공학박사학위논문

Cutting-plane Generation Methods for Binary Knapsack Problem with Generalized Upper Bounds and Chance-constrained Binary Knapsack Problem

일반화된 상한제약이 있는 이진배낭문제와 확률제약이 있는 이진배낭문제에 대한 절단평면 생성 기법

2023 년 8 월

서울대학교 대학원

산업공학과

김 준 영

Cutting-plane Generation Methods for Binary Knapsack Problem with Generalized Upper Bounds and Chance-constrained Binary Knapsack Problem

일반화된 상한제약이 있는 이진배낭문제와 확률제약이 있는 이진배낭문제에 대한 절단평면 생성 기법

지도교수 이 경 식

이 논문을 공학박사 학위논문으로 제출함

2023 년 6 월

서울대학교 대학원

산업공학과

김 준 영

김준영의 공학박사 학위논문을 인준함

2023 년 6 월

위 원 장 홍 성 필 (인)

부위원장 이 경 식 (인)

위 원 홍 유 석 (인)

위 원 박 경 철 (인)

위 원 이 희 상 (인)

Abstract

Cutting-plane Generation Methods for Binary Knapsack Problem with Generalized Upper Bounds and Chance-constrained Binary Knapsack Problem

Junyoung Kim
Department of Industrial Engineering
The Graduate School
Seoul National University

Optimization problems with binary decision variables, known as binary integer programs, can represent a wide range of decision-making problems in various industries. Recent algorithmic advancements in general-purpose optimization solvers have significantly improved the ability to solve a binary integer program, making it a viable computational tool for addressing real-world operational issues. One of the critical factors in this success is the use of cutting planes for the binary knapsack problem. These cutting planes enhance the formulations of problems, providing tighter relaxation for the feasible solution sets by refining the solution space. Therefore, cutting planes improve the performance of relaxation-based optimization methods such as the branch-and-bound method.

However, as industries progress rapidly, more challenging issues have arisen, which can be represented by complicated binary integer programs. The current state-of-the-art solvers may not be sufficient to handle these problems, necessitating further improvements in their capabilities. These challenges have prompted studies on variants of the binary knapsack problem, such as those involving additional or nonlinear constraints, to derive more effective cutting planes and their generation methods that can be used in solving the binary integer programs.

In this thesis, we propose efficient cutting-plane generation methods for two variants of the binary knapsack problem: the binary knapsack problem with generalized upper bounds (GKP) and the chance-constrained binary knapsack problem (CKP). Our objective is to enhance the capability of solving binary integer programs by utilizing cutting planes generated from our methods.

Firstly, we investigate cutting planes for the GKP, which can be stronger than those for the binary knapsack problem. Specifically, we consider rank-1 Chvátal-Gomory (CG) cuts for the GKP, which are well-known cutting planes that can be defined for general integer linear programs. The separation problem that generates CG cuts is known to be strongly NP-hard for general integer programs. However, we show that it can be solved in pseudo-polynomial time for the GKP. We also devise an efficient heuristic for the separation problem based on its decomposition property. Through extensive computational tests, we demonstrate that the rank-1 CG cuts significantly improve the linear programming relaxation of binary integer linear programs, compared to existing cutting planes for the GKP, within a comparable computation time.

Then, we present a novel method to strengthen rank-1 CG cuts for binary integer

linear programs, improving the formulation-enhancing effect of the CG cuts. We first reveal the relationship between rank-1 CG cuts for binary integer linear programs and lifted cover inequalities for binary knapsack problems. Based on this result, our strengthening method derives a stronger cutting plane from a given rank-1 CG cut for binary integer linear programs by utilizing a lifting function of cover inequalities for binary knapsack problems. We extend the method to rank-1 CG cuts for binary integer linear programs with generalized upper bounds. Through theoretical comparison, we show that the cutting plane derived through the proposed strengthening method is stronger than those obtained through existing CG cut strengthening methods. Furthermore, computational test results show that the proposed methods can provide more enhanced formulations defined with fewer cutting planes and reduce the total computation time required to obtain the formulations.

Finally, we consider the CKP, which arises in the chance-constrained programming approach for optimization problems under uncertainty. We assume that the item weights are independently normally distributed. Then the CKP is formulated as a binary integer nonlinear program where the cutting planes for the binary knapsack problem are not valid. For the CKP, we propose an efficient lifting heuristic for its well-known cutting planes, the probabilistic cover inequalities. We first introduce a non-convex continuous relaxation for the CKP, represented as a non-convex optimization problem, and show that the relaxation provides tighter upper bounds than the existing continuous relaxations. In general, non-convex optimization problems are computationally hard to solve; however, we show that the non-convex relaxation can be solved in polynomial time. Subsequently, we devise a heuristic procedure for lifting probabilistic cover inequalities using the exact polynomial-time algorithm

for the non-convex continuous relaxation for the CKP. Computational test results demonstrate that the proposed lifting heuristic outperforms the existing methods in terms of computational efficiency, while the effectiveness of the resulting lifted probabilistic cover inequalities remains competitive.

Keywords: Binary integer program, Knapsack problem, Generalized upper bound, Chance constraint, General-purpose cut, Chvátal-Gomory cut, Cover inequality, Separation algorithm, Lifting

Student Number: 2017-23584

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

In this thesis, we propose cutting-plane generation methods for two variants of the binary knapsack problem: the binary knapsack problem with generalized upper bounds and the chance-constrained binary knapsack problem. Our objective is to enhance the capability of solving binary integer programs by utilizing the cutting planes generated from the proposed methods.

