New Valid Inequalities for Knapsack and Fixed-Charge Problems

Georgia Souli, B.Sc., M.Res

Submitted for the degree of Doctor of Philosophy at Lancaster University.

November 2020

STOR-i
excellence with impact

Abstract

A wide variety of important problems, in Operational Research and other fields, can be modelled as optimisation problems with integer-constrained variables. Algorithms and software for integer programming have improved substantially. One of the key ingredients to this success is the use of strong valid linear inequalities, also known as cutting planes. A key concept is that the convex hull of feasible solutions forms a polyhedron.

One strand of the literature on cutting planes is concerned with the knapsack polytope. In the 1970s, Balas and Wolsey derived a family of inequalities, called lifted cover inequalities (LCIs), for the knapsack polytope. We have taken a lifting procedure due to Balas, and shown that it can be substantially improved, so that it yields stronger and more general LCIs.

In 2000, Carr and co-authors introduced another family of valid inequalities for the knapsack polytope. These inequalities, called knapsack cover inequalities, can be rather weak. We have used two lifting procedures to strengthen these inequalities. The first procedure is based on integer rounding, whereas the second uses superadditivity.

Another important class of optimisation problems are those that involve fixed

charges. In 1985, Padberg, Van Roy and Wolsey introduced a procedure which enables one to take known valid inequalities for the knapsack polytope, and convert them into valid inequalities for the fixed-charge polytope. We have shown how this procedure can be extended to obtain a wider family of inequalities for the fixed-charge and single-node flow polytopes.

Finally, we have considered problems where a fixed charge is incurred if and only if at least one variable in a set takes a positive value. We have derived strong valid inequalities for these problems and shown that they generalise and dominate a subclass of the flow cover inequalities for the classical fixed-charge problem.

Acknowledgements

First and foremost, I would like to thank the EPSRC and Morgan Stanley for the financial support.

Furthermore, I would like to thank my supervisor, Adam Letchford, for his expert guidance over the last three years. His vast knowledge and experience have been invaluable.

I would also like to thank Alex Armstrong for his support and guidance. I am truly grateful for all the great opportunities that he has given me to challenge myself to keep learning and growing.

STOR-i Centre for Doctoral Training has been an amazing place to learn and grow. I would like to thank the directors, Jonathan Tawn and Idris Eckley, for their advice and pastoral support throughout the MRes and the PhD. I would also like to thank the STOR-i administrators, Jennifer Bull, Kim Wilson, Wendy Shimmin and Nicola Sarjent, for their continuing support.

Thank you to Niamh Lamin, the STOR-i intern that I supervised during the first summer of my PhD. Niamh's preliminary polyhedral experiments were very helpful for the fifth chapter of the thesis.

Last but not least, I would like to thank my family and friends in Greece for continuing to be there for me when I moved to the UK. Mum, Dad, Maria, Joanna, Niki, Fenia and Maria-Christina I am very grateful for having you in my life! I also feel very lucky for all the new friends that I made in Lancaster. I genuinely feel that I have found a second family here. Edwin, Euan, Hankui, Rob, Sean and Zak thank you so much!

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

This thesis is constructed as a series of papers and thus Chapters 2, 3, 4 and 5 should be read as separate entities.

Chapter 2 has been published as A.N. Letchford & G. Souli (2019) On lifted cover inequalities: A new lifting procedure with unusual properties. *Operations Research Letters*, vol. 47, pp. 83-87.

Chapter 3 has been published as A.N. Letchford & G. Souli (2020) Lifting the knapsack cover inequalities for the knapsack polytope. *Operations Research Letters*, vol. 48, pp. 607-611.

Chapter 4 has been published as A.N. Letchford & G. Souli (2019) New valid inequalities for the fixed-charge and single-node flow polytopes. *Operations Research Letters*,

vol. 47, pp. 353-357.

Chapter 5 has been published as A.N. Letchford & G. Souli (2020) Valid inequalities for mixed-integer programmes with fixed charges on sets of variables. *Operations Research Letters*, vol. 48, pp. 240-244.

Georgia Souli

Contents

List of Figures

List of Abbreviations

CI	Cover Inequality
FCI	Flow Cover Inequality
FCNF	Fixed-Charge Network Flow
FCP	Fixed-Charge Polytope
ILP	Integer Linear Program
KCI	Knapsack Cover Inequality
KP	Knapsack Polytope
LCI	Lifted Cover Inequality
LKCI	Lifted Knapsack Cover Inequality
LP	Linear Program
MILP	Mixed-Integer Linear Program
MIR	Mixed-Integer Rounding
RKI	Rotated Knapsack Inequality
SNFP	Single-Node Flow Polytope

Chapter 1

Introduction

1.1 Optimisation

Many times it is required to determine the best course of action. For example, airlines decide on which destinations to fly with the aim of maximising their revenue. Supermarkets try to determine how many members of staff to allocate to each shift in order to minimise cost. Factories design their production line with the aim of maximising their revenue. The branch of applied mathematics that is concerned with methods for finding the best among a huge range of alternatives is called *optimisation*. Optimisation problems arise in many fields, such as Operational Research, Statistics, Computer Science and Engineering.

In practice, optimisation consists of the following steps. First, the problem in question needs to be formulated mathematically. To formulate a problem as an optimisation problem, one needs to define the variables, the objective function and the constraints. The variables represent the decisions to be made and their respective val-

ues are to be determined. The objective function is a performance measure, expressed as a function of the variables, and is to be maximised or minimised. The constraints represent restrictions on the values of the variables.

Once an optimisation problem has been formulated, one must design, analyse and implement one or more solution algorithms. Optimisation solution algorithms should be capable of yielding good quality solutions in reasonable computing times. In this thesis, we have explored several specific optimisation problems with the view to produce quicker algorithms.

The remaining of this chapter is structured as follows. In Section 1.2, we introduce a class of optimisation problems, known as linear programs, and some extensions. In Section 1.3, we show how binary variables can be used to formulate a wide range of problems. Section 1.4 is an introduction to computational complexity. In Section 1.5, we review some classical approaches for mixed-integer linear programs. In Section 1.6, we introduce polyhedral approaches, that can help improve the classical approaches substantially. Finally, in Section 1.7, we discuss the structure of the thesis.

1.2 Linear Programming and Extensions

Let m and n be positive integer numbers, denoting the number of variables and the number of constraints of the optimisation problem. Let c be an n-dimensional real-valued vector, b be an m-dimensional real-valued vector and A be an $m \times n$ real-valued matrix. Let x be the n-dimensional vector of decision variables that need to

be determined. A *linear program* is a problem of the form

$$\min c^T x$$

$$\text{s.t. } Ax \leq b$$

$$x \in \mathbb{R}^n_+.$$

The quantity $c^T x$ is called the objective function and the matrix A is called the constraint matrix. The vector c contains the objective function coefficients, and the vector b contains the right-hand sides of the inequalities.

Any vector x that satisfies the constraints is called a *feasible solution* to the optimisation problem. The set of all feasible solutions is called the *feasible region* of the problem. Provided that the feasible region is not an empty set (i.e., there is at least one feasible solution), and that the objective function is not unbounded (i.e., it cannot take infinitely small values), there exists at least one feasible solution that minimises the objective function. Such a solution is called *optimal*.

Linear programming has been studied in great depth and, nowadays, there are solution algorithms and computer software capable of solving large-scale linear problems within a reasonable amount of time. The most well-known solution algorithms are the simplex method [24] and interior point methods [13]. We omit detail for brevity.

Note that the variables of a linear program are allowed to take fractional values. If all the variables are required to take integer values, then the problem is called an *Integer Linear Program* (ILP). If some of the variables are required to be integer (and the remaining are allowed to take fractional values), then the problem is called a *Mixed-Integer Linear Program* (MILP). If we consider a MILP, but relax the

integrality constraints, the resulting problem is called the *continuous relaxation* or *LP relaxation* of the MILP. In general, MILPs are much harder to solve than LPs. Famous textbooks on MILP include [65, 73].

1.3 Binary Variables in Model Formulation

Integer variables are very useful for modelling quantities that can only take integer values, such as the number of people that perform a particular task. A particularly useful type of integer variables is associated with decisions where there are only two possible choices: yes and no. Such a decision can be represented by a variable y such that

$$y = \begin{cases} 1, & \text{if decision is yes} \\ 0, & \text{if decision is no.} \end{cases}$$

A variable that is restricted to just two values, 0 and 1, is called *binary*. Mixed-integer linear programs such that some of the variables are required to be binary are called mixed 0-1 linear programs.

Binary variables enable us to model a wide range of problems as integer programming problems. In the remaining of this section, we present some examples. The interested reader can find more examples in [55, 68].

1.3.1 The knapsack problem

The knapsack problem is a classic example of modelling yes-no decisions. Consider a set of n items and a knapsack with capacity b. For every item j, let a_j be its (positive)

weight, v_j be its (positive) value and

$$y_j = \begin{cases} 1, \text{ if item } j \text{ is included in the knapsack} \\ \\ 0, \text{ otherwise.} \end{cases}$$

The aim is to choose which items to include in the knapsack in order to maximise the total value. Given that the knapsack has a limited capacity, the total weight of the items that are included in the knapsack should not exceed the knapsack capacity. The knapsack problem can be formulated as follows

$$\begin{aligned} &\max \sum_{j=1}^{n} v_j y_j \\ &\text{s.t. } \sum_{j=1}^{n} a_j y_j \leq b \\ &y_j \in \{0,1\} \quad (j = 1, \ldots, n). \end{aligned}$$

1.3.2 Fixed charges

One of the main assumptions of MILP is *proportionality*. This means that the contribution of a variable to the objective function is proportional to the value of the variable. However, in several applications there is a fixed set-up cost associated to some activity (e.g., cost to initiate the activity).

Let x be a non-negative continuous variable that represents the level of the activity, k denote the "set-up cost" and c denote the cost per unit of activity. The total cost is equal to

$$f(x) = \begin{cases} k + cx, & \text{if } x > 0 \\ \\ 0, & \text{if } x = 0. \end{cases}$$

In order to convert the problem to the MILP format, we introduce a binary variable y such that

$$y = \begin{cases} 1, \text{ if } x > 0 \\ \\ 0, \text{ if } x = 0. \end{cases}$$

Then, we can write $f(x) = cx + ky$. In order to ensure that y is equal to 1 if $x > 0$, we also add to the problem the constraint $x \leq My$, where M is a "large" positive number. This approach is known as the *"big M"* method.

1.3.3 Either-or constraints

Another main assumption of MILP is that all the constraints of the problem must be satisfied. However, sometimes it is possible to choose between two constraints so that only one must hold. For example, there may be a choice as to which of two resources to use. In that case, only the constraint for the chosen resource has to hold. For example, suppose that it is required that either the constraint $x_1 + x_2 \leq 5$ or the constraint $x_1 + 2x_2 \leq 6$ has to hold (not necessarily both).

In order to convert the problem to the MILP format, we introduce a very large positive number M and a binary variable y, and add to the problem the constraints

$$x_1 + x_2 \leq 5 + My$$

$$x_1 + 2x_2 \leq 6 + M(1 - y).$$

To see how this formulation actually works, we consider the two possible values of y.

If $y = 1$, then the constraints become

$$x_1 + x_2 \leq 5 + M$$

$$x_1 + 2x_2 \leq 6.$$

The first inequality is satisfied trivially for any x_1, x_2, and, essentially, only the second constraint is enforced. On the other hand, if $y = 0$, then the constraints become

$$x_1 + x_2 \leq 5$$

$$x_1 + 2x_2 \leq 6 + M.$$

Similarly, the second inequality is satisfied trivially for any x_1, x_2, and, essentially, only the first constraint is enforced.

1.3.4 Semi-continuous variables

A semi-continuous variable, x, takes either the value 0, or a value greater than or equal to some constant ℓ. In order to model this type of variable in the MILP setting, we can introduce x as a continuous variable and then, also, introduce a binary variable, y, and the constraint

$$\ell y \leq x \leq My,$$

where M is again a very large positive number. This constraint ensures that, if $y = 0$, then x is also equal to 0. If $y = 1$, then x takes values greater than or equal to ℓ.

1.4 Computational Complexity

Once an optimisation problem has been formulated, one must design and implement solution algorithms. Before we introduce some well-known solution algorithms for MILP, it is important to discuss how we measure the efficiency of an algorithm. The interested reader is also referred to [27] for more detail on the topic.

Efficiency can be measured in a number of different ways. For example, an important measure of efficiency is the amount of computer memory that is required by the algorithm. However, the most commonly used measure of the efficiency is the algorithm's *running time*, which is the number of computational steps required by the algorithm.

Let n be a measure of the size of an instance of an optimisation problem. The *big* O notation is used to represent an asymptotic upper bound for the running time of an algorithm (based on a worst-case instance). The running time $f(n)$ of an algorithm is said to be $O(g(n))$ if there exist constants c and n_0 such that $f(n) \leq cg(n)$ for all $n \geq n_0$. For example, if the running time of an algorithm is known to never exceed $5n^2 - 3n + 9$, the algorithm would be said to run in $O(n^2)$ time.

In order to classify the difficulty of different optimisation problems, we consider a type of problems known as *decision problems*. These are problems that can be answered with a single 'yes' or 'no'. For example, given an integer w and a minimisation integer program with objective function z, the corresponding decision problem would be "is there a feasible solution x such that $z(x) < w$?".

An algorithm is said to *run in polynomial time*, or more briefly to *be polynomial*, if

its running time is $O(n^k)$, where k is a known constant. The class of decision problems for which there is a polynomial-time algorithm is denoted by $\mathcal{P}$. Essentially, $\mathcal{P}$ is the class of problems that can be solved "quickly".

The class of decision problems for which, given a 'yes" instance of the problems, this can be checked in polynomial time, is called $\mathcal{NP}$. Consider, for example, the problem of determining whether a given graph contains a Hamiltonian circuit. Suppose that, after using a huge amount of computing resources, we eventually find such a circuit. If we are given a list of the edges in the circuit, we can check in $O(n)$ time that the given list does indeed correspond to a circuit. So, the Hamiltonian circuit problem is in $\mathcal{NP}$. It is straightforward that $\mathcal{P} \subseteq \mathcal{NP}$. However, it has not been verified yet whether $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$.

A decision problem is called $\mathcal{NP}$-*complete* if it can be shown that any other problem in $\mathcal{NP}$ can be mapped to this problem in polynomial time in its input size. An optimisation problem is called $\mathcal{NP}$-*hard* if the corresponding decision problem is $\mathcal{NP}$-complete. For example, the Travelling Salesman Problem is $\mathcal{NP}$-hard, since the Hamiltonian circuit problem, which is $\mathcal{NP}$-complete [43], can be reduced to it.

An interesting class of $\mathcal{NP}$-hard problems are *weakly* $\mathcal{NP}$-*hard* problems. An optimisation problem is called weakly $\mathcal{NP}$-hard if there is a solution algorithm whose running time is bounded by a polynomial in both the size of the problem and the magnitude of the numbers involved. The corresponding algorithm is called *pseudo-polynomial*. For example, the knapsack problem is weakly $\mathcal{NP}$-hard. Let n denote the number of items and b denote the capacity of the knapsack. The knapsack problem can be solved in $O(nb)$ time using a technique called dynamic programming. This

does not give a true polynomial algorithm, since a positive integer b is exponential in the number of bits needed to encode b. The knapsack problem is $\mathcal{NP}$-hard, but, as long as b is "small", the problem is relatively easy to solve.

1.5 Classical Approaches for MILP

Mixed-integer linear programming is a very active area of research. Over the years, several solution algorithms have been developed. In the following subsections, we briefly introduce some of the most well-known classical solution approaches for MILP, i.e., branch-and-bound, cutting planes, dynamic programming and heuristics. We also refer the interested reader to [3, 6, 39] for more detail on the topic.

1.5.1 Branch-and-bound

Branch-and-bound was first introduced in [45]. It follows a "divide-and-conquer" strategy in the sense that the algorithm partitions the feasible region of the problem into subregions and explores each subregion recursively. The basic structure of the algorithm is an enumeration tree. The root node corresponds to the original problem. The algorithm starts with solving the LP relaxation of the original problem. If the solution satisfies the integrality constraints, then this solution is also the optimal solution to the original problem. Otherwise, we branch, which means that we create two (or more) child nodes of the parent node by adding constraints to the problem of the parent node.

At each node, we solve the LP relaxation of the corresponding MILP. If the optimal

solution satisfies the integrality constraints, then we do not consider this node any further, and the given solution is a candidate solution to the original problem. If it is better than the best candidate solution found so far (which is called the incumbent solution), then we update the incumbent solution. If the optimal solution does not satisfy the integrality constraints, there are two options. If the value of this solution is worse than the incumbent solution, there is no point in exploring this node any further. If it is better, then we branch again. The procedure continues until there are no more nodes to consider.

1.5.2 Cutting planes

Cutting planes, or cuts, were first introduced in [30]. A cutting plane for a MILP is an inequality that is satisfied by all feasible solutions of the MILP but not by all feasible solutions of the LP relaxation. The idea of the algorithm is intuitively rather simple. We solve the LP relaxation of the problem, and then, if the solution does not satisfy the integrality constraints, we find a cutting plane that is violated by this solution and add it to the formulation. Then, we solve the strengthened LP relaxation, and repeat the process until we find the optimal solution to the MILP.

Figure 1.5.1 demonstrates a very simple example of an ILP with 2 integer variables (x_1 and x_2), 5 linear constraints and 5 feasible integer solutions. The optimal solution to the LP relaxation is marked by $*$, and the dotted line shows a cutting plane that removes $*$ while all integer feasible solutions remain intact.

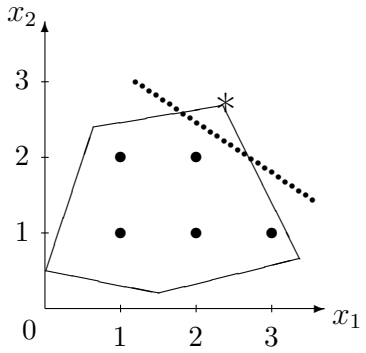

Figure 1.5.1: Example of a cutting plane.

1.5.3 Dynamic programming

In Section 1.4, we mentioned that the knapsack problem is an $\mathcal{NP}$-hard problem, but can be solved in $O(nb)$ time using a technique called *dynamic programming*. Dynamic programming was introduced by Bellman [10]. This method transforms a complex optimisation problem into a sequence of simpler subproblems. The basic idea is to decompose the problem into a nested sequence of smaller subproblems, such that the optimal solutions to the smaller subproblems can help us determine the optimal solutions to the larger ones.

1.5.4 Heuristics

All the techniques for MILP that we have reviewed so far are exact methods. This means that they provide the optimal solution to the MILP. However, exact methods do not always work well in practice. In many cases the required computational cost for the exact method can be prohibitive. Heuristics (first introduced in [62]) and meta-heuristics (first introduced in [28]) are methods that provide good feasible solutions for the problem within reasonable computational time. However, the obtained solutions

are not guaranteed to be optimal, and we do not know how far these solutions are from being optimal.

More specifically, a *heuristic* is a technique that employs a practical method to find a good (near-optimal) solution at a reasonable computational cost. A *metaheuristic* is a higher-level methodology that guides and modifies the operations of subordinate heuristics to produce high quality solutions. The interested reader can find more detail on the topic in [2, 14].

1.6 Polyhedral Approaches

In this section, we discuss how polyhedral approaches can be used to create faster algorithms for MILP. This section is structured as follows. In Subsection 1.6.1, we introduce relevant definitions and notation from polyhedral theory. In Subsection 1.6.2, we introduce the knapsack polytope. Finally, in Subsection 1.6.3, we introduce some algorithms that have been developed as a result of the use of polyhedral theory. The interested reader is also referred to [20] for a more detailed discussion on the topic.

1.6.1 Definitions and notation

Let $x^1, \ldots, x^k \in \mathbb{R}^n$ and $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$. The vector $x = \lambda_1 x^1 + \cdots + \lambda_k x^k$ is called a *linear combination* of $x^1, \ldots, x^k$. If $\sum_{i=1}^{k} \lambda_i = 1$, x is called an *affine combination* of $x^1, \ldots, x^k$. If x is an affine combination of $x^1, \ldots, x^k$ and $\lambda_i \geq 0$ for all $i = 1, \ldots, k$, then x is called a *convex combination* of $x^1, \ldots, x^k$. A set of vectors is called *affinely*

independent if no member of the set can be written as an affine combination of the others.

