphs, *Duality for mixed-integer linear programs*, Int. J. Oper. Res. (Taichung) **4** (2007), no. 3, 118–137.

[29] John N. Hooker, *Integrated methods for optimization*, 2nd ed., International Series in Operations Research & Management Science, 170, Springer, New York, 2012.

[30] Robert G. Jeroslow, *Cutting-plane theory: algebraic methods*, Discrete Math. **23** (1978), no. 2, 121–150.

[31] ______, *Minimal inequalities*, Math. Programming **17** (1979), no. 1, 1–15.

[32] Ellis L. Johnson, *Cyclic groups, cutting planes, shortest paths*, Mathematical programming (Proc. Advanced Sem., Univ. Wisconsin, Madison, Wis., 1972), 1973, pp. 185–211.

[33] ______, *Integer programming*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 32, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa., 1980. Facets, subadditivity, and duality for group and semi-group problems; With a foreword by Stanley Zionts.

[34] ______, *On the group problem for mixed integer programming*, Math. Programming Stud. **2** (1974), 137–179. Approaches to integer programming.

[35] Diego Klabjan, *Subadditive approaches in integer programming*, European Journal of Operational Research **183** (2007), no. 2, 525–545.

[36] Monique Laurent, *A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming*, Math. Oper. Res. **28** (2003), no. 3, 470–496.

[37] R. R. Meyer, *On the existence of optimal solutions to integer and mixed-integer programming problems*, Math. Programming **7** (1974), 223–235.

[38] M. Perregaard, *Generating Disjunctive Cuts for Mixed Integer Programs*, 2003. Thesis (Ph.D.)–Carnegie Mellon University.

[39] Jean-Philippe P. Richard and Santanu S. Dey, *The Group-Theoretic Approach in Mixed Integer Programming*, 50 Years of Integer Programming 1958-2008, Springer Berlin Heidelberg, 2010.

[40] R. Tyrrell Rockafellar, *Convex analysis*, Princeton Mathematical Series, No. 28, Princeton University Press, Princeton, N.J., 1970.

[41] Alexander Schrijver, *Theory of linear and integer programming*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons Ltd., Chichester, 1986. A Wiley-Interscience Publication.

[42] Jeremy F. Shapiro, *Dynamic programming algorithms for the integer programming problem. I. The integer programming problem viewed as a knapsack type problem*, Operations Res. **16** (1968), 103–121.

[43] ______, *Group theoretic algorithms for the integer programming problem. II. Extension to a general algorithm*, Operations Res. **16** (1968), 928–947.

[44] M. Trick, *Is most published operations research false?*, Michael Trick's Operations Research Blog. `http://mat.tepper.cmu.edu/blog/?p=161`.

[45] Laurence A. Wolsey, *The b-hull of an integer program*, Discrete Appl. Math. **3** (1981), no. 3, 193–201.

[46] ______, *Extensions of the group theoretic approach in integer programming*, Management Sci. **18** (1971/72), 74–83.

[47] ______, *Integer programming duality: price functions and sensitivity analysis*, Math. Programming **20** (1981), no. 2, 173–195.

Index