Binary integer programs represent optimization problems that involve binary decision variables. They can model a wide variety of decision-making problems in the real world, such as resource allocation and scheduling problems. Despite their broad applicability, binary integer programs were not considered viable computational tools to solve operational issues in real-world industries for a long time due to the lack of efficient solution approaches. However, in the last 20 years, significant algorithmic advancements in general-purpose optimization solvers have enabled solving practical issues in various industries using binary integer programs. According to the investigation by Bixby, Fenelon, et al. (2004), the most significant improvement was achieved by using general-purpose cutting planes as shown in Figure 1.1.

In optimization theory, cutting planes, or cuts, refer to valid inequalities, which

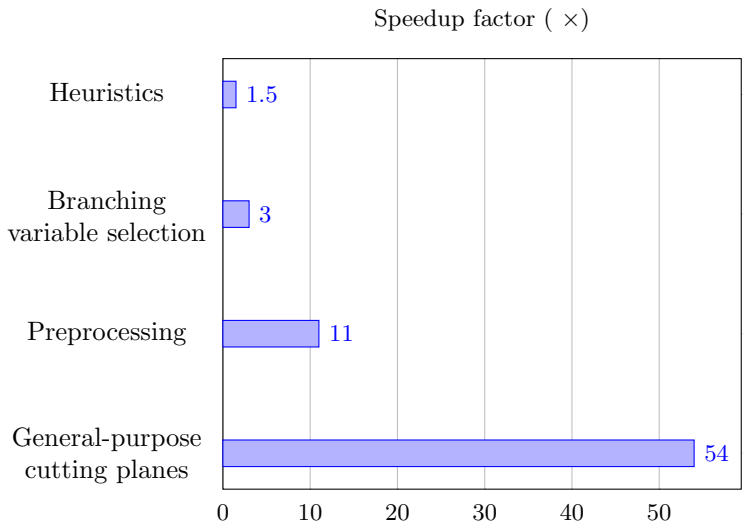


Figure 1.1: Speedup of optimization solvers by features (Bixby, Fenelon, et al., 2004)

are linear inequalities satisfied by all feasible solutions for a given optimization problem. Strictly speaking, cutting planes are a subset of valid inequalities, which can cut off some regions in the relaxation the problem formulation provides. However, we use the terms cutting planes and valid inequalities interchangeably throughout this thesis. By definition, cuts strengthen the relaxation provided by the problem formulation by refining the solution space, thus improving the performance of relaxation-based optimization methods such as the branch-and-bound algorithm (Land & Doig, 1960).

For specific binary integer programs, cuts can be derived from polyhedral studies on the feasible regions based on prior knowledge of the problems. On the other hand, general-purpose cuts refer to cuts used in solving general binary integer programs that have limited prior knowledge of the feasible solution sets. Therefore, general-purpose cuts used in practice have been typically derived from their relaxations that

are more amenable to analysis. One common relaxation considered in the literature is the single constraint relaxation, where the original problem is relaxed as an optimization problem with a single constraint and variable bounds (Dey & Tramontani, 2009).

For binary integer linear programs with linear constraints, the single-constraint relaxations can be defined with each constraint of the original problem or an implied linear inequality obtained by aggregating the constraints with non-negative multipliers. Then, if necessary, variables are complemented, and the single-constraint relaxation can be represented as the binary knapsack problem, which selects a subset of given items with non-negative weights and profits to maximize the total profit while satisfying the capacity constraint of a knapsack. In this regard, valid inequalities and their generation methods for the binary knapsack problem have been widely studied for the last few decades and successfully implemented in modern optimization software (Hojny et al., 2020). Most cuts used in practice for binary integer programs can be viewed as valid inequalities for the binary knapsack problem, and they have already become a standard feature of leading optimization software packages.

As industries progress rapidly, more challenging operational issues have arisen, which can be represented as complicated binary integer linear or even nonlinear programs. The current state-of-the-art solvers may not be sufficient to handle these problems, necessitating further improvements in their capabilities. These challenges have prompted studies on variants of the binary knapsack problem, such as those involving additional or nonlinear constraints, to derive more effective general-purpose cuts and their generation methods that can be used in solving binary integer programs.

In this context, the variants considered in this thesis have also been investigated. Firstly, the binary knapsack problem with generalized upper bounds has additional constraints where at most, one item should be selected among some of the given items, the so-called generalized upper bounds. This problem represents a tighter relaxation for the binary integer linear program with generalized upper bounds rather than the single-constraint relaxation. Therefore, studying this variant can derive stronger cuts and their generation methods that can be used in solving binary integer linear programs with generalized upper bounds.

The chance-constrained binary knapsack problem arises as a single-constraint relaxation in the chance-constrained programming approach for optimization problems with uncertain input data. In this problem, the items have uncertain weights with known probability distributions, and the capacity constraint is replaced with a chance constraint that limits the probability of selected items exceeding the capacity. The chance constraint is usually represented as a nonlinear inequality depending on the probability distributions of weights. Consequently, the valid inequalities for binary knapsack problems are not applicable to chance-constrained programs. Therefore, it is necessary to investigate this variant to derive valid inequalities and their generation methods to improve the solvability of chance-constrained programs.

This thesis mainly focuses on developing efficient cut-generation methods for these two variants of the binary knapsack problem. We demonstrate their effectiveness in improving the solvability of binary integer programs through extensive computational tests. In the remainder of this chapter, we present the background for our study and then give the organization of this thesis.