Let $S \subset \mathbb{R}^n$. The set S is called *convex* if $\theta x^1 + (1-\theta)x^2 \in S$ for all $x^1, x^2 \in S$ and $0 \leq \theta \leq 1$. Given $C \subset \mathbb{R}^n$, the *convex hull* of C, denoted by conv(C), is the smallest convex superset of C. Note that the convex hull of C is the set of all convex combinations of finitely many vectors in C.

A convex set $P \subseteq \mathbb{R}^n$ is called a *polyhedron* if it can be written as a solution set of finitely many linear inequalities. In other words, there are (a) a non-negative integer m, (b) an $m \times n$ real-valued matrix A, and (c) a real-valued vector b such that

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}.$$

If a polyhedron is a bounded set, it is called a *polytope*. Let k denote the largest number of affinely independent vectors in a polytope P. The dimension of P, denoted by $dim(P)$, is equal to $k-1$. The dimension of P can be -1 (if P is empty), 0 (if P consists of one point), and up to n if $P \in \mathbb{R}^n$. If $dim(P) = n$, then P is called *full-dimensional*.

Let $\alpha^T x \leq \beta$ be a linear valid inequality for a polyhedron P. The set

$$F = P \cap \{x \in \mathbb{R}^n : \alpha^T x \leq \beta\}$$

is called the *face* of polyhedron induced by the given valid inequality. If $F \neq \emptyset$, the valid inequality is called *supporting*. If $dim(F) = dim(P) - 1$, then F is called a *facet* of P.

In order to demonstrate some of the concepts that we have introduced in this subsection, we return to simple example of Figure 1.5.1. The inner polygon in Figure

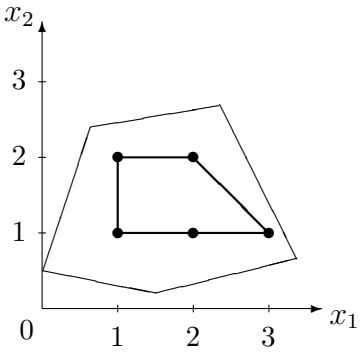

Figure 1.6.1: Convex hull.

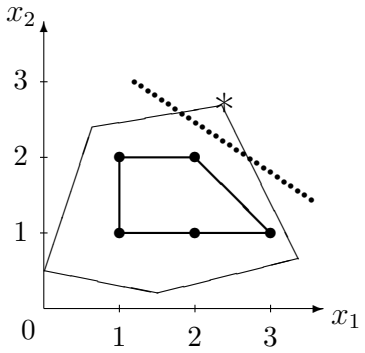

Figure 1.6.2: Weak cut.

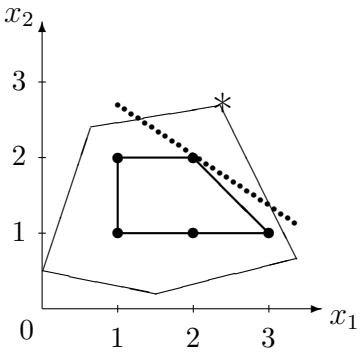

Figure 1.6.3: Stronger cut.

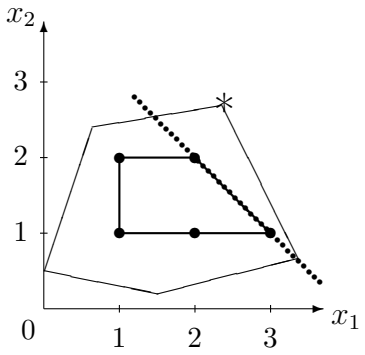

Figure 1.6.4: Facet-defining cut.

1.6.1 demonstrates the convex hull of the feasible region. Figures 1.6.2, 1.6.3 and 1.6.4 illustrate three different cutting planes that can be used to remove the solution of the LP relaxation. The strength of a cutting plane is determined by the convex hull of the feasible region of the MILP. The cutting plane in Figure 1.6.2 does not even touch the convex hull. The cutting plane in Figure 1.6.3 is stronger because it touches the convex hull and the one in Figure 1.6.4 is a strongest possible cutting plane, because it defines a facet of the convex hull.

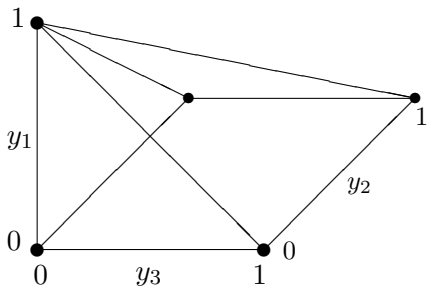

Figure 1.6.5: Example of a knapsack polytope.

1.6.2 Example: The knapsack polytope

In Section 1.3, we introduced the 0-1 knapsack problem. In the 0-1 knapsack problem, we are given n items with given weights a_j and a knapsack with capacity b. The polyhedron

$$\mathrm{conv}\Big\{y \in \{0,1\}^n : \sum_{j=1}^{n} a_j y_j \leq b\Big\}$$

is called a *knapsack polytope*. In Figure 1.6.5, we have visualised the knapsack polytope

$$P_1 = \mathrm{conv}\Big\{y \in \{0,1\}^n : 2y_1 + y_2 + y_3 \leq 2\Big\}.$$

Consider a subset C of items. It is called a *cover* if their total weight is larger than the capacity of the knapsack. In the aforementioned example, $C_1 = \{1,2\}$ and $C_2 = \{1,3\}$ are covers. Intuitively, if we have a set of $|C|$ items, whose total weight exceeds the capacity of the knapsack, we can only fit at most $|C| - 1$ of them in the knapsack. Mathematically, this can be expressed as follows

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

This inequality is called a *cover inequality*. Returning to the example, the corresponding cover inequalities are $y_1 + y_2 \leq 1$ for C_1, and $y_1 + y_3 \leq 1$ for C_2.

A cover is called *minimal* if we remove one item and the remaining set is no longer a cover. If C is a minimal cover, then the associated cover inequality defines a face of dimension at least $|C|-1$. Furthermore, there exists at least one *lifted cover inequality* of the following form

$$\sum_{j \in C} y_j + \sum_{j \in N \setminus C} \alpha_j y_j \leq |C| - 1,$$

where α_j is a non-negative integer for all the items outside the cover. The process of computing the coefficients α_j is known as *lifting*. Several fast and effective algorithms exist for lifting cover inequalities [7, 9, 23, 32, 34, 41, 74]. We discuss lifting in more detail in following chapters of the thesis.

1.6.3 Algorithms

In 1983, Crowder, Johnson and Padberg [23] first used results regarding the facial structure of 0-1 polytopes to create cutting planes. They considered 0-1 LPs, treated each constraint in the problem as a knapsack constraint and derived lifted cover inequalities from that. Then, they used these cutting planes in the root node of the branch-and-bound tree. More specifically, after solving the LP relaxation, if the node demanded further consideration, they would attempt to detect violated cuts. Any detected cuts would be added to the formulation and the LP would be solved again. If no more violated cuts could be be detected, they would continue with branch-and-bound. This technique helped them to solve to optimality ten 0-1 LPs, which were until then considered to be impossible to solve to optimality within reasonable computation times. The technique of generating cutting planes in the root node of

the branch-and-bound tree has come to be known as *cut-and-branch*.

The use of cutting planes can be expanded to the other nodes of the branch-and-bound tree. The resulting technique is called *branch-and-cut* [59]. After solving the LP relaxation at each node, if the node demands further consideration, we attempt to detect violated cuts. Any cuts detected are added to the formulation and the LP is solved again. If no more violated cuts can be detected, we branch. The problem of finding violated cuts is called the *separation* problem. There is a trade-off between the strength of the cuts and the time it takes to solve the separation problem. If the required computational time is prohibitive, we can branch instead. Furthermore, the more cuts created, the more complicated the problem. So, it might be reasonable to remove some cuts from the formulation in order to keep the size of the problem manageable.

1.7 Structure of the Thesis

The thesis is structured as follows.

Chapter 2 has been published as [48]. It is concerned with lifted cover inequalities for the knapsack polytope. More specifically, in this chapter we show how one of the earliest lifting procedures, due to Balas, can be significantly improved. The resulting procedure has some unusual properties. For example, (i) it can yield facet-defining inequalities even if the given cover is not minimal, (ii) it can yield facet-defining inequalities that cannot be ontained by standard lifting procedures, and (iii) the associated lifting function is integer-valued almost everywhere.

In Chapter 3, which has been published as [50], we revisit the knapsack polytope. In this chapter, we focus on the knapsack cover inequalities, introduced in 2000 by Carr and co-authors [15]. In general, these inequalities can be rather weak. To strengthen them, we use lifting. Since exact lifting can be time-consuming, we present two fast approximate lifting procedures. The first procedure is based on mixed-integer rounding, whereas the second uses superadditivity.

Chapter 4 has been published as [47]. It is concerned with new valid inequalities for two other polytopes, namely the fixed-charge polytope and the single-node flow polytope. More specifically, in this chapter we introduce a new procedure which enables one to take known valid inequalities for the knapsack polytope, and convert them into valid inequalities for the fixed-charge and single-node flow polytopes. The resulting inequalities are very different from the previously known inequalities, and define facets under certain conditions.

Chapter 5 has been published as [49]. It is concerned with valid inequalities for mixed-integer programs with fixed charges on sets of variables. More specifically, we consider mixed 0-1 linear programs in which one is given a collection of (not necessarily disjoint) sets of variables and, for each set, a fixed charge is incurred if and only if at least one of the variables in the set takes a positive value. We derive strong valid linear inequalities for these problems, and show that they generalise and dominate a subclass of the well-known flow cover inequalities for the classical fixed-charge problem.

Finally, Chapter 6 is the conclusion of this thesis. In this chapter, we provide an overview of our contributions and discuss further work that can stem from our work.

Chapter 2

On Lifted Cover Inequalities: A New Lifting Procedure with Unusual Properties

2.1 Introduction

Strong valid linear inequalities, also called *cutting planes*, are a key ingredient of modern exact algorithms for integer programs (see, e.g., [16, 21]). In the case of pure 0-1 linear programs (0-1 LPs), one very well-known and widely used family of cutting planes is the so-called *lifted cover inequalities* (LCIs), discovered independently by Balas [7] and Wolsey [69] (see also the surveys [5, 42]).

LCIs are obtained from a weaker family of inequalities, the so-called *cover inequalities* (CIs), by a process called *lifting*. Several procedures for lifting CIs have been proposed in the literature [7, 9, 23, 32, 34, 41, 74]. In this paper, we focus on one of

the earliest lifting procedures, which was described in Section 3 of Balas [7]. The LCIs generated by Balas' procedure are not guaranteed to define facets of the associated knapsack polytope, but they tend to be strong in practice. Moreover, the procedure is extremely fast. Specifically, it runs in only $O(n \log c)$ time, where n is the number of variables and c is the number of items in the cover.

The purpose of this paper is to show that the above-mentioned lifting procedure of Balas can be significantly improved, so that it yields both stronger and more general LCIs, while still running in only $O(n \log c)$ time. The improved procedure is sequence-independent, and it has some unusual properties:

- It can increase coefficients for variables inside the cover as well as outside.
- It can yield facet-defining LCIs even if the given cover is not minimal.
- It can even yield facet-defining inequalities that cannot be obtained by standard lifting procedures.
- The associated lifting function is integer-valued almost everywhere.

Moreover, the proof that the improved procedure is valid is itself unusual. It relies on the use of "dummy variables", by which we mean variables that do not exist in reality.

The paper is structured as follows. In Section 2.2, we review the literature. In Section 2.3, we present and analyse the improved lifting procedure. In Section 2.4, we show how to further enhance the procedure, via an analysis of superadditive functions. Finally, in Section 2.5, we discuss some preliminary computational results.

Throughout the paper, $x_1, \ldots, x_n$ will be a collection of binary variables and N will denote $\{1, \ldots, n\}$. Moreover, given a vector $v \in \mathbb{Q}^n_+$ and a set $S \subseteq N$, we will let $v(S)$ denote $\sum_{j \in S} v_j$.

2.2 Literature Review

We now briefly review the relevant literature.

2.2.1 Lifted cover inequalities

A *knapsack constraint* is a linear constraint of the form $\sum_{j \in N} a_j x_j \leq b$, where $a \in \mathbb{Z}^n_+$ and b is a positive integer. Any linear inequality involving binary variables can be converted into a knapsack constraint, by complementing variables with negative coefficients [69]. The polyhedron

$$\operatorname{conv}\left\{ x \in \{0,1\}^n : \sum_{j \in N} a_j x_j \leq b \right\}$$

is called a *knapsack polytope* [7].

A set $C \subseteq N$ such that $a(C) > b$ is called a *cover*. If C is a cover, then the inequality $x(C) \leq |C| - 1$ is valid for the knapsack polytope [29]. It is called a *cover inequality* (CI). A cover (and the associated CI) is *minimal* if $a\big(C \setminus \{k\}\big) \leq b$ for all $k \in C$. Minimal CIs dominate all other CIs.

Unfortunately, minimal CIs do not in general define facets of the knapsack polytope. On the other hand, given any minimal cover C, there exists at least one facet-

defining *lifted* cover inequality (LCI) of the form

$$x(C) + \sum_{j \in N \setminus C} \alpha_j x_j \leq |C| - 1, \tag{2.2.1}$$

where $\alpha_j \in \mathbb{Z}_+$ for $j \in N \setminus C$ [7, 58, 69]. (There may also exist facet-defining LCIs in which some α_j are fractional.) The process of computing the α_j for $j \in N \setminus C$ is called *lifting*. Encouraging computational results with LCIs were given in [23].

One can define more general LCIs of the form

$$x(C \setminus D) + \sum_{j \in N \setminus C} \alpha_j x_j + \sum_{j \in D} \beta_j x_j \leq |C \setminus D| - 1 + \beta(D), \tag{2.2.2}$$

where $D \subset C$ [58, 69]. Encouraging computational results with general LCIs are given in [32, 36, 41]. We will follow Gu *et al.* [32] in calling the computation of the α_j and β_j *up-lifting* and *down-lifting*, respectively. We will also call LCIs of the form (2.2.1) *simple*.

2.2.2 Balas' lifting procedure

Balas [7] introduced the following elegant up-lifting procedure, which can be implemented to run in $O\big(n \log |C|\big)$ time. Let C be a minimal cover and, for $r = 1, \ldots, |C|$, let $S(r)$ be the sum of the r largest a_j values over the members of C. Also let $S(0) = 0$. Given any $j \in N \setminus C$, let λ_j be the (unique) integer such that $S(\lambda_j) \leq a_j < S(\lambda_j + 1)$. Then the simple LCI

$$x(C) + \sum_{j \in N \setminus C} \lambda_j x_j \leq |C| - 1 \tag{2.2.3}$$

is valid.

Balas & Zemel [9] proved the following stronger result. Given any $j \in N \setminus C$, let μ_j be the (unique) integer such that

$$a(C) - S\big(\mu_j + 1\big) \leq b - a_j < a(C) - S\big(\mu_j\big). \tag{2.2.4}$$

Then, in any facet-defining simple LCI, we have $\lambda_j \leq \alpha_j \leq \mu_j \leq \lambda_j + 1$ for all $j \in N \setminus C$.

We will use the following example at several points through the paper.

Example 1. Let $n = 10$, $a = (15, 13, 9, 8, 8, 8, 5, 5, 5, 5)$ and $b = 16$. The set $\{7, 8, 9, 10\}$ is a minimal cover. We have $S(k) = 5k$ for $k = 0, \ldots, 4$. Since $a_1 = 15 \geq S(3)$, we have $\lambda_1 = 3$. Since $a_2 = 13 \geq S(2)$, we have $\lambda_2 = 2$. On the other hand, since $a_3, \ldots, a_6 < S(2)$, we have $\lambda_3, \ldots, \lambda_6 = 1$. The resulting (simple) LCI is therefore

$$3x_1 + 2x_2 + x_3 + \cdots + x_{10} \leq 3. \tag{2.2.5}$$

One can also check that $\mu_1 = \mu_2 = 3$ and $\mu_k = 2$ for $k \in \{3, 4, 5, 6\}$. So, in any facet-defining LCI obtained from that specific cover, $\alpha_1 = 3$, $\alpha_2 \in [2, 3]$ and $\alpha_3, \alpha_4, \alpha_5, \alpha_6 \in [1, 2]$. □

2.2.3 Other lifting procedures

It was shown in [58, 69] that one can obtain at least one facet-defining simple LCI by performing up-lifting *sequentially*, i.e., one coefficient at a time. This can be done by solving a small knapsack problem for each variable in $N \setminus C$. Zemel [74] showed how to do it in $O(n\,|C|)$ time by dynamic programming.

It is also possible to obtain facet-defining simple LCIs with fractional coefficients, by up-lifting *simultaneously* instead of sequentially. Unfortunately, this is not easy. In fact, even *recognising* a facet-defining simple LCI obtained by simultaneous up-lifting is $\mathcal{NP}$-hard [35].

It is of course possible to perform simultaneous up-lifting *approximately* in polynomial time. If an approximate simultaneous up-lifting procedure does not require any ordering of the variables in $N \setminus C$, it is called *sequence-independent* [9, 34]. The procedure of Balas, described in the previous subsection, can be viewed as a simple sequence-independent up-lifting procedure.

Wolsey [71] established a connection between sequence-independent lifting and *superadditive functions*. Gu *et al.* [34] used that result to improve Balas' up-lifting procedure, without increasing the asymptotic running time. Their improved procedure can yield simple LCIs with fractional coefficients. Further results on lifting can be found in [32, 36, 42].

2.3 The New Procedure and Its Properties

In this section, we show how to improve the procedure of Balas [7], in a way that is different from the one given in [34]. Throughout this section, we assume that we have a fixed knapsack constraint $a^T x \leq b$ and a fixed (not necessarily minimal) cover C, for ease of notation. We let c and $a_{\max}$ denote $|C|$ and $\max_{j \in C}\{a_j\}$, respectively. Finally, we assume that the items in C have been sorted in non-increasing order of a_j value, and we let $\ell_1, \ldots, \ell_c$ be the sorted values. Note that the sorting can be performed in

$O(c \log c)$ time.

2.3.1 A key quantity

The following quantity will play a crucial role in our analysis.

Definition 2.3.1. *We let $\bar{a}$ denote the unique (positive and rational) number such that*

$$\sum_{j \in C} \min\left\{a_j, \bar{a}\right\} = b.$$

For example, if $C = \{1, 3, 4\}$, $a_1 = 10$, $a_3 = 7$, $a_4 = 5$ and $b = 18$, then $\bar{a} = 6.5$, since $6.5 + 6.5 + 5 = 18$.

Remark 1: We have $\frac{b}{c} \leq \bar{a} < a_{\max}$.

Remark 2: If the items in C have already been sorted, one can compute $\bar{a}$ in $O(c)$ time. See Algorithm 1.

In the remainder of this section, we let $C^- = \left\{j \in C : a_j \leq \bar{a}\right\}$ and $C^+ = C \setminus C^-$. Note that C^- can be empty, but C^+ cannot be $\big($since $\bar{a} < a_{\max}\big)$.

2.3.2 The improved procedure

The improved version of Balas' procedure is described in the following theorem.

Theorem 2.3.2. *For all $j \in C$, let $a_j^- = \min\left\{a_j, \bar{a}\right\}$. For $r = 1, \ldots, c$, let $S^-(r)$ be the sum of the r largest a_j^- values. (Note that $S^-(c) = b$.) Also let $S^-(0) = 0$. Finally, given any $j \in N \setminus C^-$, let γ_j be the largest integer such that $S^-(\gamma_j) < a_j \leq S^-(\gamma_j + 1)$.*

Algorithm 1 Efficient computation of $\bar{a}$

input: cover C, knapsack capacity b, sorted values $\ell_1, \ldots, \ell_c$

Set $\bar{a} := \ell_1$ and $\sigma := a(C) - b$
for $k = 1, \ldots, c-1$ **do**
    Let $\delta = \bar{a} - \ell_{k+1}$
    if $k\delta < \sigma$ **then**
        Set $\bar{a} := \ell_{k+1}$ and $\sigma := \sigma - k\delta$
    else
        Set $\bar{a} := \bar{a} - \sigma/k$ and $\sigma := 0$
        break
    end if
end for
if $\sigma > 0$ **then**
    Set $\bar{a} := b/c$
end if

output: Value of $\bar{a}$.