1.2 Background

1.2.1 Binary integer program

Integer programs represent optimization problem with integral decision variables, which can be defined as follows.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t} \quad & g_k(x) \leq 0, \ k \in R \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

where $c \in \mathbb{R}^n$ and $g_k : \mathbb{Z}^n \to \mathbb{R}$ for each $k \in R$. Here, $|R| = r$ and $N = \{1, ..., n\}$.

Binary integer programs are special cases of integer programs where all decision variables are binary. Throughout this thesis, the term "binary integer program" refers to the problems formulated above, where the integrality restriction is replaced with $x \in \{0, 1\}^n$.

When $g_k(x)$ is a linear function for all $k \in R$, we call the binary integer program as the *binary integer linear program*, which can be described as follows:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t} \quad & Ax \leq \mathbf{b} \\ & x \in \{0, 1\}^n, \end{aligned}$$

where $A \in \mathbb{Z}^{r \times n}$ and $\mathbf{b} \in \mathbb{Z}^r$. We consider the system $Ax \leq \mathbf{b}$ to include variable bound constraints, $x_j \leq 1$ for each $j \in N$. Although these constraints are redundant with the binary restriction, they are necessary for the brevity of notation in our

thesis. Due to the redundant variable bound constraints, the above formulation is valid even if the binary restriction is replaced with the integrality restriction, $x \in \mathbb{Z}_+^n$. On the other hand, when $g_k(x)$ is a nonlinear function for some $k \in R$, the binary integer program is referred to as the *binary integer nonlinear program*.

The binary integer program is NP-hard in a strong sense (Garey & Johnson, 1979), which means it might not be possible to solve the problem efficiently. However, an optimal solution can be found by a brute-force search for feasible solutions, although it may be impractical for large instances. In practice, the branch-and-bound algorithm (Land & Doig, 1960) is used as a systematic way to enumerate feasible solutions. This method iteratively breaks down the problem into sub-problems by partitioning the solution spaces while eliminating some of the sub-problems that cannot contain an optimal solution using lower and upper bounds for the problem. We note that tight bounds for the problem can significantly reduce the number of sub-problems to be solved in the branch-and-bound algorithm. Therefore, the bounds' quality crucially affects the performance of the branch-and-bound algorithm.

1.2.2 Cutting plane algorithm

As mentioned previously, cuts can be defined for general optimization problems. However, for consistency, we describe cuts in the context of the binary integer program.

Given a binary integer program defined in Section 1.2.1, let $\mathcal{X} \subseteq \{0,1\}^n$ be the feasible solution set, where $\text{conv}(\mathcal{X})$ is the convex hull of $\mathcal{X}$. Because $\mathcal{X}$ is a set of a finite number of points in $\mathbb{R}^n$, $\text{conv}(\mathcal{X})$ is a polytope which can be described with a finite number of linear inequalities. Let $\mathcal{P} \subseteq \mathbb{R}^n$ be the continuous relaxation for $\mathcal{X}$, obtained by relaxing the integrality restriction of the decision variables in the formulation.

In general, the branch-and-bound algorithm for solving the binary integer program utilizes the relaxation $\mathcal{P}$ to obtain the upper bound for the optimal objective value. By being close to $\text{conv}(\mathcal{X})$, $\mathcal{P}$ provides a tighter upper bound for the binary integer program, allowing the algorithm to terminate earlier without investigating an excessive number of sub-problems. In the extreme case where $\mathcal{P} = \text{conv}(\mathcal{X})$, an optimal solution for the binary integer program can be obtained by solving $\max\{c^T x : x \in \mathcal{P}\}$, without using the branch-and-bound algorithm. However, $\mathcal{P}$ strictly includes $\text{conv}(\mathcal{X})$ in most cases.

The gap between $\mathcal{P}$ and $\text{conv}(\mathcal{X})$ can be reduced using cuts by adding them to the formulation.

Definition 1.1. *For $(\alpha, \alpha_0) \in \mathbb{R}^{n+1}$, an inequality $\alpha^T x \leq \alpha_0$ is said to be a valid inequality for $\mathcal{X}$ if $\alpha^T \hat{x} \leq \alpha_0$ for all $\hat{x} \in \mathcal{X}$.*

Definition 1.2. *For $(\alpha, \alpha_0) \in \mathbb{R}^{n+1}$, a valid inequality $\alpha^T x \leq \alpha_0$ is said to be a cut*

for $\mathcal{X}$ *if* $\alpha^T \hat{x} > \alpha_0$ *for some* $\hat{x} \in \mathcal{P} \setminus \mathcal{X}$.

Strictly speaking, cuts are a subset of valid inequalities by definition. However, we use the terms cuts and valid inequalities interchangeably throughout this thesis.

Let $\alpha^T x \leq \alpha_0$ be a cut for $\mathcal{X}$. By adding this cut to the formulation for $\mathcal{X}$, we can obtain the continuous relaxation $\mathcal{P}_\alpha = \{x \in \mathcal{P} : \alpha^T x \leq \alpha_0\}$. Since $\mathcal{X} \subseteq \mathcal{P}_\alpha \subseteq \mathcal{P}$, $\mathcal{P}_\alpha$ represents an improved relaxation for $\mathcal{X}$, which provides a tighter upper bound for the optimal objective value of the binary integer program. We say that the cut *enhances* the original formulation for $\mathcal{X}$. Adding more cuts to the formulation can yield an even more enhanced formulation, providing a further tighter relaxation for $\text{conv}(\mathcal{X})$.

The effectiveness of cuts can be compared in terms of how much they close the gap between $\text{conv}(\mathcal{X})$ and $\mathcal{P}$. Suppose we have two cuts: $\alpha^T x \leq \alpha_0$ and $\beta^T x \leq \beta_0$, where $(\alpha, \alpha_0) \in \mathbb{R}^{n+1}$ and $(\beta, \beta_0) \in \mathbb{R}^{n+1}$ are linearly independent. Let $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$ be the improved relaxations derived from $\mathcal{P}$ with the cuts $\alpha^T x \leq \alpha_0$ and $\beta^T x \leq \beta_0$, respectively.

Definition 1.3. *The cut* $\alpha^T x \leq \alpha_0$ *is said to dominate the cut* $\beta^T x \leq \beta_0$ *if* $\mathcal{P}_\alpha \subset \mathcal{P}_\beta$.

We also denote the dominance relationship in terms of the strength of cuts. If a cut dominates another one, the cut is said to be *stronger* than the other. Checking dominance between cuts may not be straightforward because it involves the comparison of polyhedra. However, the following sufficient conditions for dominance allow us to compare cuts using only their coefficients.

Definition 1.4. *The cut* $\alpha^T x \leq \alpha_0$ *dominates the cut* $\beta^T x \leq \beta_0$ *if*

1. *There exists* $\mu \in \mathbb{R}_+$ *such that* $\mu\alpha \geq \beta$ *and* $\mu\alpha_0 \leq \beta_0$.

2. *The cut* $\beta^T x \leq \beta_0$ *can be represented as a non-negative linear combination of the cut* $\alpha^T x \leq \alpha_0$ *and* $x_j \leq 1$ *for each* $j \in N$.

A cut that is not dominated by other cuts is said to be *non-dominated*. By definition, it is sufficient to consider non-dominated cuts for enhancing the formulation. Non-dominated cuts define a minimal representation of $\text{conv}(\mathcal{X})$. In other words, non-dominated cuts are facet-defining inequalities for $\text{conv}(\mathcal{X})$, which are generally challenging to characterize. However, some families of non-dominated cuts can be derived through polyhedral studies of the given binary integer program.

Now, suppose that a family of cuts, $\mathcal{F}$, is given. The upper bound provided from the formulation enhanced by these cuts can be obtained by solving the following optimization problem.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t} \quad & \alpha^T x \leq \alpha_0, \ \forall (\alpha, \alpha_0) \in \mathcal{F} \\ & x \in \mathcal{P} \end{aligned}$$

Even if $\mathcal{P}$ is a polytope, this problem may be difficult to solve due to the huge number of cuts in $\mathcal{F}$. However, this problem can be solved efficiently using the cutting plane algorithm.

Rather than incorporating all the cuts in $\mathcal{F}$ at once, the cutting plane algorithm iteratively solves a restricted version of the above problem, considering only a subset of the cuts. We note that this restricted problem is the relaxation of the above problem. Once an optimal solution for the restricted problem is obtained, the oracle identifies a cut violated by the solution. If such a cut is found, it is added to redefine

the restricted problem. This process is iterated until the oracle no longer identifies any violated cuts. The solution obtained through the cutting plane algorithm is guaranteed to be optimal for the above problem because the solution is optimal for the relaxation. The overall algorithm is described in Algorithm 1.

Algorithm 1 Cutting plane algorithm

repeat
  $\hat{x} \leftarrow \arg\max\{c^T x : x \in \mathcal{P}\}$;
  $(\alpha, \alpha_0) \leftarrow$Find a cut in $\mathcal{F}$ such that $\alpha^T \hat{x} > \alpha_0$ through the oracle ;
  if $\exists(\alpha, \alpha_0)$ **then**
    $\mathcal{P} \leftarrow \{x \in \mathcal{P} : \alpha^T x \leq \alpha_0\}$;
  end if
until $\nexists(\alpha, \alpha_0)$

For each iteration, the oracle can identify a violated cut by solving the following problem.

$$\begin{aligned} \max \quad & \alpha^T \hat{x} - \alpha_0 \\ \text{s.t} \quad & (\alpha, \alpha_0) \in \mathcal{F} \end{aligned}$$

We refer to this problem as the *separation problem*. If the optimal objective value of the separation problem is greater than 0, we can obtain a cut that separates $\hat{x}$. Otherwise, $\hat{x}$ satisfies all the cuts in $\mathcal{F}$, and the cutting plane algorithm terminates.

The efficiency of the cutting plane algorithm heavily depends on the solution approach for the separation problem. Even if effective cuts are available, their impact may be limited without efficient separation algorithms. Therefore, to enhance the formulation using cuts, it is necessary to identify a set of effective cuts and develop efficient separation algorithms for them.

1.2.3 Binary knapsack problem

The binary knapsack problem involves selecting a subset of items with non-negative weights and profits to maximize the total profit while ensuring that the sum of the selected items' weights does not exceed a certain capacity. If the weights and profits of n items are denoted by $a_j \in \mathbb{R}+$ and $c_j \in \mathbb{R}+$, respectively, and the capacity is denoted by $b \in \mathbb{R}+$, the problem can be expressed as $\max\{\sum_{j\in N} c_j x_j : x \in \mathcal{X}_B\}$ where

$$\mathcal{X}_B = \left\{ x \in \{0,1\}^n : \sum_{j\in N} a_j x_j \leq b \right\}.$$

We refer to $\text{conv}(\mathcal{X}_B)$ as the binary knapsack polytope. Without loss of generality, we assume that a_j's and b are integers where $a_j \leq b$ for each $j \in N$.