Then the inequality

$$x\big(C^-\big) + \sum_{j \in N \setminus C^-} \gamma_j x_j \;\leq\; c - 1 \tag{2.3.1}$$

is valid for the knapsack polytope, and it is at least as strong as (2.2.3).

Proof. First, we expand the definition of "knapsack polytope", by permitting b and/or some of the a_j to take fractional values. One can check that Balas up-lifting procedure is valid even in this more general setting. Without loss of generality, we assume that $C \setminus C^- = \{1, \ldots, c'\}$ for some $1 \leq c' \leq c$. We then define the following "augmented" knapsack polytope:

$$K^+ \;=\; \text{conv}\left\{ x \in \{0,1\}^{n+c'} : \sum_{j=1}^{n} a_j x_j \;+\; (\bar{a} + \epsilon) \sum_{j=n+1}^{n+c'} x_j \;\leq\; b \right\},$$

where ϵ is some small positive rational number. By construction, the set

$$\tilde{C} \;=\; C^- \cup \{n+1, \ldots, n+c'\}$$

is a cover for K^+, and it has the same cardinality as C. If ϵ is sufficiently small, then applying Balas' up-lifting procedure to the CI associated with $\tilde{C}$ yields the following LCI for K^+:

$$x(\tilde{C}) + \sum_{j \in N \setminus C^-} \gamma_j x_j \;\leq\; c - 1.$$

Now, the original knapsack polytope is the face of K^+ obtained by setting x_j to zero for $j = n+1, \ldots, n+c'$. Thus, the inequality (2.3.1) is valid for the original polytope. Finally, note that, by construction, $S^-(r) < S(r)$ for all r. Thus, $\gamma_j \geq \lambda_j$ for all $j \in N \setminus C$. Moreover, $\gamma_j \geq 1$ for all $j \in C \setminus C^-$. Thus, (2.3.1) is at least as strong as (2.2.3). □

As mentioned in the introduction, a peculiarity of the above proof is that it relies on a consideration of "dummy variables" (namely, $x_{n+1}, \ldots, x_{n+c'}$), which do not actually exist in the original problem.

We now illustrate Theorem 2.3.2 on the same example that we considered in Subsection 2.2.2.

Example 1 (cont.) We have $\bar{a} = 4$ and $C = C^+$. This means that $S^-(k) = 4k$ for $k = 0, \ldots, 4$. Since $a_1 = 15 > S^-(3)$, we have $\gamma_1 = 3$. Since $a_2 = 13 > S^-(3)$, we have $\gamma_2 = 3$. Since $a_3 = 9 > S^-(2)$, we have $\gamma_3 = 2$. Finally, since $a_j = 8 < S^-(2)$ for $j \in \{4, 5, 6\}$, we have $\gamma_j = 1$ for $j \in \{4, 5, 6\}$. Thus, the resulting (simple) LCI is

$$3(x_1 + x_2) + 2x_3 + x_4 + \cdots + x_{10} \leq 3.$$

This dominates the LCI (2.2.5). □

Now, observe that, if $\lambda_1, \ldots, \lambda_c$ have already been computed, then one can compute $S^-(0), \ldots, S^-(c)$ in $O(c)$ time. Moreover, for each $j \in N \setminus C^-$, one can compute γ_j in $O(\log c)$ time, by binary search. Thus, our improved lifting procedure can be performed in $O(n \log c)$ time.

2.3.3 Unusual properties of the new procedure

As stated in the introduction, our new lifting procedure has some unusual properties. The first is that, if we start with covers that are not minimal, we can obtain LCIs that are not simple. This is illustrated in the following example.

Example 2. Let $n = 5$, $a = (5, 5, 2, 2, 2)$ and $b = 10$. The cover $C = \{1, 2, 3, 4, 5\}$

is not minimal, but we can still apply our procedure. We have $\bar{a} = b/c = 2$, so that $S^-(r) = 2r$ for $r = 0, \ldots, 5$. Now, since $C^+ = \{1, 2\}$, we may be able to increase the coefficients of x_1 and x_2. Indeed, since $a_1 = a_2 = 5 > S^-(2)$, we have $\gamma_1 = \gamma_2 = 2$. The resulting valid inequality is $2x_1 + 2x_2 + x_3 + x_4 + x_5 \leq 4$. This is not a simple LCI, since only three variables have a left-hand side coefficient equal to 1. It is however an LCI, as one can see by setting $C = \{2, 3, 4, 5\}$, $D = \{2\}$, $\alpha_1 = 2$ and $\beta_2 = 2$ in (2.2.2). It also defines a facet of the associated knapsack polytope. □

In general, our procedure yields a simple LCI if and only if at least $|C|$ variables receive a left-hand side coefficient of 1. One can check that this is equivalent to requiring

$$\left|\left\{j \in N \setminus C^- : \bar{a} < a_j \leq S^-(2)\right\}\right| \geq |C \setminus C^-|.$$

In particular, we have the following result.

Lemma 2.3.3. *If the original cover C is minimal, then our procedure yields a simple LCI.*

Proof. Let $\sigma = a(C) - b$. Since C is minimal, we have $\sigma \leq \ell_c \leq \ell_2$. Thus,

$$a_{\max} = \ell_1 \leq \ell_1 + (\ell_2 - \sigma) = (\ell_1 + \ell_2) - \sigma \leq \ell_1^- + \ell_2^- = S^-(2).$$

This implies that $\gamma_k = 1$ for all $k \in C \setminus C^-$. □

An even more unusual property of the new procedure is that, if we start with covers that are not minimal, we can obtain inequalities that are not LCIs at all (in

the traditional sense).

Example 3. Let $n = 5$, $a = (10, 7, 7, 4, 4)$ and $b = 16$. The cover $C = \{1, \ldots, 5\}$ is not minimal, but we apply our procedure. We have $\bar{a} = b/c = 3\frac{1}{5}$, so that $S^-(0), \ldots, S^-(5)$ are 0, $3\frac{1}{5}$, $6\frac{2}{5}$, $9\frac{3}{5}$, $12\frac{4}{5}$ and 16. Now, since $C^+ = C$, we may be able to increase the coefficients for some of the variables in the cover. One can check that we get $\gamma_1 = 3$, $\gamma_2 = \gamma_3 = 2$ and $\gamma_4 = \gamma_5 = 1$. The resulting valid inequality is

$$3x_1 + 2x_2 + 2x_3 + x_4 + x_5 \leq 4.$$

One can check (by brute-force enumeration of all possible lifting orders) that this inequality cannot be obtained from a CI by standard lifting methods, sequential or otherwise. One can also check (either by hand or with the help of a software package such as `PORTA` [17]) that it defines a facet of the associated knapsack polytope. □

2.4 Additional Improvement Via Superadditivity

A further improvement in the lifting procedure can be achieved from a consideration of superadditive lifting functions. The lifting function associated with Theorem 2.3.2 is:

$$f(z) = \begin{cases} 0 & \text{if } z = 0, \\ h & \text{if } S^-(h) < z \leq S^-(h+1) \text{ for some } h = 0, \ldots, c-1; \end{cases}$$

where the domain of z is understood to be $[0, b]$. (Figure 2.4.1 shows the function $f(z)$ for Example 1.)

Our goal is to construct an even stronger lifting function. We will need the fol-

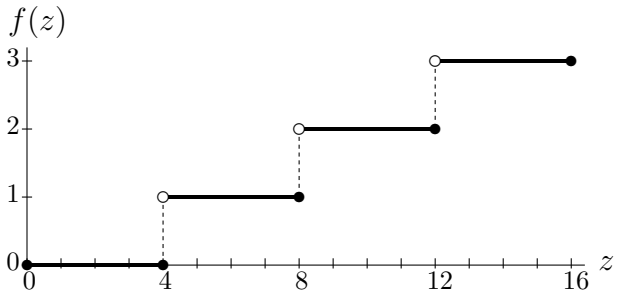

Figure 2.4.1: The lifting function $f(z)$ for Example 1.

lowing three results.

Lemma 2.4.1. *The function f is superadditive on its domain.*

Proof. Let $z, z' \in [0, b]$ be such that $z + z' \leq b$. Suppose that $f(z) = \gamma$ and $f(z') = \gamma'$. Then, by definition, we have $z > S^-(\gamma)$ and $z' > S^-(\gamma')$. Let $\ell_1^-, \ldots, \ell_c^-$ be the a_j^- values sorted in non-increasing order. We have $z > \sum_{j=1}^{\gamma} \ell_j^-$ and $z' > \sum_{j=1}^{\gamma'} \ell_j^-$. We then have:

$$z + z' > \sum_{j=1}^{\gamma} \ell_j^- + \sum_{j=1}^{\gamma'} \ell_j^- \geq \sum_{j=1}^{\gamma+\gamma'} \ell_j^-,$$

where the second inequality follows from the fact that the ℓ_j^- are sorted in non-increasing order. Thus, $f(z + z') \geq \gamma + \gamma' = f(z) + f(z')$. □

Lemma 2.4.2. *The upper bound of Balas and Zemel [9] remains valid even when the cover C is not minimal. That is, for any $k \in N \setminus C$, the lifting coefficient of x_k cannot exceed μ_k, where μ_k is the unique integer such that (2.2.4) holds.*

Proof. The lifting coefficient achieves its maximum possible value when x_k is lifted first. Note that, if we set x_k to 1, the remaining capacity in the knapsack is $b - a_k$. Then, the maximum value that $x(C)$ can take is equal to the largest integer s such

that $\sum_{j=c-s+1}^{c} \ell_j \leq b - a_k$. But $\sum_{j=c-s+1}^{c} \ell_j = a(C) - S(c-s)$. Since the right-hand side of the CI is $c-1$, the lifting coefficient cannot exceed $(c-1)-s$, which is nothing but μ_k. □

Lemma 2.4.3. *The function f reaches the Balas–Zemel bound when $z \geq b - a(C^-)$.*

Proof. Note that γ_j is the unique integer such that $S^-(\gamma_j) < a_j \leq S^-(\gamma_j + 1)$, and μ_j is the unique integer such that $S(\mu_j) + b - a(C) < a_j \leq S(\mu_j) + b - a(C)$. But $S^-(C) = b = S(C) + b - a(C)$, which implies that $S^-(k) = S(k) + b - a(C)$ for $k \geq |C^+|$. So $\gamma_j = \mu_j$ for $z \geq b - a(C^-)$. □

Lemma 2.4.3 leaves open the possibility that the value of $f(z)$ could be increased for some values of z smaller than $b - a(C^-)$. This is indeed the case.

Theorem 2.4.4. *The following function is a superadditive valid lifting function:*

$$g(z) = \begin{cases} f(z) + \frac{1}{2} & \text{if } z = h\,\bar{a} \text{ for some integer } h \in [1, |C^+| - 1] \\ f(z) & \text{otherwise.} \end{cases}$$

(See Figure 2.4.2 for an illustration.)

Proof. From the results in Wolsey [71], we need to prove that (a) g is superadditive and (b) $g(z)$ never exceeds the Balas–Zemel bound.

First, we prove superadditivity. Since f is superadditive, we need to prove only that $g(z) + g(z') \leq g(z + z')$ when $g(z) > f(z)$, i.e., when $z = h\,\bar{a}$ for some integer $h \in [1, |C^+| - 1]$. We have $z = \sum_{j=1}^{h} \ell_j^-$, where ℓ_j^- is defined as in the proof of Lemma 2.4.1. We also have $g(z) = h - \frac{1}{2}$. We consider two cases.

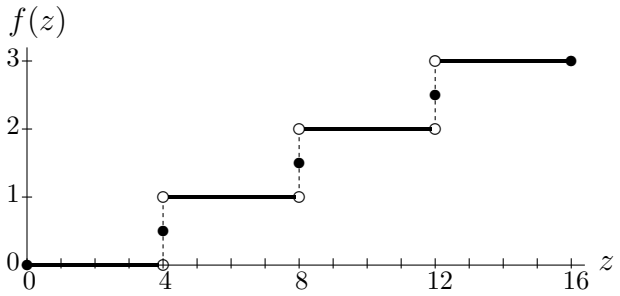

Figure 2.4.2: The improved lifting function $g(z)$ for Example 1.

1. $g(z') = f(z') = \gamma'$. In this case, we have $z' > \sum_{j=1}^{\gamma'} \ell_j^-$. This implies

$$z + z' > \sum_{j=1}^{h} \ell_j^- + \sum_{j=1}^{\gamma'} \ell_j^- \geq \sum_{j=1}^{h+\gamma'} \ell_j^-,$$

which in turn implies $g(z + z') \geq h + \gamma' > h - \frac{1}{2} + \gamma' = g(z) + g(z')$.

2. $g(z') = f(z') + \frac{1}{2}$. In this case, $z = h'\,\bar{a}$ for some integer $h' \in [1, |C^+| - 1]$. This implies

$$z + z' = \sum_{j=1}^{h} \ell_j^- + \sum_{j=1}^{h'} \ell_j^- \geq \sum_{j=1}^{h+h'} \ell_j^-,$$

which in turn implies $g(z + z') \geq h + h' - 1 = (h - \frac{1}{2}) + (h' - \frac{1}{2}) = g(z) + g(z')$.

Now we show that $g(z)$ never exceeds the Balas–Zemel bound. Since f is a valid lifting function, it follows that $f(z)$ never exceeds the bound. The only time that $g(z) > f(z)$ is when $z = h\bar{a}$ for some integer $h \in [1, |C^+| - 1]$. In this case, we have

$$b - z \;=\; b - h\bar{a} \;=\; \sum_{j=h+1}^{c} \ell_j^- \;<\; \sum_{j=h+1}^{c} \ell_j \;=\; a(C) - S(h).$$

Together with (2.2.4), this means that the Balas-Zemel upper bound is at least h. This exceeds $g(h\bar{a}) = h - \frac{1}{2}$. □

It turns out that using $g(z)$ in place of $f(z)$ can lead to stronger LCIs, even when

the a_j are integers and the cover is minimal.

Example 1 (cont.) We have $\bar{a} = 4$ and $|C^+| = 4$. Setting $h = 2$ in Theorem 2.4.4, we obtain $g(2\bar{a}) = g(8) = 3/2$. This yields the stronger LCI

$$3(x_1 + x_2) + 2x_3 + \frac{3}{2}(x_4 + x_5 + x_6) + x_7 + \cdots + x_{10} \leq 3.$$

This LCI can be shown to be facet-defining. □

We also have the following result:

Proposition 2.4.5. *The function $g(z)$ is non-dominated (that is, there does not exist a superadditive valid lifting function that is stronger than $g(z)$).*

Proof. Lemma 2.4.3 shows that $g(z)$ cannot be increased when $z \geq b - a\big(C^-\big) = |C^+|\,\bar{a}$. One can check that, for any pair (z, z') such that $z + z' = |C^+|\,\bar{a}$, we have $g(z) + g(z') = g\big(|C^+|\,\bar{a}\big) = |C^+| - 1$. Thus, $g(z)$ cannot be increased when $z < |C^+|\,\bar{a}$ either. □

Note that $g(z)$ is half-integral, and integer-valued almost everywhere. We found it surprising that a non-dominated lifting function with these properties exists. (Indeed, the lifting function presented in [34] is integer-valued only on certain intervals.)

We know of some other superadditive valid lifting functions that dominate $f(z)$. The one that we find most interesting is presented in the following proposition.

Proposition 2.4.6. *If $a_j \neq \bar{a}$ for all $j \in C$, the following function is a non-dominated*

superadditive valid lifting function :

$$g'(z) = \begin{cases} f(z) + 1 & \text{if } z = h\,\bar{a} \text{ for some integer } h \in \left(|C^+|/2, |C^+| - 1\right] \\ f(z) + \frac{1}{2} & \text{if } |C^+| \text{ is even and } z = |C^+|\bar{a}/2 \\ f(z) & \text{otherwise.} \end{cases}$$

Proof. The proof is similar to that of Theorem 2.4.4 and Proposition 2.4.5. The key difference occurs when $g'(z) = f(z) + 1$ and $g'(z') > f(z')$. In this case, let $z = h\bar{a}$ and $z' = h'\bar{a}$. Since h exceeds $|C^+|/2$ and h' is at least $|C^+|/2$, we must have $z + z' > |C^+|\,\bar{a}$. Moreover, if $a_j \neq \bar{a}$ for all $j \in C$, we have $S^-(k) < k\bar{a}$ for all $k > |C^+|$. This implies that $g'(z + z') \geq h + h' = g'(z) + g'(z')$. □

This lifting function is integer-valued at *all* points when $|C^+|$ is even.

2.5 Concluding Remarks

Our examples show that our new lifting procedure can yield non-trivial facet-defining inequalities. Therefore, a natural extension to our work is the design and implementation of efficient separation heuristics.

We conducted some initial computational experiments, where we compared our new procedure with Balas' original procedure. We focused on the *generalised assignment problem* and used the 27 instances used in [46]. In these experiments, we did not observe any significant improvement by using our procedure compared to using Balas' procedure. Note that we used the heuristic of [32] to select the initial cover. The covers generated by that heuristic are minimal. However, we know that our lifting

procedure can generate facet-defining inequalities even when the cover is not minimal. So, the lack of improvement may be due to choice of heuristic. One direction for further work could be the design of a better heuristic, which permits the generation of non-minimal covers. Future work could also include extending the computational study to other classes of problems.

Chapter 3

Lifting the Knapsack Cover Inequalities for the Knapsack Polytope

3.1 Introduction

A *knapsack constraint* is a linear constraint of the form $\sum_{i=1}^{n} a_i x_i \leq b$, where b and n are positive integers and $a \in \mathbb{Z}_+^n$. Knapsack constraints arise in many applications, such as vehicle routing [38], facility location [1], machine scheduling [61]. Moreover, any linear inequality involving binary variables can be converted into a knapsack constraint, by complementing variables with negative coefficients [69]. The polyhedron

$$\text{conv}\Big\{x \in \{0,1\}^n : \sum_{i=1}^{n} a_i x_i \leq b\Big\}$$

is called a *knapsack polytope* [7]. Valid inequalities for knapsack polytopes have proven to be very useful in exact algorithms for mixed-integer linear programming (e.g., [12, 23, 32, 34, 36, 41, 46]).

There are many papers on valid inequalities for knapsack polytopes. Most of these focus on *lifted cover* inequalities (e.g., [7, 9, 23, 32, 34, 35, 41, 48, 69, 74]), but there are a few papers on other families of inequalities. These include *weight* inequalities [67], *lifted pack* inequalities [5, 41], *Chvátal-Gomory cuts* [46], *Fenchel cuts* [12], and the inequalities in [15], which are (somewhat confusingly) called *knapsack cover* inequalities. These last inequalities have received very little attention in the literature, and have not been analysed from a polyhedral point of view.

We will see that, in general, knapsack cover inequalities can be rather weak. To strengthen them, we use *lifting* (see [7, 58, 69]). Since *exact* lifting can be time-consuming, we present two fast (and sequence-independent) approximate lifting procedures. The first procedure, which runs in $O(n)$ time, is based on a simple mixed-integer rounding argument (see [31, 54, 56]). The second procedure is stronger, but is a bit more complicated and runs in $O(n \log n)$ time. It is based on the construction of a valid superadditive lifting function (see [34, 71]). Our examples show that it is possible for both procedures to generate new facet-defining inequalities for the knapack polytope.

The paper has a simple structure. The literature is reviewed in Section 3.2, the new lifting procedures are presented in Section 3.3, and some concluding remarks are made in Section 3.4. Throughout the paper, we let N denote $\{1, \ldots, n\}$.

3.2 Literature Review

For brevity, we review here only works of direct relevance. We recall cover inequalities in Subsection 3.2.1, knapsack cover inequalities in 3.2.2, lifting in Subsection 3.2.3, and mixed-integer rounding in Subsection 3.2.4.

3.2.1 Cover inequalities

A set $C \subseteq N$ such that $\sum_{i \in C} a_i > b$ is called a *cover*. If C is a cover, then the *cover inequality* $\sum_{i \in C} x_i \leq |C| - 1$ is valid for the knapsack polytope [29]. A cover C is *minimal* if $\sum_{i \in C \setminus \{k\}} a_i \leq b$ for all $k \in C$. The minimal cover inequalities dominate all others [7, 69]. Although they are not guaranteed to define facets of the knapsack polytope, they can be strengthened to make them facet-defining (see Subsection 3.2.3).