The binary knapsack problem is of fundamental importance in the field of integer programming due to its combinatorial structure with general coefficients, making it a bridge between combinatorial optimization problems and integer programs. It also appears as a sub-structure in binary integer linear programs, leading to extensive research on cuts. Therefore, despite its simple structure, the literature on the binary knapsack problem is vast and widely distributed across various research directions, including solution approaches, computational complexity, and polyhedral studies. This section introduces valid inequalities for the binary knapsack problem. For other research directions, readers can refer to Martello & Toth (1990) and Kellerer et al. (2004). Furthermore, recent studies are covered in surveys by Hojny et al. (2020) and Cacchiani et al. (2022).

Various valid inequalities have been discovered in the literature for the binary knapsack problem. However, in this section, we mainly focus on a major family of

valid inequalities called *cover inequalities*, which are actively used in practice to solve binary integer programs. For information on other families, such as pack inequalities and their generation methods, we refer the reader to Kaparis & Letchford (2010).

Cover inequalities

A subset of items $C_B \subseteq N$ is called a *cover* if $\sum_{j \in C_B} a_j > b$. The following inequality,

$$\sum_{j \in C_B} x_j \leq |C_B| - 1,$$

is valid for $\mathcal{X}_B$, which we call a *cover inequality*. The cover inequality is the first valid inequality for the binary knapsack polytope, which is proposed independently by Balas (1975), Wolsey (1975), and Hammer et al. (1975). A cover C_B is said to be *minimal* if it does not contain a proper subset that is a cover, that is, $\sum_{j \in C_B \setminus \{k\}} a_j \leq b$ for all $k \in C_B$. Let $\lambda_B = \sum_{j \in C_B} a_j - b$. Then, the minimality condition can be rewritten as

$$\lambda_B \geq a_j, \ \forall j \in C_B.$$

The corresponding cover inequality is called a minimal cover inequality. We note that a minimal cover inequality defines a facet for the restricted binary knapsack polytope, which considers only variables that are contained in the cover, $\text{conv}(\mathcal{X}_B(C_B))$, where

$$\mathcal{X}_B(C_B) = \left\{ x \in \{0,1\}^{|C_B|} : \sum_{j \in C_B} a_j x_j \leq b \right\}.$$

The separation problem for cover inequalities is proven to be NP-hard by Klabjan et al. (1998); however, it can be formulated as a binary integer program as follows

(Crowder et al., 1983):

$$\begin{aligned} \min \quad & \sum_{j \in N} (1 - \hat{x}_j) y_j \\ \text{s.t} \quad & \sum_{j \in N} a_j y_j \geq b + 1 \\ & y \in \{0, 1\}^n \end{aligned}$$

If the optimal objective value of this problem is less than 1, the cover inequality that cuts off $\hat{x}$ can be constructed using the optimal solution y^*

Lifted cover inequalities

A cover inequality can be strengthened by improving the variable coefficients that are not contained in the cover. The resulting inequality is called a *lifted cover inequality* described as follows.

$$\sum_{j \in C_B} x_j + \sum_{j \in N \setminus C_B} \gamma_j x_j \leq |C_B| - 1 \tag{1.1}$$

An inequality of the form (1.1) may not be valid for the binary knapsack polytope depending on the lifting coefficients, γ_j's. Hence, we call an inequality of the form (1.1) as a lifted cover inequality only when (1.1) is valid. We now introduce several lifting techniques to determine the lifting coefficients of a lifted cover inequality from a given cover.

Padberg (1975) proposed a sequential lifting technique where the lifting coefficients are computed one by one in a given sequence of variables of $N \setminus C_B$. Recall that, for a given cover C_B, the cover inequality is valid for $\mathcal{X}_B(C_B)$ even if the cover

may not be minimal. Let $(j_1, ..., j_{|N \setminus C_B|})$ be the given variable sequence in $N \setminus C_B$ and $C_i = C_B \cup \{j_1, ..., j_i\}$ for each $i = 1, ..., |N \setminus C_B|$ where $C_0 = C_B$. Then, the lifting coefficients are determined as

$$\gamma^\star_{j_i} = |C_B| - 1 - \max_{x \in \mathcal{X}_B(C_i)} \left\{ \sum_{j \in C_{i-1}} \gamma^\star_j x_j : x_{j_i} = 1 \right\},$$

for each $i = 1, ..., |N \setminus C_B|$. Padberg (1975) also showed that the resulting lifted cover inequality,

$$\sum_{j \in C_B} x_j + \sum_{j \in N \setminus C_B} \gamma^\star_j x_j \leq |C_B| - 1,$$

defines a facet for $\text{conv}(\mathcal{X}_B)$ if C_B is minimal.

We refer to the optimization problem determining each lifting coefficient as the *lifting problem*. The lifting problem is a special case of the binary knapsack problem where the profits are less than or equal to n. Based on this observation, Zemel (1989) showed that all the lifting coefficients can be computed in $O(n|C_B|)$ using a dynamic programming algorithm for the binary knapsack problem. We note that the sequential lifting technique proposed by Padberg (1975) can be applied to valid inequalities for general binary integer programs, which are defined with only some of the variables (Zemel, 1978). Then, the lifting problem can be represented as a special case of the considered binary integer program.

For a given minimal cover, the sequential lifting technique can generate different facet-defining lifted cover inequalities for different variable sequences. However, generating all of them is intractable due to the enormous number of choices for variable sequences. Therefore, a variable sequencing strategy is necessary when applying the sequential lifting technique in practice. For an example of sequencing strategies, see

Gu et al. (1998).