3.2.2 Knapsack cover inequalities

Now consider a knapsack constraint of the form $\sum_{i \in N} a_i x_i \geq d$, where d and n are positive integers and $a \in \mathbb{Z}^n_+$. Crowder *et al.* [23] noted that such a constraint can be strengthened simply by replacing each a_i with $\min\{a_i, d\}$. Carr *et al.* [15] generalised this as follows. Consider any $S \subset N$, possibly empty, such that $\sum_{j \in S} a_j < d$. The inequality

$$\sum_{i \in N \setminus S} a_i x_i \geq d - \sum_{j \in S} a_j$$

is trivially valid, and it can be strengthened to yield:

$$\sum_{i \in N \setminus S} \min\left\{a_i, d - \sum_{j \in S} a_j\right\} x_i \geq d - \sum_{j \in S} a_j. \tag{3.2.1}$$

Rather confusingly, Carr *et al.* call the inequalities (3.2.1) *knapsack cover* inequalities. We will therefore refer to them as KCIs. The standard cover inequalities, mentioned in the previous subsection, will be called CIs.

3.2.3 Lifting

We now recall the basics of *lifting* [58, 70], focusing on 0-1 linear programs (0-1 LPs). Let $P \subset [0,1]^n$ be the convex hull of feasible solutions to a 0-1 LP, let S be a proper subset of N, and let $P(S)$ be the face of P obtained by setting x_i to 0 for all $i \in S$. Suppose we know that $\dim\big(P(S)\big) = \dim(P) - |S|$, and that the inequality

$$\sum_{i \in N \setminus S} \alpha_i x_i \leq \beta$$

defines a facet of $P(S)$. Then, there exists at least one inequality of the form

$$\sum_{i \in N \setminus S} \alpha_i x_i + \sum_{i \in S} \gamma_i x_i \leq \beta,$$

that defines a facet of P. (In particular, any minimal cover inequality can be strengthened to make it facet-defining for the knapsack polytope.)

The process of computing the γ_i is called *lifting*. Lifting is usually done *sequentially*, i.e., one variable at a time. To compute each lifting coefficient, one has to solve an auxiliary 0-1 LP, which may be time-consuming. Fortunately, fast exact and approximate algorithms are available for sequentially lifting CIs [7, 9, 23, 32, 41, 69, 74]. For other kinds of inequalities, Wolsey [70] suggests solving the LP relaxations of the auxiliary 0-1 LPs.

There can sometimes exist facet-defining lifted inequalities that cannot be obtained by sequential lifting [69]. To obtain such inequalities, one must lift several variables

simultaneously. Unfortunately, simultaneous lifting is very complicated, even for CIs [9, 35]. Wolsey [71] devised an elegant way to perform simultaneous lifting *approximately*, based on superadditive functions. This approach, sometimes called *sequence-independent* lifting, has been used to good effect in, e.g., [5, 34, 48]. However, the resulting inequality is not guaranteed to define a facet of P. For brevity, we omit the details.

3.2.4 Mixed-integer rounding

Finally, we recall some results from cutting-plane theory. Let $P \subset \mathbb{R}^n_+$ be a polyhedron, and suppose that the inequality $\alpha^T x \leq \beta$, with $\beta \notin \mathbb{Z}$, is valid for P. It is well known that the inequality $\sum_{i \in N} \lfloor \alpha_i \rfloor x_i \leq \lfloor \beta \rfloor$ is satisfied by all integer points in P [18, 30]. Less well known is that one can derive a stronger inequality as follows [31, 56]. Given a real number r, let $\phi(r)$ denote $r - \lfloor r \rfloor$, the so-called *fractional part* of r. Also define the folllowing (continuous and non-decreasing) function

$$F_\beta(r) = \begin{cases} \lfloor r \rfloor, & \text{if } \phi(r) \leq \phi(\beta) \\ \lfloor r \rfloor + \frac{\phi(r) - \phi(\beta)}{1 - \phi(\beta)}, & \text{if } \phi(r) > \phi(\beta). \end{cases}$$

The strengthened inequality takes the form:

$$\sum_{i \in N} F_\beta\big(\alpha_i\big) x_i \leq \lfloor \beta \rfloor.$$

We follow [54, 56] in calling these inequalities *mixed-integer rounding* (MIR) inequalities.

3.3 Lifting Knapsack Cover Inequalities

In this section, we show how to strengthen the KCIs by lifting. In Subsection 3.3.1, we present some simple results and examples to motivate our study. In Subsection 3.3.2, we define lifted KCIs formally and give examples. In Subsections 3.3.3 and 3.3.4, we present our sequence-independent lifting procedures.

We remind the reader that there is one KCI (3.2.1) for every $S \subseteq N$ satisfying $\sum_{j\in S} a_j < d$. Throughout this section, we let d^- denote $d - \sum_{j\in S} a_j$, and we sometimes refer to the sets $L = \{i \in N \setminus S : a_i > d^-\}$ and $R = N \setminus (S \cup L)$. (The idea here is that L contains indices with "large" a_i value, and R contains the "remaining" indices.) With this notation, the KCIs can be written in the simpler form

$$\sum_{i\in R} a_i x_i \;+\; d^- \sum_{i\in L} x_i \;\geq\; d^-. \tag{3.3.1}$$

We let e denote the all-ones vector of length n. We also frequently refer to the following two polytopes:

$$\begin{aligned} P^{\geq} &= \operatorname{conv}\Big\{x \in \{0,1\}^n : a^T x \geq d\Big\} \\ P^{\leq} &= \operatorname{conv}\Big\{\bar{x} \in \{0,1\}^n : a^T \bar{x} \leq e^T a - d\Big\}. \end{aligned}$$

Note that these polytopes are congruent, via the trivial mapping $\bar{x}_i = 1 - x_i$ for $i \in N$.

3.3.1 Motivation

In some preliminary experiments with the software package `PORTA` [17], we found that KCIs usually (though not always) define low-dimensional faces of $P^{\geq}$. A partial explanation is given by the following two lemmas:

Lemma 3.3.1. *If $\sum_{j\in R} a_j < d^-$, then the KCI (3.3.1) is equivalent to or dominated by the inequality $\sum_{i\in L} x_i \geq 1$. Note that this inequality is equivalent to a CI for $P^{\leq}$.*

Proof. By the definition of d^-, the stated condition can be written as $\sum_{j\in R\cup S} a_j < d$. Under this condition, the inequality $\sum_{i\in L} x_i \geq 1$ is valid for $P^{\geq}$. Writing this as $d^- \sum_{i\in L} x_i \geq d^-$, we see that it is at least as strong as the KCI. Writing it as $\sum_{i\in L} \bar{x}_i \leq |L| - 1$ instead, we see that it is equivalent to a CI for $P^{\leq}$. □

Lemma 3.3.2. *If $\sum_{j\in R} a_j \geq d^-$, but $\sum_{j\in R\setminus\{i\}} a_j < d^-$ for all $i \in R$, then the KCI (3.3.1) is dominated by the inequalities $\sum_{j\in L\cup\{i\}} x_j \geq 1\,(i \in R)$. Note that these inequalities are equivalent to CIs for $P^{\leq}$.*

Proof. Suppose the stated conditions hold. If $x_i = 0$ for all $i \in L$, then we must set x_i to 1 for all $i \in R$. Thus, the following inequalities are valid for $P^{\geq}$:

$$\sum_{j\in L\cup\{i\}} x_j \geq 1 \qquad (i \in R). \tag{3.3.2}$$

Writing these inequalities in the form $\sum_{j\in L\cup\{i\}} \bar{x}_j \leq |L|$, we see that they are equivalent to CIs for $P^{\leq}$. Now, for each $i \in R$, multiply the inequality (3.3.2) by $a_i d^- / \sum_{j\in R} a_j$, and sum the resulting $|R|$ inequalities together, to yield:

$$\frac{d^-}{\sum_{j\in R} a_j} \sum_{i\in R} a_i x_i + d^- \sum_{i\in L} x_i \geq d^-.$$

Since $\sum_{j\in R} a_j \geq d^-$ by assumption, this last inequality is at least as strong as the KCI. □

When the conditions in Lemmas 3.3.1 and 3.3.2 do not hold, the KCI may or may not define a facet of $P^{\geq}$. This is shown in the following example.

Example 1: Let $n = d = 7$ and $a = (1, 2, 2, 2, 4, 4, 7)^T$. Taking $S = \{1\}$ yields the

KCI $2(x_2+x_3+x_4)+4(x_5+x_6)+6x_7 \geq 6$. We have $R = \{2,\ldots,6\}$ and $\sum_{i\in R} a_i = 14$, so Lemmas 3.3.1 and 3.3.2 do not apply. One can check (either by hand or with the help of a package like `PORTA`) that this KCI defines a facet of $P^{\geq}$. On the other hand, if we take $S = \{2\}$, we get the KCI $x_1 + 2(x_3 + x_4) + 4(x_5 + x_6) + 5x_7 \geq 5$. We have $R = \{1,3,4,5,6\}$ and $\sum_{i\in R} a_i = 13$, so, again, the lemmas do not apply. Yet, this KCI does not define a facet, since every extreme point of $P^{\geq}$ that satisfies it at equality also satisfies $x_1 + x_7 = 1$. □

3.3.2 Lifted KCIs

We propose to lift KCIs, *regardless* of whether or not Lemma 3.3.1 or Lemma 3.3.2 applies. The idea is as follows. The KCI (3.3.1) is equivalent to the following valid inequality for $P^{\leq}$:

$$\sum_{i\in R} a_i\bar{x}_i + d^- \sum_{i\in L} \bar{x}_i \leq \sum_{i\in R} a_i + d^-(|L| - 1).$$

It is now apparent that we may be able to lift the variables in S, to obtain a valid inequality for $P^{\leq}$ of the form:

$$\sum_{i\in R} a_i\bar{x}_i + d^- \sum_{i\in L} \bar{x}_i + \sum_{i\in S} \gamma_i\bar{x}_i \leq \sum_{i\in R} a_i + d^-(|L| - 1).$$

The corresponding valid inequality for $P^{\geq}$ takes the form:

$$\sum_{i\in R} a_i x_i + d^- \sum_{i\in L} x_i \geq d^- + \sum_{i\in S} \gamma_i(1 - x_i). \tag{3.3.3}$$

We call (3.3.3) a *lifted knapsack cover inequality* or LKCI. In general, LKCIs are not guaranteed to define facets of $P^{\geq}$. On the other hand, the following example

shows that LKCIs can define non-trivial facets even when Lemma 3.3.1 applies.

Example 2: Let $n = 5$, $d = 8$ and $a = (2, 2, 2, 5, 5)^T$. Taking $S = \{1, 2\}$ yields the KCI $2x_3 + 4(x_4 + x_5) \geq 4$. We have $R = \{3\}$ and $\sum_{i \in R} a_i = 2 < d^- = 4$. Thus, Lemma 3.3.1 applies. One can check however that the LKCI $2x_3 + 4(x_4 + x_5) \geq 4 + 2(1 - x_1) + 2(1 - x_2)$ is valid and facet-defining for $P^{\leq}$. Moreover, if we write the LKCI in the form $\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + 2(\bar{x}_4 + \bar{x}_5) \leq 3$, we see that it is not equivalent to a lifted CI. □

Now recall that lifting can be done sequentially or simultaneously. If one wishes to lift a KCI sequentially, one must solve an auxiliary knapsack problem (KP) to compute each lifting coefficient [58, 70]. This is likely to be too time-consuming to be useful in practical computation. Following Wolsey [71], one could compute approximate lifting coefficients sequentiallly, by solving the continuous relaxations of the KPs. We prefer however to use sequence-independent lifting, as described in the following two subsections.

3.3.3 Lifting via mixed-integer rounding

It turns out that one can lift KCIs using mixed-integer rounding. This can be done in four steps, as follows.

1. Write the constraint $a^T x \geq d$ in the form

$$\sum_{i \in S} a_i \bar{x}_i - \sum_{i \in N \setminus S} a_i x_i \leq -d^-. \tag{3.3.4}$$

2. Let $a^+ = \max_{i \in N \setminus S}\{a_i\}$, and assume that $a^+ > d^-$ (since, if not, the KCI is redundant.) Divide (3.3.4) by a^+ to obtain

$$\sum_{i \in S} \left(\frac{a_i}{a^+}\right) \bar{x}_i + \sum_{i \in N \setminus S} \left(\frac{-a_i}{a^+}\right) x_i \leq -\frac{d^-}{a^+}.$$

3. We now set $-d^-/a^+$ to β, and apply mixed-integer rounding to get

$$\sum_{i \in S} F_\beta\big(a_i/a^+\big)\bar{x}_i + \sum_{i \in N \setminus S} F_\beta(-a_i/a^+)x_i \leq \lfloor \beta \rfloor.$$

We can simplify this inequality, as follows. Since $a^+ > d^-$, we have $\lfloor \beta \rfloor = -1$ and $\phi(\beta) = 1 - d^-/a^+$. For $i \in R$, we have that $a_i \leq a^+$ (by the definition of a^+) and $a_i \leq d^-$ (by the definition of R). Hence, $\phi(-a_i/a^+) = 1 - a_i/a^+ \geq \phi(\beta)$ and $F_\beta(-a_i/a^+) = -a_i/d^-$. For $i \in L$, we have that $a_i \leq a^+$ (by the definition of a^+) and $a_i > d^-$ (by the definition of L). Hence, $\phi(-a_i/a^+) = 1 - a_i/a^+ < \phi(\beta)$ and $F_\beta(-a_i/a^+) = \lfloor -a_i/a^+ \rfloor = -1$. So, we get

$$\sum_{i \in S} F_\beta\big(a_i/a^+\big)\bar{x}_i - \sum_{i \in R} \frac{a_i}{d^-} x_i - \sum_{i \in L} x_i \leq -1.$$

4. Multiplying the MIR inequality by d^- and re-arranging, we obtain:

$$\sum_{i \in R} a_i x_i + d^- \sum_{i \in L} x_i \geq d^- + d^- \sum_{i \in S} F_\beta\big(a_i/a^+\big)(1 - x_i). \tag{3.3.5}$$

This is the desired LKCI.

The following example shows that the above MIR procedure can yield non-trivial facet-defining LKCIs.

Example 3: Let $n = 7$, $d = 17$ and $a = (3, 3, 3, 4, 7, 7, 7)^T$. Taking $S = \{4, 5\}$ yields the KCI $3(x_1 + x_2 + x_3) + 6(x_6 + x_7) \geq 6$. We have $d^- = 6$ and $a^+ = 7$, which gives

$\beta = -6/7$. We have $F_\beta\big(a_4/a^+\big) = F_{1/7}(4/7) = 1/2$ and $F_\beta\big(a_5/a^+\big) = F_{1/7}(1) = 1$. The resulting LKCI is therefore $3(x_1+x_2+x_3)+6(x_6+x_7) \geq 6+3(1-x_4)+6(1-x_5)$. One can check (either by hand or with the help of a package like `PORTA`) that this LKCI defines a facet of $P^{\geq}$. Moreover, if we write the LKCI in the form $\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + 2(\bar{x}_5 + \bar{x}_6 + \bar{x}_7) \leq 5$, we see that it is not equivalent to a lifted CI. □

For the purpose of what follows, we will find it helpful to express the lifting coefficients in the LKCI (3.3.5) in a more explicit form.

Proposition 3.3.3. *For any $r \geq 0$, let $f(r)$ denote $d^- F_\beta\big(r/a^+\big)$. We have*

$$f(r) = \begin{cases} d^- \lfloor r/a^+ \rfloor & \text{if } r \bmod a^+ \leq a^+ - d^- \\ d^- \lceil r/a^+ \rceil - a^+ + r \bmod a^+ & \text{if } r \bmod a^+ > a^+ - d^-. \end{cases}$$

Proof. Write r as $ka^+ + \epsilon$, where $k = \lfloor r/a^+ \rfloor$ and $\epsilon = r \bmod a^+$. If $\epsilon \leq a^+ - d^-$, we have $\phi(r/a^+) = \epsilon/a^+ \leq 1 - d^-/a^+ = \phi(\beta)$, and therefore

$$F_\beta\big(r/a^+\big) = k.$$

On the other hand, if $\epsilon > a^+ - d^-$, we have $\phi(r/a^+) > \phi(\beta)$, and therefore

$$\begin{aligned} F_\beta\big(r/a^+\big) &= k + \frac{\phi\big(r/a^+\big) - \big(1 - d^-/a^+\big)}{d^-/a^+} \\ &= k + \frac{a^+\phi\big(r/a^+\big) - a^+ + d^-}{d^-} \\ &= k + 1 - \tfrac{a^+ - \epsilon}{d^-}. \end{aligned}$$

We have established that:

$$F_\beta\Big(\frac{r}{a^+}\Big) = \begin{cases} \lfloor r/a^+ \rfloor & \text{if } r \bmod a^+ \leq a^+ - d^- \\ \lceil r/a^+ \rceil - \frac{a^+ - \epsilon}{d^-} & \text{if } r \bmod a^+ > -d^-. \end{cases}$$

Multiplying by d^- yields the result. □

We remark that the MIR function F_β is superadditive for any β (see [56]). Thus, the function f is superadditive for any d^- and a^+.

3.3.4 Lifting via superadditivity

Consider again the LKCI (3.3.3), and suppose that we wish to lift x_k first, for some $k \in S$. Let r denote a_k. Following [58, 70], we can compute the largest possible value of γ_k by computing

$$z(r) = \min \quad \sum_{i\in R} a_i x_i + d^- \sum_{i\in L} x_i \tag{3.3.6}$$

$$\text{s.t.} \quad \sum_{i\in L\cup R} a_i x_i \geq d^- + r \tag{3.3.7}$$

$$x_i \in \{0,1\} \qquad (i \in L \cup R), \tag{3.3.8}$$

and then setting γ_k to $z(r) - d^-$. The function $g(r) = z(r) - d^-$ is called the *exact lifting function*. Note that the domain of g is $\left[0, \sum_{j\in L\cup R} a_j - d^-\right]$ since, if r exceeds $\sum_{j\in L\cup R} a_j - d^-$, the above 0-1 LP becomes infeasible.

It follows from the main result in [71] that, if g is superadditive, then we can use it to lift all variables in S simultaneously. Unfortunately, g is not superadditive in general. This is demonstrated in the following example.

Example 4: Let $n = 7$, $a = (3, 3, 3, 7, 8, 9, 17)$, $d = 23$ and $S = \{7\}$. The reduced knapsack constraint is $3x_1 + 3x_2 + 3x_3 + 7x_4 + 8x_5 + 9x_6 \geq 6$. We have $d^- = 6$ and $a^+ = 9$. The KCI is $3(x_1 + x_2 + x_3) + 6(x_4 + x_5 + x_6) \geq 6$. The function g is shown in Figure 3.3.1. To see that g is not superadditive, note that, for example,

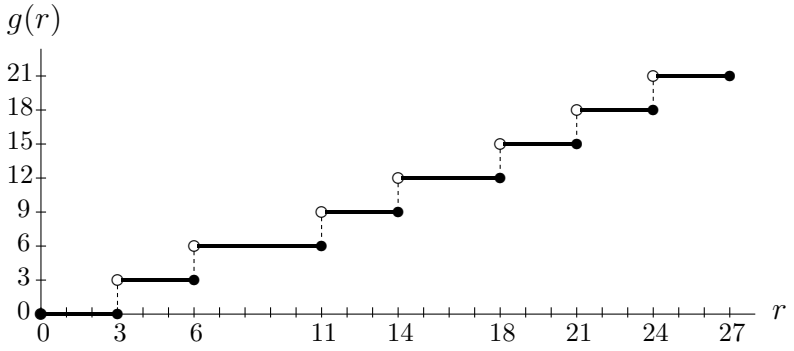

Figure 3.3.1: The exact lifting function $g(r)$ for Example 4.

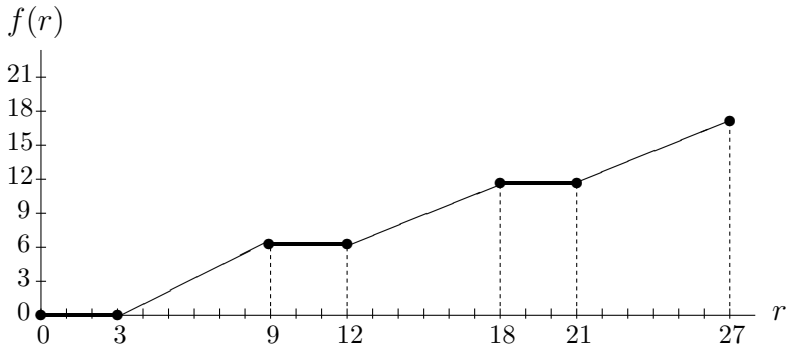

Figure 3.3.2: The lifting function $f(r)$ for Example 4.

$g(14) = 9 < 2g(7) = 12.$ □

Following the approach in [34, 71], we are led to search for *superadditive valid lifting functions*, i.e., superadditive functions that do not exceed g. As, we remarked in subsection 3.3.3, the MIR lifting function, called f, is such a function. Figure 3.3.2 shows the function f for Example 4. Note that f is piecewise-linear, and each "piece" has a slope equal to either 0 or 1.

We now introduce a third lifting function, called h, which we will show to be

superadditive and intermediate between f and g. Let $h(r) = \tilde{z}(r) - d^-$, where

$$\tilde{z}(r) = \min \quad y + d^- \textstyle\sum_{i \in L} x_i \tag{3.3.9}$$

$$\text{s.t.} \quad y + \textstyle\sum_{i \in L} a_i x_i \geq d^- + r \tag{3.3.10}$$

$$x_i \in \{0, 1\} \qquad (i \in L) \tag{3.3.11}$$

$$y \geq 0. \tag{3.3.12}$$

Note that, due to the continuous variable y, $\tilde{z}(r)$ is feasible for all non-negative values of r. Hence, the domain of h is the whole of $\mathbb{R}_+$. In the following proposition, we give an explicit formula for h.

Proposition 3.3.4. *For $k = 1, \ldots, |L|$, let $S(k)$ be the sum of the k largest a_j values over the members of L. We trivially set $S(0) = 0$. Then, we can write $h(r)$ as follows*

$$h(r) = \begin{cases} kd^-, & \text{if } S(k) \leq r < S(k+1) - d^- \text{ for } k \in \{0, \ldots, |L| - 1\} \\ r - (S(k) - d^-), & \text{if } S(k) - d^- \leq r < S(k) \text{ for } k \in \{1, \ldots, |L| - 1\}. \\ r - (S(|L|) - d^-), & \text{if } r \geq S(|L|) - d^-. \end{cases}$$

Proof. Consider the mixed 0-1 knapsack problem (3.3.9)-(3.3.12). Observe that every time that we set a variable x_i to 1, we incur a cost d^-. This increases the LHS of the constraint (3.3.10) by a_i, which is greater than d^- by the definition of L. Hence, to find the optimal solution, it makes sense to keep switching on binary variables in decreasing order of a_i as long as the constraint (3.3.10) is violated. The continuous variable y will only take a positive value if the remaining violation of the constraint (3.3.10) is so "small" that it is not worth switching on an additional binary variable. □

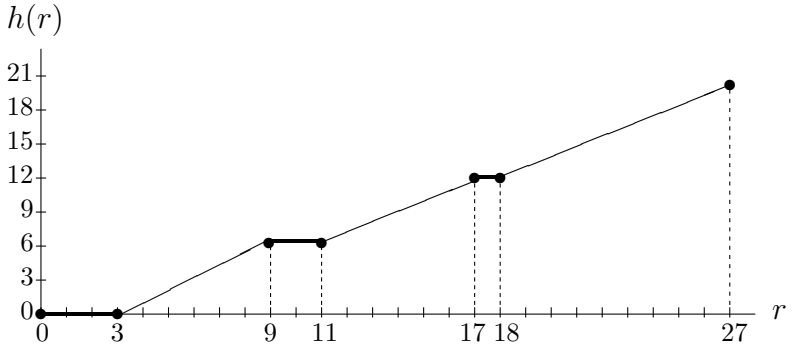

Figure 3.3.3: The lifting function $h(r)$ for Example 4.

To aid the reader, we show the lifting function h for Example 4.

Example 4 (cont.): Recall that $d^- = 6$ and $L = \{4, 5, 6\}$. Moreover, $S(0) = 0$, $S(1) = 9$, $S(2) = 17$ and $S(3) = 24$. Using these, one can compute the function h, which is shown in Figure 3.3.3. □

Note that, like f, h is piecewise-linear, and each "piece" has a slope equal to either 0 or 1. In the following proposition, we prove that h is intermediate between f and g.

Proposition 3.3.5. *For all* $r \in \left[0, \sum_{j \in L \cup R} a_j - d^-\right]$*, we have* $f(r) \leq h(r) \leq g(r)$.

Proof. We will first prove that $h(r) \leq g(r)$. Given that $g(r) = z(r) - d^-$ and $h(r) = \tilde{z}(r) - d^-$, it is sufficient to show that $\tilde{z}(r) \leq z(r)$. To see why this inequality holds, note that the mixed 0-1 knapsack problem (3.3.9)-(3.3.12) is a relaxation of the integer program (3.3.6)–(3.3.8), obtained by replacing the binary variables x_i for $i \in R$ with a single continuous variable y.

We will now prove that $h(r) \geq f(r)$. Recall that f and h are piecewise-linear. For the function h, the k-th "piece" with slope 0 has length $\ell_k^h = S(k+1) - S(k) - d^-$. For the function f, the k-th piece with slope 0 has length $\ell_k^f = a^+ - d^-$. By the definitions

of S and a^+, we have $\ell_k^h \leq \ell_k^f$. Hence, $h(r) \geq f(r)$ for all $r \in \left[0, \sum_{j \in L \cup R} a_j - d^-\right]$. (The reader may find it helpful to compare Figures 3.3.2 and 3.3.3.) □

In the following proposition, we prove that h is superadditive, which immediately implies that we can use h for simultaneous lifting.

Proposition 3.3.6. *The function h is superadditive in its domain.*

Proof. We will prove that h is superadditive by contradiction. Suppose that h is not superadditive. Then, there are values r_1 and r_2 such that $h(r_1 + r_2) < h(r_1) + h(r_2)$. Recall that each "piece" of the function has a slope of either 0 or 1. We will call the points where h is non-differentiable "breakpoints", and the points where h is differentiable "interior" points.

Suppose that r_1 is either an interior point where the slope is 0, or a breakpoint where the slope on the left of r_1 is 0 and the slope on the right is 1. Then, there exists a small $\epsilon > 0$ such that $h(r_1 - \epsilon) = h(r_1)$. We therefore have $h(r_1 - \epsilon) + h(r_2) = h(r_1) + h(r_2) > h(r_1 + r_2) \geq h(r_1 + r_2 - \epsilon)$. This means that $r_1 - \epsilon$ and r_2 also form a counter-example. So, we can assume w.l.o.g. that r_1 is neither an interior point with slope 0 nor a breakpoint where the slope on the left is 0.

Now, suppose that r_1 is an interior point with slope 1. Then, there exists a small $\epsilon > 0$ such that $h(r_1 + \epsilon) = h(r_1) + \epsilon$. We therefore have $h(r_1 + \epsilon) + h(r_2) = h(r_1) + h(r_2) + \epsilon > h(r_1 + r_2) + \epsilon \geq h(r_1 + r_2 + \epsilon)$. This means that $r_1 + \epsilon$ and r_2 also form a counter-example. So, we can assume w.l.o.g. that r_1 is a breakpoint where the slope on the left is 1 and the slope on the right is 0.

The same argument enables us to assume that r_2 is also a breakpoint of the

same type. Hence, to complete the proof, we have to show superadditivity for the case where both r_1 and r_2 are breakpoints of that type. Note these points are such that there exist positive integers k_1, k_2 such that $r_1 = S(k_1)$ and $r_2 = S(k_2)$. So, $r_1 + r_2 = S(k_1) + S(k_2) \geq S(k_1 + k_2)$ by the definition of S. The function h is increasing. So, $h(r_1 + r_2) \geq h(S(k_1 + k_2))$. By the definition of h, we have that $h(r_1) = h(S(k_1)) = k_1 d$, $h(r_2) = h(S(k_2)) = k_2 d$ and $h(S(k_1 + k_2)) = (k_1 + k_2)d^-$. This implies that $h(r_1 + r_2) \geq h(r_1) + h(r_2)$, which is a contradiction. □

We now revisit Example 4 to demonstrate the LKCIs that we get using f and h.

Example 4 (cont.): Recall that $d^- = 6$ and $L = \{4, 5, 6\}$. Using the MIR lifting function, we get the LKCI $3(x_1 + x_2 + x_3) + 6(x_4 + x_5 + x_6) \geq 6 + 11(1 - x_7)$. Using the lifting function h, we get the stronger LKCI $3(x_1 + x_2 + x_3) + 6(x_4 + x_5 + x_6) \geq 6 + 12(1 - x_7)$. One can check (either by hand or with the help of a package like `PORTA`) that the latter is facet-defining. Note that this inequality is not equivalent to a lifted cover inequality. □

3.4 Concluding Remarks

We have introduced two lifting procedures for knapsack cover inequalities. Our examples show that it is possible for these lifting procedures to yield non-trivial facet-defining inequalities. An interesting extension to our work would be the design and implementation of efficient separation heuristics for LKCIs. It would also be interesting to compare LKCIs with lifted cover inequalities.

Chapter 4

New Valid Inequalities for the Fixed-Charge and Single-Node Flow Polytopes

4.1 Introduction

Polyhedral methods have proven to be remarkably useful for solving pure and mixed 0-1 linear programs (see, e.g., [20, 22]). In the case of large, sparse instances without special structure, three families of polytopes have proven to be of particular importance: the knapsack, fixed-charge and single-node flow polytopes. The *knapsack* polytope is the convex hull of vectors $y \in \{0,1\}^n$ satisfying

$$\sum_{i \in N} a_j y_j \leq b,$$

where b and the a_j are positive integers, and N denotes $\{1, \ldots, n\}$ [7, 69]. The *fixed-charge* polytope is the convex hull of pairs $(x, y) \in \mathbb{R}^n_+ \times \{0, 1\}^n$ satisfying

$$\sum_{j \in N} x_j \leq d \tag{4.1.1}$$

$$x_j \leq u_j y_j \qquad (j \in N), \tag{4.1.2}$$

where d and the u_j are positive integers [60]. Finally, the *single-node flow* polytope is the convex hull of pairs $(x, y) \in \mathbb{R}^n_+ \times \{0, 1\}^n$ satisfying

$$\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq d \tag{4.1.3}$$

$$\ell_j y_j \leq x_j \leq u_j y_j \qquad \left(j \in N^+ \cup N^-\right), \tag{4.1.4}$$

where d and the u_j are again positive integers, the ℓ_j are non-negative integers, N^+ and N^- are disjoint sets, and n now denotes $|N^+ \cup N^-|$ [63].

Fixed charges appear in numerous applications. For example, in production planning, there is often a fixed set-up cost to use a machine. In vehicle routing, there is usually a hiring cost to use a vehicle. Finally, in network design, there is an installation cost to deliver a commodity (e.g., gas, water, electricity) from point A to point B of a network.

Several families of valid linear inequalities are known for the knapsack polytope, including cover, extended cover and lifted cover inequalities [7, 69], weight inequalities [67] and lifted pack inequalities [5, 41]. Inequalities for the fixed-charge polytope include flow cover inequalities [60] and lifted flow cover inequalities [33, 34]. Inequalities for the single-node flow polytope include generalized flow cover inequalities [63, 64], reverse flow cover inequalities [66], lifted generalised flow cover inequalities [33] and

lifted flow pack inequalities [4]. Inequalities like these have proven to be so useful that many of them are now generated by default in the leading integer programming solvers (such as `CPLEX`, `Gurobi` and `Xpress`).

The purpose of this note is to present a procedure which enables one to take valid inequalities for the knapsack polytope and convert them into valid inequalities for the fixed-charge and single-node flow polytopes. We call the resulting inequalities *rotated knapsack inequalities* or RKIs. Even if one applies our procedure to simple inequalities for the knapsack polytope, such as cover inequalities, one can still obtain new and non-trivial inequalities for the other polytopes.

The paper is structured as follows. In Section 4.2, we review some of the well-known inequalities for our three families of polytopes. In Section 4.3, we present our procedure for the fixed-charge polytope, show that the resulting inequalities can define facets, and examine the special cases that arise when the initial inequality is a cover or extended cover inequality. In Section 4.4, we extend our procedure to the single-node flow polytope. Finally, Section 4.5 includes some concluding remarks and suggestions for further research.

4.2 Literature Review

We now review the literature. The following subsections are concerned with valid inequalities for the knapsack polytope, the fixed-charge polytope and the the single-node flow polytope.

4.2.1 Knapsack polytope

As mentioned above, many families of inequalities are known for the knapsack polytope. For brevity, we recall here only a few results from [7, 69]. A set $C \subseteq N$ is called a *cover* if $\sum_{j \in C} a_j > b$. If C is a cover, then the *cover* inequality $\sum_{j \in C} y_j \leq |C| - 1$ is valid. The strongest cover inequalities are obtained when C is *minimal* (i.e., no proper subset of C is a cover).

Cover inequalities can be strengthened in various ways. For example, let $a^* = \max_{j \in C} a_j$ and let $E = \{j \in N \setminus C : a_j \geq a^*\}$ be the *extension* of C. The *extended* cover inequality takes the form

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

4.2.2 Fixed-charge polytope

Padberg *et al.* [60] presented two families of inequalities for the fixed-charge polytope. The inequalities of the first family are derived as follows. A set $F \subseteq N$ is called a *flow cover* if $\sum_{j \in F} u_j > d$. Given a flow cover F and a (possibly empty) set $L \subseteq N \setminus F$, we let λ denote $\sum_{j \in F} u_j - d$ and u^+ denote $\max_{j \in F} u_j$. The following *flow cover* inequality is valid:

$$\sum_{j \in F \cup L} x_j \leq d - \sum_{j \in F} \alpha_j (1 - y_j) + \sum_{j \in L} \alpha_j y_j,$$

where α_j is $\max\{0, u_j - \lambda\}$ for $j \in F$, and $\max\{u^+, u_j\} - \lambda$ for $j \in L$. The flow cover inequalities were slightly strengthened in [33], yielding *lifted* flow cover inequalities.

The second family is very different. Let $P^=$ be the face of the fixed-charge polytope obtained by setting the inequality (4.1.1) to equality. One can check that, if $(x, y) \in$

$P^=$, then y must lie within the knapsack polytope

$$K = \text{conv}\Big\{y \in \{0,1\}^n : \sum_{j\in N} u_j y_j \geq d\Big\}.$$

Let $\alpha^T y \geq \beta$ be any valid inequality for K with $\alpha \in \mathbb{Z}^n_+$ and $\beta \in \mathbb{Z}_+$. It is shown in [60] that there exists a positive rational δ such that the inequality

$$\sum_{j\in N} x_j \leq d + \delta\Big(\sum_{j\in N} \alpha_j y_j - \beta\Big)$$

is valid for the fixed-charge polytope. To the best of our knowledge, this procedure has received no attention in the literature.

4.2.3 Single-node flow polytope

Van Roy & Wolsey [63] extended the flow cover inequalities to the single-node flow polytope. Now, a pair (F^+, F^-) is called a *generalised flow cover* if $F^+ \subseteq N^+$, $F^- \subseteq N^-$ and $\sum_{j\in F^+} u_j - \sum_{j\in F^-} \ell_j > d$. Given a generalised flow cover (F^+, F^-) and sets $L^+ \subseteq N^+ \setminus F^+$ and $L^- \subseteq N^- \setminus F^-$, one can construct a valid inequality of the form

$$\sum_{j\in F^+\cup L^+} x_j - \sum_{j\in N^-} x_j \leq d - \sum_{j\in F^+} \alpha_j(1-y_j) + \sum_{j\in L^+} \alpha_j y_j + \sum_{j\in F^-} \alpha_j(1-y_j) - \sum_{j\in L^-} \alpha_j y_j,$$

where $\alpha_j \in \mathbb{Z}_+$ for $j \in F^+ \cup L^+ \cup F^- \cup L^-$. (For brevity, we skip the details on how to compute the α_j.) These are called *generalised flow cover* (GFC) inequalities. They have been generalised and strengthened in various ways [33, 34, 66, 63]. A related family of inequalities, the lifted flow pack inequalities, were studied by Atamtürk [4].

4.3 Fixed-Charge Polytope

In this section, we consider the fixed-charge polytope. Subsection 4.3.1 presents our procedure for generating inequalities, Subsection 4.3.2 gives a sufficient condition for the resulting inequalities to define facets, and Subsection 4.3.3 analyses two special cases.

4.3.1 General procedure

Let P denote the fixed-charge polytope and $F \subseteq N$ be a flow cover. The inequality $\sum_{j\in F} x_j \leq d$ is trivially valid for P. Let $P^=$ be the face of P obtained by setting this inequality to equality. That is,

$$P^= := \text{conv}\Big\{(x,y) \in \mathbb{R}^n_+ \times \{0,1\}^n : \text{(4.1.1), (4.1.2)}, \sum_{j\in F} x_j = d\Big\}.$$

One can check that the following inequality is valid for $P^=$:

$$\sum_{j\in F} u_j y_j \geq d. \tag{4.3.1}$$

Let $\bar{y}_j = 1 - y_j$ for all $j \in F$. Then, we can write (4.3.1) as

$$\sum_{j\in F} u_j \bar{y}_j \leq \sum_{j\in F} u_j - d,$$

and define the *restricted knapsack polytope*

$$K = \text{conv}\Big\{\bar{y} \in \{0,1\}^F : \sum_{j\in F} u_j \bar{y}_j \leq \sum_{j\in F} u_j - d\Big\}. \tag{4.3.2}$$

Let $\alpha^T \bar{y} \leq \beta$ be a supporting valid inequality for K with $\alpha \in \mathbb{Z}^F_+$ and β a positive integer. (All non-trivial valid inequalities for K have this form; see [69].) Complementing

the $\bar{y}_j$ variables, and writing $\beta^* = \sum_{j\in F} \alpha_j - \beta$, we obtain the inequality

$$\sum_{j\in F} \alpha_j y_j \geq \beta^*, \tag{4.3.3}$$

which is valid and supporting for $P^=$. Our goal is to 'rotate' this inequality, in order to make it valid and supporting for P.

At this point, it is helpful to project P and $P^=$ onto a 2-dimensional subspace, having $\sum_{j\in F} x_j$ and the left-hand side of (4.3.3) as axes. This is illustrated in Figure 4.3.1. The thick vertical lines represent feasible solutions in P, and the horizontal line inside the ellipse at the top-right represents $P^=$. One can see that the inequality (4.3.3) is valid for $P^=$ but not for P. Moreover, given that the inequality (4.3.3) is supporting for $P^=$, any feasible solution satisfying $\sum_{j\in F} \alpha_j y_j < \beta^*$ also satisfies $\sum_{j\in F} x_j < d$. Therefore, there exists a positive rational number δ such that the inequality

$$\sum_{j\in F} x_j \leq d + \delta\Big(\sum_{j\in F} \alpha_j y_j - \beta^*\Big)$$

is valid and supporting for P. Setting δ to the largest such value yields the desired rotated knapsack inequality (RKI), which is represented by a dashed line in the figure.

In order to determine δ, we define the function $\phi : \mathbb{Z}_+ \to \{0, 1, \ldots, d\}$ with

$$\phi(t) = \max\Big\{ \sum_{j\in F} x_j : (4.1.1), (4.1.2), \sum_{j\in F} \alpha_j y_j \leq t, (x, y) \in \mathbb{R}^F_+ \times \{0, 1\}^F \Big\}.$$

By definition, ϕ is non-decreasing in t. Also, from the definition of β^*, and the fact that the inequality (4.3.3) is supporting for $P^=$, we have $\phi(t) = d$ if and only if $t \geq \beta^*$. Moreover, for $t = 0, \ldots, \beta^* - 1$, the constraint (4.1.1) is redundant, and we have:

$$\phi(t) = \max\Big\{ \sum_{j\in F} u_j y_j : \sum_{j\in F} \alpha_j y_j \leq t, y \in \{0, 1\}^F \Big\}.$$

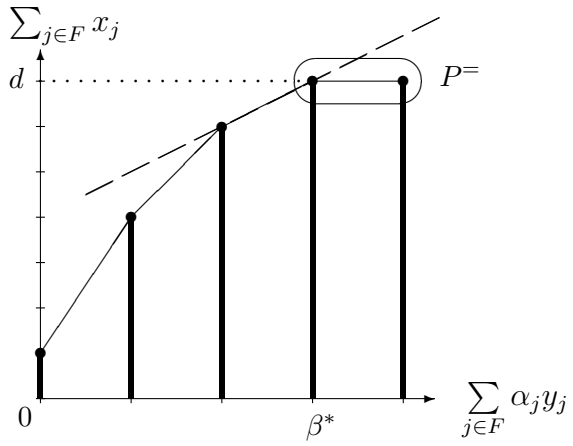

Figure 4.3.1: Projection of feasible points onto 2-dimensional subspace.