On the other hand, *simultaneous lifting techniques* compute all lifting coefficients simultaneously, which may be faster than the sequential lifting technique. These methods utilize *lifting functions* to determine the lifting coefficients.

Definition 1.5. *A function $\psi : \mathbb{R} \to \mathbb{R}$ is a lifting function if the inequality*

$$\sum_{j \in C_B} x_j + \sum_{j \in N \setminus C_B} \psi(a_j) \leq |C_B| - 1$$

is valid for conv($\mathcal{X}_B$) for any minimal cover $C_B \subseteq N$.

The lifted cover inequality obtained from a lifting function is not guaranteed to define a facet for the binary knapsack polytope. However, some lifting functions may derive a facet-defining lifted cover inequality that cannot be obtained through the sequential lifting technique. For the sake of simplicity, let $C_B = \{1, ..., |C_B|\}$ be the minimal cover, and let the variables be sorted in non-increasing order of a_j. In addition, let $\mu_i = \sum_{j=1}^{i} a_j$ for each $i \in C_B$ and $\mu_0 = 0$. Balas (1975) proposed a lifting function defined as

$$\psi(z) = k \quad \text{if } \mu_k \leq z < \mu_{k+1},$$

for some $k = 0, ..., |C_B| - 1$. The author also showed that, if $a_j \leq \mu_{\psi(a_j)+1} - \lambda_B$ for all $j \in N \setminus C_B$, the resulting lifted cover inequality is the unique facet-defining inequality for conv($\mathcal{X}_B$) of the form (1.1). We note that the lifted cover inequality can be obtained in $O(n \log |C_B|)$.

Wolsey (1977) characterized a class of functions that can be used as a lifting

function. Let

$$\psi^{\star}(z) = |C_B| - 1 - \max\left\{ \sum_{j \in C_B} x_j : \sum_{j \in C_B} a_j x_j \leq b - z \right\}.$$

Theorem 1.1 (Wolsey, 1977)**.** *If $\psi(z)$ is superadditive and $\psi(z) \leq \psi^{\star}(z)$ for all $z \in [0, b]$, then $\psi(z)$ is a lifting function.*

Based on this theorem, several superadditive lifting functions have been proposed (Gu et al., 2000; Marchand, Martin, et al., 2002; Letchford & Souli, 2019). We only describe the lifting function proposed by Gu et al. (2000), which will be utilized in this thesis.

A lifting function h proposed by Gu et al. (2000) is defined as

$$h(z) = \begin{cases} k & \text{if } \mu_k - \lambda_B + \rho_k \leq z \leq \mu_{k+1} - \lambda_B \\ k - \frac{\mu_k - \lambda_B + \rho_k - z}{\rho_1} & \text{if } \mu_k - \lambda_B < z < \mu_k - \lambda_B + \rho_k \end{cases},$$

for some $k = 0, ..., |C_B| - 1$ where $\rho_k = \max\{0, a_{k+1} - (a_1 - \lambda_B)\}$ for all $k = 0, ..., |C_B| - 1$. It can be easily shown that $h(z)$ is greater than or equal to the lifting function proposed by Balas (1975). Therefore, the resulting lifted cover inequality,

$$\sum_{j \in C_B} x_j + \sum_{j \in N \setminus C_B} h(a_j) x_j \leq |C_B| - 1,$$

is stronger than the one derived by Balas (1975). Gu et al. (2000) showed that the above inequality is the unique facet-defining inequality for $\text{conv}(\mathcal{X}_B)$ of the form (1.1) if, for all $j \in N \setminus C_B$, $\mu_k - \lambda_B + \rho_k \leq a_j \leq \mu_{k+1} - \lambda_B$ for some $k = 0, \ldots, |C_B| - 1$. In other words, the lifted cover inequality defines a facet if the lifting coefficients are

integers. We note that $h(a_j)$'s can also be obtained in $O(n \log |C_B|)$.

General lifted cover inequalities

Lifted cover inequalities can be expressed in a general form as

$$\sum_{j \in C_B^\star} x_j + \sum_{j \in C_B \setminus C_B^\star} \gamma_j x_j + \sum_{j \in N \setminus C_B} \gamma_j x_j \leq |C_B^\star| - 1 + \sum_{j \in C_B \setminus C_B^\star} \gamma_j \qquad (1.2)$$

where $C_B^\star \subseteq C_B$. These inequalities are known as *general lifted cover inequalities* (Van Roy & Wolsey, 1987). When $C_B^\star = C_B$, general lifted cover inequalities are equivalent to lifted cover inequalities. The lifting coefficients that ensure the validity of (1.2) for $\text{conv}(\mathcal{X}_B)$ can also be computed using the sequential lifting technique.