The problem of determining $\phi(t)$ for $t = 0, \ldots, \beta^* - 1$ is therefore a parametric 0-1 knapsack problem. It can be solved in $O\big(|F| \min\{d, \beta^*\}\big)$ time by dynamic programming. Once this has been done, one can compute δ easily, in $O(\beta^*)$ time, as follows:

$$\delta = \min_{0 \le t < \beta^*} \left\{ \frac{d - \phi(t)}{\beta^* - t} \right\}.$$

4.3.2 Facet-defining RKIs

A natural question is to determine conditions under which RKIs define facets of P. The following proposition gives a sufficient condition. (We remark that a similar condition was given in [19] for rank inequalities to define facets of stable set polytopes.)

Proposition 4.3.1. *Suppose the inequality $\sum_{j\in F} \alpha_j \bar{y}_j \le \beta$, with $\alpha \in \mathbb{Z}_+^F$ and β a positive integer, defines a facet of K. Let G be a graph defined as follows. The vertex set is F. For all pairs $\{i, j\} \subset F$, the edge $\{i, j\}$ is present in G if and only if there exists a vector $\bar{y} \in \{0, 1\}^F$ satisfying*

$$\bar{y}_i = \bar{y}_j = 0, \ \sum_{j\in F} \alpha_j \bar{y}_j = \beta, \ \textit{and} \ \sum_{j\in F} u_j \bar{y}_j < \sum_{j\in F} u_j - d.$$

If G is connected, then the resulting RKI defines a facet of P.

Proof. For brevity, we only sketch the proof. Let us call a pair $(x, y) \in \mathbb{R}^n_+ \times \{0,1\}^n$ a 'root' if it satisfies the RKI at equality. Note that the roots are of two types: (i) those with $\sum_{j\in F} \alpha_j y_j = \beta^*$, $\sum_{j\in F} x_j = d$, and $x_j = 0$ for $j \in N \setminus F$, and (ii) those with $\sum_{j\in F} \alpha_j y_j < \beta^*$ and $\sum_{j\in F} x_j < d$. Now, since the original inequality defines a facet of K, there exist $|F|$ affinely independent roots of the first type. Without loss of generality, we assume that these roots have $y_j = 0$ for $j \in N \setminus F$. So we can construct $n - |F|$ additional affinely independent roots of the first type, by changing y_j from 0 to 1 for each $j \in N \setminus F$ in turn. We now have n affinely independent roots of the first type.

Now, let T be a spanning tree in G, and note that it has $|F| - 1$ edges. From the definition of G, it follows that, for each edge $\{i, j\} \in T$, there exists a root of the first type, such that $y_i = y_j = 1$ and $\sum_{j\in F} u_j y_j > d$. Without loss of generality, we can assume that this root satisfies $x_i \in (0, u_i)$ and $x_j \in (0, u_j)$. If this root is not affinely independent of the roots that we have seen so far, then we can make it affinely independent by increasing x_i by some small quantity ϵ, and decreasing x_j by ϵ. In this way, we construct $|F| - 1$ additional affinely independent roots of the first type.

Next, observe that there exists a root of the second type for which $x_j = 0$ and $y_j = 1$ for all $j \in N \setminus F$, and this root is affinely independent of the previous ones. Moreover, for any $j \in N \setminus F$, we can construct another root of the second type by increasing x_j by some small quantity ϵ. In this way, we obtain an additional $n - |F|$

affinely independent roots. □

We illustrate this theory with an example.

Example 1: Let $n = 6$, $d = 10$ and $u = (1, 3, 3, 3, 5, 5)$. If we let $F = \{2, 3, 4, 5, 6\}$, we have

$$K = \left\{ \bar{y} \in \{0,1\}^F : 3\bar{y}_2 + 3\bar{y}_3 + 3\bar{y}_4 + 5\bar{y}_5 + 5\bar{y}_6 \leq 9 \right\}.$$

The inequality

$$\bar{y}_2 + \bar{y}_3 + \bar{y}_4 + 2\bar{y}_5 + 2\bar{y}_6 \leq 3$$

defines a facet of K. (In fact it is a so-called "general LCI", see [32, 41]). So the inequality $y_2 + y_3 + y_4 + 2y_5 + 2y_6 \geq 4$ is valid for $P^=$. One can check that $\phi(0) = 0$, $\phi(1) = u_2 = 3$, $\phi(2) = u_2 + u_3 = 6$, $\phi(3) = u_2 + u_3 + u_4 = 9$ and $\phi(4) = d = 11$. This yields $\delta = (11 - 9)/(4 - 3) = 2$. Therefore, the RKI

$$x_2 + x_3 + x_4 + x_5 + x_6 \leq 3 + 2(y_2 + y_3 + y_4 + 2y_5 + 2y_6)$$

is valid for P. Moreover, the graph G contains the edge $\{i, j\}$ for $i = 2, 3, 4$ and $j = 5, 6$, since $3 + 5 = 8 < 9$. So G is connected, and the RKI defines a facet of P. □

4.3.3 Special cases

We now consider the particular RKIs that are obtained when the given valid inequality for K is a cover or extended cover inequality. It turns out that, in both of these special cases, there is a closed-form expression for δ.

Let $F \subseteq N$ be a flow cover, and let K be defined as in (4.3.2). A set $C \subseteq F$ is a cover for K if and only if

$$\sum_{j \in C} u_j > \sum_{j \in F} u_j - d. \tag{4.3.4}$$

Moreover, a cover C is minimal if and only if

$$\sum_{j \in C \setminus \{k\}} u_j \leq \sum_{j \in F} u_j - d \qquad (\forall k \in C). \tag{4.3.5}$$

Given a minimal cover C, the cover inequality $\sum_{j \in C} \bar{y}_j \leq |C| - 1$ is valid and supporting for K. Complementing, we find that the inequality $\sum_{j \in C} y_j \geq 1$ is valid and supporting for $P^{=}$.

Now, if $\sum_{j \in C} y_j = 0$, the largest value that $\sum_{j \in F} u_j y_j$ can take is $\sum_{j \in F \setminus C} u_j$. Moreover, from (4.3.4), this quantity is less than d. So $\phi(0) = \sum_{j \in F \setminus C} u_j < d$. On the other hand, when $\sum_{j \in C} y_j = 1$, the largest value that $\sum_{j \in F} u_j y_j$ can take is $u^* + \sum_{j \in F \setminus C} u_j$, where $u^* = \max_{j \in C} u_j$. From (4.3.5), this quantity is larger than d. So $\phi(t) = d$ for $t = 1, \ldots, |C|$. This implies that $\delta = d - \sum_{j \in F \setminus C} u_j$, and the resulting RKI is

$$\sum_{j \in F} x_j \leq d + \left(d - \sum_{j \in F \setminus C} u_j \right) \left(\sum_{j \in C} y_j - 1 \right). \tag{4.3.6}$$

As for extended cover inequalities, we set $E = \{j \in F \setminus C : u_j \geq u^*\}$, and the extended cover inequality takes the form

$$\sum_{j \in C \cup E} \bar{y}_j \leq |C| - 1.$$

Complementing yields the following valid inequality for K:

$$\sum_{j \in C \cup E} y_j \geq 1 + |E|.$$

Using the same argument as before, we have $\phi(0) = \sum_{j \in F \setminus (C \cup E)} u_j$ and $\phi(t) = d$ for $t \geq |E| + 1$. Now, let $S(t)$ denote the sum of the t largest u_j values for $j \in E$. One can check that $\phi(t)$ equals $\phi(0) + S(t)$ for $t = 0, \ldots, |E|$. In particular, $\phi(|E|) = \sum_{j \in F \setminus C} u_j < d$. Now, observe that $\phi(t) - \phi(t-1) \geq u^*$ for $t = 1, \ldots, |E|$, but

$$\phi(|E| + 1) - \phi(|E|) = d - \sum_{j \in F \setminus C} u_j \leq u^*,$$

where the inequality on the right is implied by (4.3.5). So $\delta = d - \sum_{j \in F \setminus C} u_j$, and the resulting RKI is :

$$\sum_{j \in F} x_j \leq d + \Big(d - \sum_{j \in F \setminus C} u_j \Big) \Big(\sum_{j \in C \cup E} y_j - 1 - |E| \Big). \tag{4.3.7}$$

One can check that the RKI (4.3.7) dominates the RKI (4.3.6).

4.4 Single-Node Flow Polytope

We now extend our results to the single-node flow polytope, which we will again denote by P. This case turns out to be considerably more complicated. After some experimentation with the software package `PORTA` [17], we managed to find a procedure that yields inequalities that are facet-defining in several cases.

Let U^+ and L^+ be disjoint subsets of N^+, and let U^- and L^- be disjoint subsets of N^-. The sets U^-, L^+ and L^- are permitted to be empty, but U^+ must be non-empty.

Let $P^{\geq}$ be the convex hull of the feasible solutions that satisfy the inequality

$$\sum_{j \in U^+} x_j - \sum_{j \in L^-} x_j \geq d + \sum_{j \in U^-} u_j - \sum_{j \in L^+} \ell_j. \tag{4.4.1}$$

Since $P^{\geq}$ is contained in P, every $(x, y) \in P^{\geq}$ must satisfy the trivially valid inequality

$$\sum_{j \in U^+ \cup L^+} x_j \leq \sum_{j \in N^-} x_j + d. \tag{4.4.2}$$

Multiplying (4.4.1) by 2 and adding (4.4.2) we get

$$\sum_{j \in U^+ \cup N^-} x_j - \sum_{j \in L^+ \cup L^-} x_j - \sum_{j \in L^-} x_j \geq d + 2 \sum_{j \in U^-} u_j - 2 \sum_{j \in L^+} \ell_j.$$

Hence, all points in $P^{\geq}$ satisfy the inequality

$$\sum_{j \in U^+ \cup (N^- \setminus L^-)} x_j - \sum_{j \in L^+ \cup L^-} x_j \geq d + 2 \sum_{j \in U^-} u_j - 2 \sum_{j \in L^+} \ell_j.$$

Weakening this using (4.1.4), we find that all points in $P^{\geq}$ satisfy

$$\sum_{j \in U^+ \cup (N^- \setminus L^-)} u_j y_j - \sum_{j \in L^+ \cup L^-} \ell_j y_j \geq d + 2 \sum_{j \in U^-} u_j - 2 \sum_{j \in L^+} \ell_j.$$

Now, as before, let $\bar{y}_j$ denote $1 - y_j$. Also let R^- denote $N^- \setminus (U^- \cup L^-)$. All points in $P^{\geq}$ satisfy:

$$\sum_{j \in U^+ \cup U^- \cup R^-} u_j \bar{y}_j + \sum_{j \in L^+ \cup L^-} \ell_j y_j \leq \sum_{j \in U^+ \cup R^-} u_j - \sum_{j \in U^-} u_j + 2 \sum_{j \in L^+} \ell_j - d. \tag{4.4.3}$$

We now define a knapsack polytope, K, as the convex hull of pairs $(y, \bar{y}) \in \{0, 1\}^{U^+ \cup U^- \cup R^- \cup L^+ \cup L^-}$ that satisfy (4.4.3). We remark in passing that a necessary condition for K to be full-dimensional is

$$\sum_{j \in U^+ \cup R^-} u_j + 2 \sum_{j \in L^+} \ell_j > d + \sum_{j \in U^-} u_j.$$

Next, let

$$\sum_{j \in U^+ \cup U^- \cup R^-} \alpha_j \bar{y}_j + \sum_{j \in L^+ \cup L^-} \beta_j y_j \leq \gamma.$$

be a supporting valid inequality for K with non-negative coefficients. Complementing yields

$$\sum_{j \in U^+ \cup U^- \cup R^-} \alpha_j y_j - \sum_{j \in L^+ \cup L^-} \beta_j y_j \geq \sum_{j \in U^+ \cup U^- \cup R^-} \alpha_j - \gamma.$$

To simplify the notation, let $\gamma^* = \sum_{j \in U^+ \cup U^- \cup R^-} \alpha_j - \gamma$.

As in Section 4.3, there exists a largest positive rational δ such that the inequality

$$\sum_{j \in U^+} x_j - \sum_{j \in L^-} x_j \leq d + \sum_{j \in U^-} u_j - \sum_{j \in L^+} \ell_j + \delta \left(\sum_{j \in U^+ \cup U^- \cup R^-} \alpha_j y_j - \sum_{j \in L^+ \cup L^-} \beta_j y_j - \gamma^* \right)$$

is valid for P. This inequality is the desired RKI.

As before, in order to determine the value of δ, we define an auxiliary function. For $t \in \mathbb{Z}$, let

$$\phi(t) = \max \left\{ \sum_{j \in U^+} x_j - \sum_{j \in L^-} x_j : \text{(4.1.3), (4.1.4)}, \sum_{j \in U^+ \cup U^- \cup R^-} \alpha_j y_j - \sum_{j \in L^+ \cup L^-} \beta_j y_j \leq t,\ x_j \geq 0,\ y_j \text{ binary} \right\}.$$

Again, $\phi(t)$ is an integer-valued, non-decreasing function, but its natural domain is

$$\mathbb{Z} \cap \left[-\sum_{L^+ \cup L^-} \beta_j, \sum_{j \in U^+ \cup U^- \cup R^-} \alpha_j \right].$$

One can compute the values taken by $\phi(t)$ over this domain efficiently by dynamic programming. (Details omitted for brevity.) Once the $\phi(t)$ values have been computed, one can compute the value δ as follows. Let t^* be the minimum value of t such that $\phi(t) = d + \sum_{j \in U^-} u_j - \sum_{j \in L^+} \ell_j$. (Note that $t^* \geq \gamma^*$.) Then let

$$\delta = \min_{-\sum_{j \in L^+ \cup L^-} \beta_j \leq t < t^*} \left\{ \frac{d - \phi(t)}{\gamma^* - t} \right\}.$$

Again, we illustrate this theory with an example.

Example 2: Let $n = 7$, $N^+ = \{1, 2, 3\}$, $N^- = \{4, 5, 6, 7\}$, $d = 4$, $u = (4, 3, 3, 2, 2, 2, 2)$ and $\ell = (1, 1, 1, 1, 1, 1, 1)$. Suppose we set $U^+ = \{2, 3\}$, $L^+ = \emptyset$, $U^- = \emptyset$ and $L^- = \{4\}$.

The knapsack constraint is $3y_2 + 3y_3 - y_4 + 2y_5 + 2y_6 + 2y_7 \geq 4$. Complementing gives $3\bar{y}_2 + 3\bar{y}_3 + y_4 + 2\bar{y}_5 + 2\bar{y}_6 + 2\bar{y}_7 \leq 8$. The inequality $2\bar{y}_2 + 2\bar{y}_3 + y4 + \bar{y}_5 + \bar{y}_6 + \bar{y}_7 \leq 5$ is valid for K. (Again, it is a non-simple LCI.) So the inequality $2y_2 + 2y_3 - y_4 + y_5 + y_6 + y_7 \geq 2$ is valid for $P^{\geq}$. We have $\phi(-1) = -1$, $\phi(0) = 0$, $\phi(1) = 2$, $\phi(2) = 3$ and $\phi(3) = 4$. So $t^* = 3$ and $\delta = 1$, and we obtain the RKI $x_2 + x_3 - x_4 \leq 4 + \big(2y_2 + 2y_3 - y_4 + y_5 + y_6 + y_7 - 2\big)$. One can check that this RKI defines a facet of P. Other RKIs for this instance include, for example, the following:

$$\begin{aligned} x_1 + x_2 - x_4 - x_5 &\leq 4 + \big(3y_1 + 2y_2 - y_4 - y_5 + y_6 + y_7 - 3\big) \\ x_2 + x_3 - x_5 &\leq 4 + (2y_2 + 2y_3 + y_4 - y_5 + y_6 + y_7 - 3) \\ x_1 + x_3 - x_7 &\leq 4 + (2y_1 - y_2 + y_3 + y_4 + y_5 + y_6 - y_7 - 1). \end{aligned}$$

One can check that these too are facet-defining. □

4.5 Concluding Remarks

We have introduced new families of valid inequalities for the fixed-charge and single-node flow polytopes. The inequalities, called rotated knapsack inequalities (RKIs), are completely different from the well-known flow cover inequalites and variants. Note that our procedure can yield a huge number of RKIs, because the number of possible choices for the subsets U^+, U^-, etc. and the number of facets of the restricted knapsack polytope can both grow exponentially in the size of the problem. A natural topic for research is to find a necessary and sufficient condition for an RKI to be facet-

defining. Another pressing question is the development of effective exact or heuristic separation algorithms for the RKIs.

Chapter 5

Valid Inequalities for Mixed-Integer Programs with Fixed Charges on Sets of Variables

5.1 Introduction

It is well known that a wide range of important optimisation problems can be modelled as *mixed-integer linear programs* (MILPs). A key ingredient of modern exact MILP algorithms is the use of strong valid linear inequalities, also known as *cutting planes*, to strengthen the continuous relaxation of the problem (see, e.g., [20, 22]).

One strand of the literature on cutting planes is concerned with MILPs that involve *fixed charges* (see, e.g., [4, 33, 34, 37, 47, 52, 53, 54, 60, 66, 63, 64, 72]). A fixed charge is an additional cost that is incurred whenever a certain variable takes a positive value. The textbook way to model fixed charges is as follows. Suppose that x_j is a continuous

non-negative variable, and the fixed charge d_j is incurred whenever $x_j > 0$. Suppose also that we know an upper bound u_j on the value taken by x_j in an optimal solution. We define a new binary variable, say y_j, taking the value 1 if and only if the fixed charge is incurred. We then add $d_j y_j$ to the cost function, and add the linear constraint $x_j \leq u_j y_j$.

Unfortunately, when the textbook approach is used, the continuous relaxation of the resulting MILP is often very weak. In this situation, cutting planes are essential. In particular, the so-called *flow cover* inequalities [60, 63], and various extensions of them [33, 34, 54], have proven to be so effective that they are now generated by default not only in commercial MILP solvers (such as `CPLEX`, `Gurobi` and `Xpress`), but also in open-source solvers (such as `CBC` and `SCIP`).

In this paper, we consider a more general situation, in which fixed charges are associated with *sets* of variables. More precisely, we suppose that we have continuous variables $x_1, \ldots, x_n$, a collection of (not necessarily disjoint) sets $S_1, \ldots, S_m$ satisfying $\bigcup_{i=1}^{m} S_i = \{1, \ldots, n\}$, and a fixed charge d_i for $i = 1, \ldots, m$. The idea is that, for each i, the fixed charge d_i is incurred if $x_j > 0$ for *any* $j \in S_i$.

To see how this situation could arise in practice, consider a set of products that share a common machine. If any of the products are to be manufactured on a given day, then the machine must be set up at the start of that day. Another application can be found in vehicle routing, if, for example, there is a call-out charge for each vehicle. This situation could also arise in transportation problems if there are excess suppliers and there is a fixed charge if a supplier is used.

The paper has the following structure. Section 5.2 is a literature review. In Section

5.3, we consider the case in which the sets $S_1, \ldots, S_m$ are *nested*. We derive a family of valid inequalities for the associated polytope, and show that the inequalities both generalise and dominate a subclass of the flow cover inequalities. In Section 5.4, we extend some of our results to the general case, in which the sets can intersect in an arbitrary way. Throughout the paper, N and M denote $\{1, \ldots, n\}$ and $\{1, \ldots, m\}$, respectively.

5.2 Literature Review

We now review the relevant literature. In Subsection 5.2.1, we briefly review valid inequalities for fixed-charge problems. In Subsection 5.2.2, for reasons which will become clear later, we review some papers on what we call *optimality cuts*.

5.2.1 Valid inequalities for the fixed charge polytope

Padberg *et al.* [60] introduced the *fixed-charge polytope*, defined as the convex hull of pairs $(x, y) \in \mathbb{R}^n_+ \times \{0, 1\}^n$ satisfying

$$\sum_{j=1}^{n} x_j \leq d$$

$$x_j \leq u_j y_j \qquad (j = 1, \ldots, n),$$

where d and the u_j are positive integers.

A set $C \subseteq N$ is called a *cover* if $\sum_{j \in C} u_j > d$. Given a cover C, we let λ denote the "excess capacity" $\sum_{j \in C} u_j - d$, we let u^+ denote $\max_{j \in C} u_j$, and we define the set $C^* = \{j \in C : u_j > \lambda\}$. Padberg *et al.* showed that, given a cover C and a (possibly

empty) set $L \subseteq N \setminus C$, the following *flow cover inequality* (FCI) is valid:

$$\sum_{j \in C \cup L} x_j \leq d - \sum_{j \in C^*} (u_j - \lambda)(1 - y_j) + \sum_{j \in L} \alpha_j y_j,$$

where α_j is $\max\{u^+, u_j\} - \lambda$ for $j \in L$. As a special case, when $L = \emptyset$ we obtain:

$$\sum_{j \in C} x_j \leq d - \sum_{j \in C^*} (u_j - \lambda)(1 - y_j). \tag{5.2.1}$$

We will call inequalities of this type *simple* FCIs.

The FCIs were extended to *fixed-charge network flow* (FCNF) problems in [63]. They have been further generalised and strengthened in various ways [33, 34, 66, 63]. A related family of inequalities, the *flow pack* inequalities, were studied by Atamtürk [4]. For a very different family of inequalities, derived from knapsack polytopes, see our recent paper [47].

5.2.2 Optimality cuts

By an *optimality* cut, we mean a linear inequality that is satisfied by all *optimal* MILP solutions, but may be violated by one or more sub-optimal solutions. Note that some authors expand this definition to include linear inequalities that are satisfied by *at least one* optimal solution. This concept appears, sometimes under different names, in many places (e.g., [8, 11, 25, 26, 37, 40, 44, 57]).

Among these works, Hooker *et al.* [37] is itself concerned with FCNF problems. Consider a directed network in which, for each arc a, there is a continuous flow variable x_a and a binary fixed-charge variable y_a. Consider a node i. Let a be an arc entering node i, and let $\alpha, \ldots, \omega$ be the arcs leaving i. Suppose that $y_a = 1$ in an optimal

solution. Then x_a must be positive, since, if that were not so, we could have made a cost saving by setting y_a to 0. The flow entering node i via arc a must then exit node i via one or more of the outgoing arcs. This in turn implies that at least one of the y variables associated with the outgoing arcs must take the value 1. Thus, the inequality $y_a \leq y_\alpha + \cdots + y_\omega$ is an optimality cut (called a *logic cut* in [37]).

5.3 The Nested Case

In this section, we consider the special case in which the sets $S_1, \ldots, S_m$ are *nested* (that is, for all pairs $\{i, i'\} \subset M$, we have $S_i \cap S_{i'} = \emptyset$, $S_i \subset S_{i'}$ or $S_{i'} \subset S_i$). The section is organised as follows. In Subsection 5.3.1, we give some notation and terminology. In Subsection 5.3.2, we prove a negative complexity result. In Subsection 5.3.3, we show that our problem is closely related to FCNF problems. In Subsection 5.3.4, we present our valid inequalities. Finally, in Subsection 5.3.5, we consider the effect on our inequalities when optimality cuts are present.

5.3.1 Notation and terminology

From now on, we let P denote the convex hull of the pairs $(x, y) \in \mathbb{R}^n_+ \times \{0,1\}^m$ satisfying

$$\sum_{j \in S_i} x_j \leq u_i y_i \ (i = 1, \ldots, m). \tag{5.3.1}$$

Note that, in the nested case, we can assume without loss of generality that $S_m = N$. Indeed, if there did not exist some set S_i that contained all the other sets, then P would be the Cartesian product of simpler polytopes of the same kind.

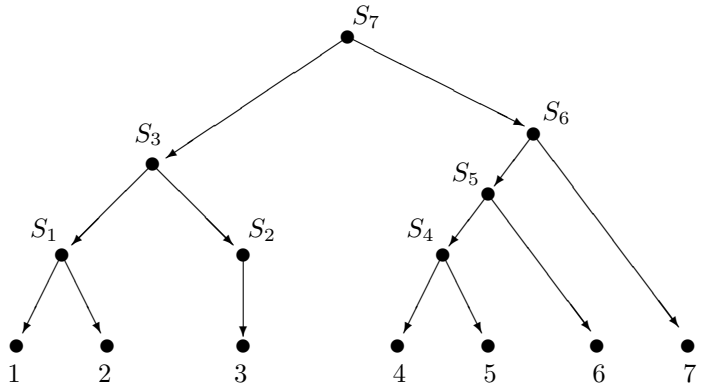

Figure 5.3.1: Visualisation of nested sets using a directed tree.

Given $i, i' \in M$, we will call i a *descendant* of i' if $S_i \subset S_{i'}$. If, in addition, there is no $k \in M$ such that $S_i \subset S_k \subset S_{i'}$, we call i a *child* of i'. We let $\delta(i)$ and $\chi(i)$ denote the set of descendants and children, respectively, of i. We also let $\rho(i)$ denote $S_i \setminus \bigcup_{k \in \chi(i)} S_k$. (Note that at most one of $\chi(i)$ and $\rho(i)$ can be empty.) We also represent the sets $S_1, \ldots, S_m$ as a rooted directed tree, as follows. There is one node for each set S_i, and one node for each $j \in N$. There is an arc from S_i to $S_{i'}$ if i' is a child of i, and there is an arc from S_i to j if $j \in \rho(i)$.

The above notation and terminology is illustrated in the following example.

Example 1. Let $n = 7$ and $m = 7$, and suppose that $S_1 = \{1, 2\}$, $S_2 = \{3\}$, $S_3 = \{1, 2, 3\}$, $S_4 = \{4, 5\}$, $S_5 = \{4, 5, 6\}$, $S_6 = \{4, 5, 6, 7\}$ and $S_7 = \{1, \ldots, 7\}$. The corresponding tree is shown in Figure 5.3.1. One can check that, for example, 4 is a descendent of 6 and 1 is a child of 3. One can also check that $\delta(2) = \emptyset$, $\delta(6) = \{4, 5\}$, $\chi(3) = \{1, 2\}$, $\chi(4) = \emptyset$, $\rho(3) = \emptyset$ and $\rho(5) = \{6\}$. □

5.3.2 Complexity

Our first result is that, even in the nested case, optimisation over P is difficult.

Proposition 5.3.1. *Even in the nested case, optimising a linear function over P is $\mathcal{NP}$-hard.*

Proof. We reduce `SUBSET SUM`, shown to be $\mathcal{NP}$-complete in [27], to the problem in question. An instance of `SUBSET SUM` is given by positive integers $p_1, \ldots, p_k$ and a "target" t. The task is to determine if there exists a subset of $\{p_1, \ldots, p_k\}$ whose sum is t. To convert this to our problem, set n to k and m to $k+1$. For $i \in N$, set S_i to $\{i\}$, set u_i to p_i, give x_i a profit of 1 and give y_i a cost of $p_i(1-\epsilon)$, where ϵ is a small positive quantity. Finally, set S_m to N, u_m to t, and give y_m a cost of zero. Since all u_i values are integers, there exists an optimal solution in which x is integral. Then, for any given $i \in N$, it is not worth setting y_i to 1 unless x_i is set to p_i, in which case we gain a profit of ϵp_i. So the answer to `SUBSET SUM` is "yes" if and only if there is a point $(x, y) \in P$ with profit ϵt. (A suitable value for ϵ is $1/n$.) □

In light of this result, we cannot expect to obtain a complete description of P in terms of linear inequalities. So, we must be content with a partial linear description.

5.3.3 Relation to FCNF

Given a set of nodes V and a set of arcs $\mathcal{A}$, the FCNF polytope is the convex hull of the pairs $(x, y) \in \mathbb{R}_+^{|\mathcal{A}|} \times \{0,1\}^{|\mathcal{A}|}$, such that

$$\sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = b_i \qquad (i \in V)$$

$$x_{ij} \leq u_{ij} y_{ij} \qquad ((i,j) \in \mathcal{A}).$$

Next, we will show that, in the nested case, there is a strong connection between the polytope P and FCNF problems. Consider Figure 5.3.1 once more. For $i \in M$, let f_i denote the flow entering the node that represents S_i, and set the capacity of the corresponding arc to u_i. We also interpret the binary variable y_i as an indicator of whether that arc is being used or not. Finally, for $j \in N$, we let node j be a "sink" node, and we interpret x_j as representing the flow entering node j. By construction, a triple $(x, y, f) \in \mathbb{R}_+^n \times \{0,1\}^n \times \mathbb{R}_+^n$ represents a feasible flow if and only if (x, y) satisfies (5.3.1).

This observation is formalised in the following proposition.

Proposition 5.3.2. *Suppose that $S_1, \ldots, S_m$ are nested. Let $M' = \{i \in M : \chi(i) = \emptyset\}$, and let P^f be the convex hull of the triples $(x, y, f) \in \mathbb{R}_+^n \times \{0,1\}^m \times \mathbb{R}_+^m$ satisfying*

$$f_i = \textstyle\sum_{j \in S_i} x_j \qquad (i \in M') \tag{5.3.2}$$

$$f_i = \textstyle\sum_{k \in \chi(i)} f_k + \sum_{j \in \rho(i)} x_j \quad (i \in M \setminus M') \tag{5.3.3}$$

$$f_i \leq u_i y_i \qquad (i \in M). \tag{5.3.4}$$

Then (a) P^f is an FCNF polytope of a directed acyclic graph, (b) P is the projection of P^f onto (x, y)-space, and (c) P is affinely congruent to P^f.

Proof. By construction, the equations (5.3.2) and (5.3.3) enforce conservation of flow in a directed acyclic graph, and the constraints (5.3.4) enforce arc capacities, as well as ensuring that no flow can pass through an arc unless the arc is open. This proves claim (a). Now, by conservation of flow, we must have $f_i = \sum_{j \in S_i} x_j$ for all $i \in M$. Thus, if we eliminate the f variables, using the equations (5.3.2) and (5.3.3), the constraints (5.3.4) reduce to (5.3.1). This proves claim (b). To see that claim (c) holds, it suffices to note that there is an invertible affine transformation which maps each extreme point of P to an extreme point of P_f. This transformation consists of leaving x and y unchanged, and setting f_i to $\sum_{j \in S_i} x_j$ for all $i \in M$. □

This link with FCNF polytopes has an interesting consequence:

Corollary 5.3.3. *Consider the following set function, which maps sets $T \subseteq N$ to $\mathbb{Z}_+$:*

$$\phi(T) = \max \left\{ \sum_{j \in T} x_j : \sum_{j \in S_i} x_j \leq u_i \, (i \in M), \; x_j \geq 0 \, (j \in N) \right\}. \tag{5.3.5}$$

If the sets $S_1, \ldots, S_m$ are nested, then ϕ is submodular.

Proof. Wolsey ([72], Th. 4) proved the following. Let $G = (V, A)$ be a directed graph, let $u \in \mathbb{R}^A_+$ be a vector of arc capacities, and let $p \in \mathbb{R}^A_+$ be a vector of arc profits. Given any $S \subset V$, the maximum-profit flow through the arcs leaving S is a submodular function of the set of arcs that are open. To apply this result to our problem, it suffices to set S to the set of nodes representing $S_1, \ldots, S_m$, and set p_j to 1 for each arc that connects a node in $S_1, \ldots, S_m$ to a node in N. □

5.3.4 The new inequalities

From now on, we assume w.l.o.g. that, for each $i \in M$ such that $\rho(i) = \emptyset$, the condition $u_i \leq \sum_{k \in \chi(i)} u_k$ holds. (If it did not hold, then one could decrease u_i without losing any feasible solutions.)

Now, let $T \subseteq N$ be any set such that $\phi(T) = u_i$ for some $i \in M$. The following inequality is trivially valid and supporting for P:

$$\sum_{j \in T} x_j \leq u_i. \tag{5.3.6}$$

We will use sequential lifting [58, 70] to strengthen (5.3.6). More precisely, let $I = \big\{k \in M : S_k \cap T \neq \emptyset\big\}$, let c denote $|I|$, and let $s(1), \ldots, s(c)$ be an arbitrary ordering of the elements of I. We will compute a valid inequality of the form

$$\sum_{j \in T} x_j \leq u_i - \sum_{k=1}^{c} \beta_k \big(1 - y_{s(k)}\big), \tag{5.3.7}$$

where the (non-negative) coefficients β_k are computed according to the given ordering.

The coefficients β_k can be calculated as follows. Let $\phi(T, k)$ denote the maximum value that $\sum_{j \in T} x_j$ can take when $y_{s(k)} = 0$. If $\phi(T, k) = u_i$, then $\beta_k = 0$. Otherwise, let $\Delta(k) = \Big\{i \in \{1, \ldots, k-1\} : s(i) \in \delta\big(s(k)\big)\Big\}$ and set

$$\beta_k = u_i - \phi(T, k) - \sum_{\ell \in \Delta(k)} \beta_\ell. \tag{5.3.8}$$

Note that, for fixed T and k, one can compute $\phi(T, k)$ and β_k in $O(m + n)$ time. Thus, for fixed T, one can compute all lifting coefficients in $O\big(m(m+n)\big)$ time. The following example shows that it is possible for this procedure to yield many inequalities that define facets of P. It also shows that it is possible for different lifting sequences

to lead to different facets.

Example 2. Suppose that $n = 3$, $m = 5$, $S_i = \{i\}$ for $i = 1, 2, 3$, $S_4 = \{1, 2\}$, $S_5 = \{1, 2, 3\}$, $u_i = 4$ for $i = 1, 2, 3$, $u_4 = 7$ and $u_5 = 10$. Taking $T = \{1, 2\}$, the inequality (5.3.6) is $x_1 + x_2 \leq 7$. We have $I = \{1, 2, 4, 5\}$. Lifting in the order $1, 2, 4, 5$, we obtain $\Delta(1) = \Delta(2) = \emptyset$, $\Delta(3) = \{1, 2\}$ and $\Delta(4) = \{1, 2, 3\}$. This yields $\beta_1 = \beta_2 = 7 - 4 = 3$, $\beta_3 = 7 - 0 - 3 - 3 = 1$ and $\beta_4 = 7 - 0 - 3 - 3 - 1 = 0$. The resulting valid inequality is

$$x_1 + x_2 \leq 3y_1 + 3y_2 + y_4. \tag{5.3.9}$$

One can check that different lifting orders yield seven additional inequalities:

$$\begin{aligned}
x_1 + x_2 &\leq 3y_1 + 3y_2 + y_5 \\
x_1 + x_2 &\leq 3y_1 + 4y_4 \\
x_1 + x_2 &\leq 3y_1 + 4y_5 \\
x_1 + x_2 &\leq 3y_2 + 4y_4 \\
x_1 + x_2 &\leq 3y_2 + 4y_5 \\
x_1 + x_2 &\leq 7y_4 \\
x_1 + x_2 &\leq 7y_5.
\end{aligned}$$

One can also check (either by hand or with the help of a software package such as `PANDA` [51]) that all eight lifted inequalities define facets of P. Finally, one can check that a further sixteen facets of P can be obtained by taking $T = \{1, 2, 3\}$ and using different lifting sequences. □

We also have the following lemma:

Lemma 5.3.4. *The inequalities (5.3.7) generalise and dominate the simple flow cover inequalities (5.2.1).*

Proof. Let n, d, u and C be given, as in Subsection 5.2.1. To put this into our framework, set m to $n+1$, and set S_i to $\{i\}$ for $i = 1, \ldots, n$. Also set S_m and N to $\{1, \ldots, n+1\}$, u_m to d, and T to C. We have $I = C \cup \{m\}$. Let r denote $d - \sum_{j \in C^*}(u_j - \lambda)$, and note that $r > 0$. One can check that, if we lift y_m last, we obtain:

$$\sum_{i=1}^{n} x_j \leq d - \sum_{j \in C^*}(u_j - \lambda)(1 - y_j) - r(1 - y_m).$$

This dominates (5.2.1). □

An interesting question is whether there can exist facet-defining inequalities of the form (5.3.7) that cannot be obtained by lifting sequentially.

5.3.5 Optimality cuts

We end this section with some remarks on optimality cuts. Note that if y_i takes the value 0 in an optimal solution, then y_k should also take the value 0 for all $k \in \chi(i)$. Therefore, one can strengthen the LP relaxation by adding the optimality cut $y_k \leq y_i$ for all $i, k \in M$ such that $k \in \chi(i)$. Given the connection between our problem and FCNF problems mentioned in Subsection 5.3.2, these can be viewed as a special kind of logic cuts.

Note that, once these optimality cuts have been added, the convex hull of the feasible pairs (x, y) is no longer an FCNF polytope. Nevertheless, our procedure for generating valid inequalities still applies. It turns out, however, that most lifting sequences no longer yield facets. Indeed, a necessary condition for obtaining a facet is that, for each $i \in I$ with $\delta(i) \neq \emptyset$, the descendants of i are lifted before i itself is lifted. The effect of this is that, for a given T, one of the valid inequalities dominates all of the others. This is illustrated in the following example.

Example 2 (cont.) We add the optimality cuts $y_1 \leq y_4$, $y_2 \leq y_4$, $y_3 \leq y_5$ and $y_4 \leq y_5$. Taking $T = \{1, 2\}$ and lifting in the order 1,2,4,5, as before, we obtain inequality (5.3.9). Together with the optimality cuts, this inequality dominates the other seven inequalities mentioned in Example 2. One can verify that it also defines a facet of the modified polytope. □

In other words, the optimality cuts eliminate some of the facets of the polytope P. The effect is to make the polytope simpler. Note that Lemma 5.3.4 still holds for the modified polytope.

5.4 The General Case

In this last section, we consider the general case, in which the sets S_i do not need to be nested. It turns out that this case is much more complicated, in several respects.

A first complication is that one can no longer assume that $S_m = N$. For example, if $n = 3$, $m = 2$, $S_1 = \{1, 2\}$ and $S_2 = \{2, 3\}$, then the polytope P is not the Cartesian

product of simpler polytopes.

A second complication is that optimising over P becomes even more difficult.

Proposition 5.4.1. *In the general case, optimising a linear function over P is $\mathcal{NP}$-hard in the strong sense.*

Proof. We reduce the maximum independent set problem, proven to be strongly $\mathcal{NP}$-hard in [27], to the problem in question. Let $G = (V, E)$ be a graph with n nodes and p edges. Set m to $n + p$. For $i = 1, \ldots, n$, set S_i to $\{i\}$, give x_i a profit of 3 and give y_i a cost of 2. For $r = 1, \ldots, p$, let S_{r+i} contain the end-nodes of the rth edge. Set all u values to 1. There exists an independent set of size k in G if and only if there is a solution with profit k. □

A third complication is that one can sometimes decrease one or more u values, by solving a series of LPs. This is shown in the following example.

Example 3. Let $n = 3$, $m = 7$, $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{3\}$, $S_4 = \{1, 2\}$, $S_5 = \{1, 3\}$, $S_6 = \{2, 3\}$ and $S_7 = \{1, 2, 3\}$. Also, let $u_1 = u_2 = u_3 = 2$, $u_4 = u_5 = u_6 = 3$ and $u_7 = 5$. Recall the function ϕ from Subsection 5.3.3. Solving the LP (5.3.5) with $T = S_7$, we find that $\phi(S_7) = 4.5$. Hence u_7 can be reduced from 5 to 4.5. □

This example also shows that, even when the original vector u is integral, it is possible for $\phi(T)$ to be fractional for a given $T \subseteq N$.

A fourth complication is that the polytope P is no longer an FCNF polytope in

general. In fact, the function ϕ is no longer submodular.

Example 3 (cont.) We have $\phi(S_4) = \phi(S_5) = 3$, $\phi(S_4 \cup S_5) = \phi(S_7) = 4.5$, and $\phi(S_4 \cap S_5) = \phi(S_1) = 2$. Since $3 + 3 < 4.5 + 2$, ϕ is not submodular. □

On the positive side, the function ϕ remains subadditive.