The inequality $\sum_{j \in C_B^\star} x_j \leq |C_B^\star| - 1$ is valid for $\text{conv}(\mathcal{X}_B(C_B, C_B^\star))$ where

$$\mathcal{X}_B(C_B, C_B^\star) = \left\{ x \in \{0,1\}^{|C_B|^\star} : \sum_{j \in C_B^\star} a_j x_j \leq b - \sum_{j \in C_B \setminus C_B^\star} a_j \right\}.$$

If C_B is a minimal cover for $\mathcal{X}$, then $C_B^\star$ is also a minimal cover for $\mathcal{X}_B(C_B, C_B^\star)$, that is, the inequality $\sum_{j \in C_B^\star} x_j \leq |C_B^\star| - 1$ defines a facet of $\text{conv}(\mathcal{X}_B(C_B, C_B^\star))$. Let $(j_1, ..., j_{|N \setminus C_B^\star|})$ be the given variable sequence in $N \setminus C_B^\star$. Additionally, let $C_i^d = \{j_l \in C_B \setminus C_B^\star : l \leq i\}$ and $C_i = C_B^\star \cup C_i^d \cup \{j_l \in N \setminus C_B : l \leq i\}$ for each $i = 1, ..., |N \setminus C_B^\star|$ where $C_0 = C_B^\star$ and $C_0^d = \emptyset$. The lifting coefficients can be sequentially determined as follows.

$$\gamma_{j_i}^{\star} = \begin{cases} |C_B^{\star}| - 1 + \sum\limits_{j \in C_i^d} \gamma_j^{\star} - \max\limits_{x \in \mathcal{X}_B(C_B, C_i)} \left\{ \sum\limits_{j \in C_{i-1}} \gamma_j^{\star} x_j : x_{j_i} = 1 \right\} & , j_i \in N \setminus C_B \\ \max\limits_{x \in \mathcal{X}_B(C_B, C_i)} \left\{ \sum\limits_{j \in C_{i-1}} \gamma_j^{\star} x_j : x_{j_i} = 0 \right\} - \left(|C_B^{\star}| - 1 + \sum\limits_{j \in C_{i-1}^d} \gamma_j^{\star} \right) & , j_i \in C_B \setminus C_B^{\star} \end{cases},$$

for each $i = 1, ..., |N \setminus C_B^{\star}|$. The resulting inequality defines a facet for $\text{conv}(\mathcal{X}_B)$ if C_B is a minimal cover for $\mathcal{X}_B$.

Gu et al. (1998) remarked that the sequential lifting technique for general lifted cover inequalities can be performed in $O(n^3|C_B|)$ computational complexity by adapting the algorithm proposed by Zemel (1989). We note that there is no dominance relationship between general lifted cover inequalities and lifted cover inequalities. However, it has been demonstrated that general lifted cover inequalities are more effective in enhancing the relaxation of binary integer linear programs rather than lifted cover inequalities (Gu et al., 1998; Kaparis & Letchford, 2010).

1.2.4 General-purpose cuts

For binary integer linear programs, general-purpose cuts are typically derived from the single constraint relaxation, where the original problem is relaxed as a single-constrained optimization problem. Such a single-constraint relaxation can be defined with each constraint of the original problem or an implied linear inequality obtained by aggregating the constraints with non-negative multipliers. Then, the relaxation can be described as

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t} \quad & u^T A x \leq u^T \mathbf{b} \end{aligned}$$

$$x \in \{0,1\}^n,$$

where $u \in \mathbb{R}^r_+$. Let $\mathcal{X}_S$ be the feasible solution set of the above problem. We also refer to $\mathcal{X}_S$ as the single-constraint relaxation for the feasible solution set of the binary integer linear program, $\mathcal{X}$.

Because $\mathrm{conv}(\mathcal{X}_S)$ can be represented as a binary knapsack polytope, the valid inequalities presented in Section 1.2.3 can be used as general-purpose cuts, which we call *knapsack cuts*. However, other general-purpose cuts are used together with knapsack cuts to solve binary integer linear programs in practice. In this section, we introduce such general-purpose cuts.

Chvátal-Gomory cuts

Let us consider the following inequality derived from the constraint that defines $\mathcal{X}_S$.

$$\lfloor u^T A \rfloor x \leq u^T b$$

Because the coefficients of the decision variables in this inequality are less than those in the constraint defining $\mathcal{X}_S$, it is clear that this inequality is valid for $\mathcal{X}_S$. Because decision variables are integers, the following inequality is also valid for $\mathcal{X}_S$.

$$\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor \tag{1.3}$$

Inequalities of the form (1.3) can be obtained with any $u \in \mathbb{R}^r_+$ through the above procedure. Such inequalities are known as *Chvátal-Gomory (CG)* cuts (Chvátal, 1973; Gomory, 1958) while the above procedure is called a CG procedure.

Because CG procedures utilize not the binarity of the decision variables, but integrality, CG cuts are valid for not only $\mathcal{X}_S$ but also its relaxation $\mathcal{X}_{SR}$ where the decision variables can have general integers, which is described as follows.

$$\mathcal{X}_{SR} = \{x \in \mathbb{Z}_+^n : u^T Ax \leq u^T \mathbf{b}\}$$

Originally, CG cuts were defined for integer linear programs (Chvátal, 1973; Gomory, 1958). However, in this thesis, we describe results on CG cuts in the perspective of binary integer linear programs for consistency.

Let $\mathcal{P}$ be the linear relaxation of the binary integer linear program, described as

$$\mathcal{P} = \{x \in [0,1]^n : Ax \leq \mathbf{b}\}.$$

Let $\mathcal{P}^1$ be the relaxation improved by all the CG cuts of the form (1.3). $\mathcal{P}^1$ is referred to as the *Chvátal closure* for $\mathcal{P}$. Because infinitely many CG cuts can be generated through CG procedures with different multipliers, the Chvátal closure may seem like it is not a polytope. However, Chvátal (1973) showed that it is a rational polytope if $\mathcal{P}$ is a rational polytope. For general integer linear programs, Schrijver (1980) showed that the Chvátal closure is a rational polyhedron if the linear relaxation is a rational polyhedron. These results imply that a finite number of CG cuts is necessary to describe $\mathcal{P}^1$, and hence, some of the CG cuts are dominated and do not need to be considered in enhancing the formulation.