Proposition 5.4.2. *In the general case, ϕ is subadditive.*

Proof. Let S, T be subsets of N. Let x^* be the vector that maximises $\sum_{j \in S \cup T} x_j$ in (5.3.5). Let $x^1, x^2 \in \mathcal{X}$ be defined as follows

$$x^1_j = \begin{cases} x^*_j, \text{if } j \in S \\ 0, \text{if } j \notin S \end{cases} \qquad x^2_j = \begin{cases} 0, \text{if } j \in S \\ x^*_j, \text{if } j \notin S. \end{cases}$$

We have

$$\phi(S \cup T) = \sum_{j \in S \cup T} x^*_j = \sum_{j \in S} x^1_j + \sum_{j \in T \setminus S} x^2_j \leq \sum_{j \in S} x^1_j + \sum_{j \in T} x^2_j \leq \phi(S) + \phi(T).$$

□

A fifth complication is that the closed formula (5.3.8) that we presented in Subsection 5.3.3 can no longer be used to compute the lifting coefficients. Moreover, even when the correct coefficients are used, the resulting inequality is no longer guaranteed to be facet-defining.

Example 3 (cont.) Suppose we take $T = \{1, 2, 3\}$. The inequality (5.3.6) is $x_1 + x_2 + x_3 \leq 4.5$. We have $I = M$. If we lift $4, 5$ and 6 first, the formula (5.3.8) yields $\beta_1 = \beta_2 = \beta_3 = 4.5 - 2 = 2.5$, which yields the invalid inequality

$x_1 + x_2 + x_3 \leq -3 + 2.5(y_4 + y_5 + y_6)$. If we lift exactly (by solving a sequence of small mixed 0-1 LPs, as in [58, 70]), then we obtain $\beta_1 = 2.5$, $\beta_2 = 2$ and $\beta_3 = 0$. The resulting valid inequality $x_1 + x_2 + x_3 \leq 2.5y_4 + 2y_5$ does not define a facet of P. □

In fact, we do not know whether lifting can be performed in polynomial time in the general case, and we do not have a necessary and sufficient condition for a lifted inequality (5.3.7) to define a facet. On the positive side, it turns out that the lifted inequalities still define facets of P in many cases.

Example 3 (cont.) Suppose we change u_7 from 5 to 4. Taking $T = \{1, 2, 3\}$ again, the inequality (5.3.6) is $x_1 + x_2 + x_3 \leq 4$. Lifting in the order $1, 2, 3, 7, 4, 5, 6$, we get

$$x_1 + x_2 + x_3 \leq y_1 + y_2 + y_3 + y_7. \tag{5.4.1}$$

Lifting in the order $1, 2, 5, 6, 3, 4, 7$, we get

$$x_1 + x_2 + x_3 \leq y_1 + y_2 + y_5 + y_6. \tag{5.4.2}$$

One can check that these two inequalities define facets of P. One can also check that, using different lifting sequences, one can obtain an additional 42 facets. □

Finally, we mention that, as in the nested case, when optimality cuts are present, most lifting sequences no longer yield facets. Interestingly, however, we can now obtain more than one facet for a given T.

Example 3 (cont.) Suppose that, as before, $u_7 = 4$ and $T = \{1, 2, 3\}$. We add the optimality cuts $y_1 \leq y_4$, $y_1 \leq y_5$ and so on. One can check that the inequalities

(5.4.1) and (5.4.2) remain facet-defining, and so do the following two inequalities:

$$x_1 + x_2 + x_3 \leq y_1 + y_3 + y_4 + y_6$$

$$x_1 + x_2 + x_3 \leq y_2 + y_3 + y_4 + y_5.$$

□

An interesting open question is whether, in the general case, one can obtain facet-defining inequalities with fractional coefficients, even if the original u values are integers.

5.5 Acknowledgements

We thank Niamh Lamin for performing some preliminary polyhedral computations for us.

Chapter 6

Conclusion

6.1 Summary

As mentioned in Chapter 1, strong valid linear inequalities are a key component of modern exact algorithms for integer programming. In this thesis, we introduced a number of procedures to generate valid linear inequalities for certain specific problems. Firstly, we derived valid inequalities for the knapsack polytope. Secondly, we showed how to convert valid inequalities for the knapsack polytope into valid inequalities for the fixed-charge and single-node flow polytopes. Finally, we derived valid inequalities for mixed-integer programs with fixed charges on sets of variables.

In Chapter 2, we introduced a new procedure for lifting cover inequalities. More specifically, we showed how one of the earliest lifting procedures, due to Balas, can be significantly improved, so that it yields both stronger and more general LCIs, while still being very fast. The resulting procedure can yield facet-defining inequalities that cannot be obtained by standard lifting procedures. The procedure has a number of

other interesting properties, including that it can yield facet-defining inequalities even if the given cover is not minimal. Interestingly, we did not use superadditivity to prove that our lifting procedure is valid. However, we did use superadditivity to improve the lifting procedure further. The resulting associated lifting function is integer-valued almost everywhere.

In Chapter 3, we revisited the knapsack polytope, focusing on another family of inequalities, called knapsack cover inequalities. In general, these inequalities can be rather weak. To strengthen them, we used two fast approximate lifting procedures. The first procedure is based on a simple mixed-integer rounding argument. The second procedure is slightly more complicated and is based on the construction of a superadditive lifting function. Both procedures can yield new facet-defining inequalities for the knapsack polytope.

In Chapter 4, we introduced a new procedure for converting valid inequalities for the knapsack polytope into valid inequalities for the fixed-charge and single-node flow polytopes. The resulting inequalities are very different from previously known inequalities for these polytopes, and define facets under certain conditions. We have considered the particular inequalities that are obtained when the given inequality for the knapsack polytope is a cover or extended cover inequality. In both cases, we managed to get closed-form expressions. One very interesting result is that, even if one applies our procedure to simple inequalities for the knapsack polytope (such as cover inequalities), one can still obtain new and non-trivial inequalities for the other polytopes.

In Chapter 5, we considered mixed-integer programs with fixed charges on sets of

variables. We first considered the case in which the sets are nested and established a connection with fixed-charge network flow problems. We derived strong valid linear inequalities for this case, and showed that they generalise and dominate a subclass of the well-known flow cover inequalities for the classical fixed-charge problem. Then, we considered the general case, in which the set of variables do not need to be nested. We discussed the complications that arise, and extended some of our results.

6.2 Further Work

We believe that the work presented in this thesis has the potential to be extended in a number of ways. In this section, we outline some suggestions for further work that can stem from our work.

6.2.1 Lifted cover inequalities

In Chapter 2, we introduced a new procedure for lifting cover inequalities. The interesting properties of this procedure encourage us to believe that our work can be extended in a number of directions, from both polyhedral and computational perspectives.

We have come across several examples where our procedure yields facet-defining inequalities. So, an interesting question would be to derive necessary and/or sufficient conditions for our inequalities to be facet-defining. An even more interesting question would be to find a necessary and sufficient condition for the knapsack polytope to be completely described by our LCIs, together with the trivial bounds $0 \leq x_i \leq 1$.

We have proved that our procedure yields valid inequalities which are at least as strong as the ones generated by Balas' procedure, and we have come across several examples where our procedure yields stronger inequalities. One possible direction for further work would be a more extensive comparison of the two procedures. For example, one could try adding the two different cuts to the LP relaxation of the some test instances and compare the gap that each type of cuts closes. This can also be expanded to a cut-and-branch or even a branch-and-cut algorithm.

Furthermore, one could design a wide range of computational experiments to study our new lifting procedure in more detail. For example, one could add the new inequalities to existing solvers (such as CPLEX, Gurobi, SCIP, etc.) to test if the new inequalities improve the solvers' performance.

In order to use the inequalities in practice, one would need to design effective separation algorithms. One question is whether the separation problem for our inequalities can be solved exactly in pseudo-polynomial time. As for separation heuristics, one would have to bear in mind the fact that our lifting procedure can generate useful inequalities even when the cover is not minimal.

Finally, we showed that our procedure for lifting cover inequalities can be improved using superadditivity. Our examples demonstrate that the improved lifting functions can lead to stronger LCIs. It would be interesting to design some computational experiments to explore how much the additional improvement helps in practice.

6.2.2 Lifted knapsack cover inequalities

In Chapter 3, we revisited the knapsack polytope and introduced two lifting procedures for knapsack cover inequalities. One direction for further work would be to compare LKCIs with various kinds of lifted cover inequalities.

Our examples show that it is possible for these lifting procedures to yield non-trivial facet-defining inequalities. So, a natural extension to our work would be to derive necessary and/or sufficient conditions for the inequalities to be facet-defining.

Another potential extension to our work would be the design and implementation of efficient separation heuristics for LKCIs. Again, a question is whether the separation problem for either version of our LKCIs can be solved exactly in pseudo-polynomial time.

Finally, it would be worthwhile incorporating LKCIs (and some or all of the inequalities that we introduced in this thesis) into an exact algorithm for Fixed-Charge Network Flow problems. We believe that the LKCIs have the potential to perform very well in practice. Thus, it would be interesting to conduct some extensive computational experiments to study their performance in more detail.

6.2.3 Rotated knapsack inequalities

In Chapter 4, we introduced new families of valid inequalities for the fixed-charge and single-node flow polytopes. The inequalities, called rotated knapsack inequalities, are very different to the well-known flow cover inequalities. So, one direction for further work would be to compare rotated knapsack inequalities to flow cover inequalities.

Another idea for further work would be to add the new inequalities to solvers and test if the new inequalities can improve the solvers' performance. Note that our procedure can yield a huge number of RKIs. Thus, it is important to develop effective separation algorithms for the RKIs.

We have considered the special cases when the given inequality for the knapsack polytope is a cover or extended cover inequality. One could also try other particular families of valid inequalities for the knapsack polytope. For example, one could use the LCIs that we introduced in Chapter 2 as input.

Finally, due to the huge number of inequalities that this procedure can yield, another pressing question is to derive necessary and sufficient conditions for an RKI to define a facet of the single-node flow polytope.

6.2.4 Valid inequalities for problems with fixed charges on sets of variables

In Chapter 5, we introduced valid inequalities for mixed-integer problems with fixed charges on sets of variables. When the sets are nested, we have proved that our inequalities generalise and dominate simple flow cover inequalities. So, an natural question is how the two families of inequalities compare in practice.

It would also be interesting to see how well the new inequalities perform in practice. Hence, another potential direction for further work would be to test if the inequalities can improve existing solvers' performance. Note that this procedure can yield a huge number of inequalities. Hence, it is imperative to derive effective separation heuristics.

Again, an interesting question is whether the separation problem can be solved exactly in pseudo-polynomial time.

The non-nested case is much more complicated than the nested case. We have discussed some of the complications in Chapter 5. It would be interesting to study this case further and try to get a better understanding of the corresponding polytope.

Bibliography

[1] K. Aardal, Y. Pochet, and L. A. Wolsey. Capacitated facility location: valid inequalities and facets. *Math. Oper. Res.*, 20:562–582, 1995.

[2] E.H. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization.* Princeton University Press, Princeton, NJ, 2003.

[3] R.W. Ashford and R.C. Daniel. Some lessons in solving practical integer programs. *J. Oper. Res. Soc.*, 43:425–433, 1992.

[4] A. Atamtürk. Flow pack facets of the single node fixed-charge flow polytope. *Oper. Res. Lett.*, 29:107–114, 2001.

[5] A. Atamtürk. Cover and pack inequalities for (mixed) integer programming. *Ann. Oper. Res.*, 139:21–38, 2005.

[6] A. Atamtürk and M.W.P. Savelsbergh. Integer-programming software systems. *Ann. Oper. Res.*, 140:67–124, 2005.

[7] E. Balas. Facets of the knapsack polytope. *Math. Program.*, 8:146–164, 1975.

[8] E. Balas. Cutting planes from conditional bounds: a new approach to set covering. *Math. Program. Stud.*, 12:19–36, 1980.

[9] E. Balas and E. Zemel. Facets of the knapsack polytope from minimal covers. *SIAM J. Appl. Math.*, 34:119–148, 1978.

[10] R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[11] M. Bellmore and H.D. Ratliff. Set covering and involutory bases. *Mgmt. Sci.*, 18:194–206, 1971.

[12] E.A. Boyd. Generating Fenchel cutting planes for knapsack polyhedra. *SIAM J. Optim.*, 3:734–750, 1993.

[13] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2004.

[14] E.K. Burke and G. Kendall, editors. *Search Methodologies*. Springer US, New York, 2005.

[15] R.D. Carr, L.K. Fleischer, V.J. Leung, and C.A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of SODA XI*, pages 106–115. SIAM, New York, 2000.

[16] D.-S. Chen, R.G. Batson, and Y. Dang. *Applied Integer Programming*. Wiley, Hoboken, NJ, 2011.

[17] T. Christof and A. Loebl. PORTA (polyhedron representation transformation algorithm). software package, available for download at `http://www.iwr.uni-heidelberg.de/groups/comopt/software`.

[18] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discr. Math.*, 4:305–337, 1973.

[19] V. Chvátal. On certain polytopes associated with graphs. *J. Combin. Th. B*, 18: 138–154, 1975.

[20] M. Conforti, G. Cornuéjols, and G. Zambelli. Polyhedral approaches to mixed integer linear programming. In M. Juenger et al., editors, *50 Years of Integer Programming*, pages 343–385. Springer, Heidelberg, 2010.

[21] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer Programming*, volume 271 of *Graduate Texts in Mathematics*. Springer, 2015.

[22] W. Cook. Fifty-plus years of combinatorial integer programming. In M. Juenger et al., editors, *50 Years of Integer Programming*, pages 387–430. Springer, Heidelberg, 2010.

[23] H. Crowder, E. Johnson, and M.W. Padberg. Solving large-scale zero-one linear programming programs. *Oper. Res.*, 31:803–834, 1983.

[24] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[25] E. Fernández and K. Jørnsten. Partial cover and complete cover inequalities. *Oper. Res. Lett.*, 15:19–33, 1994.

[26] M. Fischetti, J.J. Salazar Gonzalez, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS J. Comput.*, 10:133–148, 1998.

[27] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[28] F. Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13:533 – 549, 1986.

[29] F. Glover. Unit-coefficient inequalities for zero-one programming. *Management Science Report 73-7*, University of Colorado, July 1973.

[30] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 64:275–278, 1958.

[31] R.E. Gomory. An algorithm for the mixed integer problem. Technical Report RAND-P-1885, Rand Corporation, Santa Monica, CA, 1960.

[32] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: computation. *INFORMS J. Comput.*, 10:427–437, 1998.

[33] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Lifted flow cover inequalities for mixed 0-1 integer programs. *Math. Program.*, 85:439–467, 1999.

[34] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Sequence-independent lifting in mixed integer programming. *J. Comb. Optim.*, 4:109–129, 2000.

[35] D. Hartvigsen and E. Zemel. The complexity of lifted inequalities for the knapsack problem. *Discr. Appl. Math.*, 39:113–123, 1992.

[36] K.L. Hoffman and M.W. Padberg. Improving LP-representations of zero-one linear programs for branch-and-cut. *ORSA J. Comput.*, 3:121–134, 1991.

[37] J.N. Hooker, H. Yan, I.E. Grossmann, and R. Raman. Logic cuts for processing networks with fixed charges. *Comput. & Oper. Res.*, 21:265–279, 1994.

[38] M. K. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. A branch-and-cut algorithm for the capacitated profitable tour problem. *Discr. Optim.*, 14:78–96, 2014.

[39] E.L. Johnson, G.L. Nemhauser, and M.W.P. Savelsbergh. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS J. Comput.*, 12:2–23, 2000.

[40] K. Kaparis and A.N. Letchford. Local and global lifted cover inequalities for the multidimensional knapsack problem. *Eur. J. Oper. Res.*, 186:91–103, 2008.

[41] K. Kaparis and A.N. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Math. Program.*, 124:69–91, 2010.

[42] K. Kaparis and A.N. Letchford. Cover inequalities. In J.J. Cochran et al., editors, *Encyclopedia of Operations Research and Management Science*. Wiley, New York, 2011.

[43] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller, J.W.

Thatcher, and J.D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, New York, 1972.

[44] G. Lancia, F. Rinaldi, and P. Serafini. Local search inequalities. *Discr. Optim.*, 16:76–89, 2015.

[45] A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[46] K.S. Lee. Separation heuristic for the rank-1 Chvátal-Gomory inequalities for the binary knapsack problem. *J. Korean Inst. Indust. Eng.*, 38:74–79, 2012.

[47] A.N. Letchford and G. Souli. New valid inequalities for the fixed-charge and single-node flow polytopes. *Oper. Res. Lett.*, 47:353–357, 2019.

[48] A.N. Letchford and G. Souli. On lifted cover inequalities: a new lifting procedure with unusual properties. *Oper. Res. Lett.*, 47:83–87, 2019.

[49] A.N. Letchford and G. Souli. Valid inequalities for mixed-integer programmes with fixed charges on sets of variables. *Oper. Res. Lett.*, 48:240–244, 2020.

[50] A.N. Letchford and G. Souli. Lifting the knapsack cover inequalities for the knapsack polytope. *Oper. Res. Lett.*, 48:607–611, 2020.

[51] S. Lörwald and G. Reinelt. PANDA: a software for polyhedral transformations. *EURO J. Comput. Optim.*, 3:297–308, 2015.

[52] Q. Louveaux and L.A. Wolsey. Lifting, superadditivity, mixed integer rounding and single node flow sets revisited. *Ann. Oper. Res.*, 153:47–77, 2007.

[53] H. Marchand and L.A. Wolsey. The 0-1 knapsack problem with a single continuous variable. *Math. Program.*, 85:15–33, 1999.

[54] H. Marchand and L.A. Wolsey. Aggregation and mixed-integer rounding to solve MIPs. *Oper. Res.*, 49:363–371, 2001.

[55] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988.

[56] G.L. Nemhauser and L.A. Wolsey. A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Math. Program.*, 46:379–390, 1990.

[57] P. Nobili and A. Sassano. A separation routine for the set covering polytope. In E. Balas, G. Cornuéjols, and R. Kannan, editors, *Proceedings of the 2nd IPCO Conference*, pages 201–219. CMU Press, Pittsburgh, PA, 1992.

[58] M.W. Padberg. A note on zero-one programming. *Oper. Res.*, 23:833–837, 1975.

[59] M.W. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.*, 6:1–7, 1987.

[60] M.W. Padberg, T.J. Van Roy, and L.A. Wolsey. Valid linear inequalities for fixed charge problems. *Oper. Res.*, 33:842–861, 1985.

[61] Y. Pochet and L.A. Wolsey. *Production Planning by Mixed Integer Programming*. Springer Science & Business Media, 2006.

[62] G. Polya. *How to Solve It: A New Aspect of Mathematical Method*. Princeton University Press, Princeton, NJ, 1945.

[63] T.J. Van Roy and L.A. Wolsey. Valid inequalities for mixed 0-1 programs. *Disc. Appl. Math.*, 14:199–213, 1986.

[64] T.J. Van Roy and L.A. Wolsey. Solving mixed integer programming problems using automatic reformulation. *Oper. Res.*, 35:45–57, 1987.

[65] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Chichester, 1998.

[66] J.I.A. Stallaert. The complementary class of generalized flow cover inequalities. *Disc. Appl. Math.*, 77:73–80, 1997.

[67] R. Weismantel. On the 0-1 knapsack polytope. *Math. Program.*, 77:49–68, 1997.

[68] H.P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons, Chichester, 5th edition, 2013.

[69] L.A. Wolsey. Faces for a linear inequality in 0–1 variables. *Math. Program.*, 8: 165–178, 1975.

[70] L.A. Wolsey. Facets and strong valid inequalities for integer programs. *Oper. Res.*, 24:367–372, 1976.

[71] L.A. Wolsey. Valid inequalities and superadditivity for 0-1 integer programs. *Math. Oper. Res.*, 2:66–77, 1977.

[72] L.A. Wolsey. Submodularity and valid inequalities in capacitated fixed charge networks. *Oper. Res. Lett.*, 8:119–124, 1989.

[73] L.A. Wolsey. *Integer Programming.* John Wiley & Sons, New York, 1998.

[74] E. Zemel. Easily computable facets of the knapsack polytope. *Math. Oper. Res.*, pages 760–765, 1989.