Let us redefine the Chvátal closure as

$$\mathcal{P}^1 = \{x \in [0,1]^n : \bar{A}x \leq \bar{\mathbf{b}}\},$$

where the number of constraints in the system $\bar{A}x \leq \bar{\mathbf{b}}$ is finite. Then, we can define CG cuts derived from $\mathcal{P}^1$ and the Chvátal closure of $\mathcal{P}^1$, denoted as $\mathcal{P}^2$. We call $\mathcal{P}^2$ the second Chvátal closure for $\mathcal{P}$. By definition, $\mathcal{P}^2$ is tighter than $\mathcal{P}^1$. In a similar way, the t-th Chvátal closure, $\mathcal{P}^t$, can be defined from $\mathcal{P}^{t-1}$ with CG cuts derived from $\mathcal{P}^{t-1}$. The CG cuts valid for $\mathcal{P}^t$ but not $\mathcal{P}^{t-1}$ are called *rank-t* CG cuts. Chvátal (1973) showed that there exists $t \in \mathbb{Z}_+$ such that $\mathcal{P}^t = \text{conv}(\mathcal{X})$. The smallest such t is called the Chvátal rank of $\mathcal{P}$. This result implies that every valid inequality for $\mathcal{X}$ can be categorized into CG cuts with some rank. Hence, the notion of rank provides a way to compare the strength between families of cuts beyond comparing each cut individually.

Although CG cuts can describe $\text{conv}(\mathcal{X})$, their use in practice is restricted due to the absence of efficient generation methods. Even for rank-1 CG cuts, their separation problem is proven to be NP-hard in a strong sense for general integer linear programs (Eisenbrand, 1999), while the computational complexity for binary integer linear programs is unknown. Fischetti & Lodi (2007) presented a mixed-integer program for the separation problem for rank-1 CG cuts, which can be solved using general-purpose optimization solvers. Although the authors could demonstrate the effectiveness of rank-1 CG cuts using the proposed model, it is still computationally challenging to solve the separation problem.

One possible way to utilize CG cuts is generating them through restricted separation problems, such as restricting the choice of multipliers for the single-constraint relaxation. For example, one can use CG cuts derived from the optimal simplex tableau obtained through solving the linear programming (LP) relaxation of the binary integer linear program. The binary integer linear program can be reformulated

into the standard form by introducing slack variables $s \in \mathbb{Z}_+^r$ as follows.

$$\max\{c^T x : Ax + s = \mathbf{b},\ x \in \{0,1\}^n,\ s \in \mathbb{Z}_+^r\} \tag{1.4}$$

Let $(x^\star, s^\star)$ be the optimal solution to the LP relaxation of the above problem, obtained by the simplex method. If $x^\star$ is not integral, there exists a row in the optimal simplex tableau, which corresponds to x_i such that $0 < x_i^\star < 1$ for some $i \in N$. Such a row can be written as

$$v^T Ax + v^T s = v^T \mathbf{b}, \tag{1.5}$$

for some $v \in \mathbb{R}^r$. We note that $v^T \mathbf{b} = x_i^\star$. Then, the following inequality can be obtained from the integrality of the decision variables.

$$\lfloor v^T A \rfloor x + \lfloor v \rfloor^T s \leq \lfloor v^T b \rfloor \tag{1.6}$$

With (1.5), the inequality (1.6) can be rewritten as

$$(v^T A - \lfloor v^T A \rfloor)x + (v - \lfloor v \rfloor)^T s \geq v^T b - \lfloor v^T b \rfloor.$$

The above inequality is called a *Gomory fractional cut*. The Gomory fractional cut separates $(x^\star, s^\star)$ from the LP relaxation, thereby improving the relaxation. The Gomory fractional cut may seem different from CG cuts due to the slack variables. However, since $s = b - Ax$, the Gomory fractional cut can be rewritten as

$$\lfloor \hat{u}^T A \rfloor x \leq \lfloor \hat{u}^T b \rfloor,$$

where $\hat{u} = v - \lfloor v \rfloor$, and hence, $\hat{u} \in \mathbb{R}^r_+$. The above inequality is a CG cut of the form (1.3) defined with $\hat{u}$. Therefore, Gomory fractional cuts are a special case of CG cuts. It is noteworthy that, after adding the above Gomory fractional cut to the formulation of the problem (1.4), another Gomory fractional cut can be generated from the enhanced formulation using the same procedure. This cutting plane algorithm, known as Gomory's cutting plane algorithm, provides the integral optimal solution to the binary integer linear program in finite iterations (Gomory, 1958). It is worth noting that Gomory's cutting plane algorithm was the first general-purpose solution approach for integer linear programs, developed before the branch-and-bound algorithm.

On the other hand, for some specific problems, the separation problem for rank-1 CG cuts can be solved efficiently. For example, blossom inequalities for the matching polytope (Edmonds, 1965) and odd-hole inequalities for the stable set polytope (Padberg, 1973) can be interpreted as CG cuts defined with multipliers that have 0 or $\frac{1}{2}$, the so-called $\{0, \frac{1}{2}\}$-cuts (Caprara & Fischetti, 1996). The separation problems for these inequalities can be solved in polynomial time.

The separation problem associated with the binary knapsack polytope and the variants has also been studied in the literature, and our study is closely related to them. Therefore, we will provide detailed reviews of them in the following chapter.

Gomory mixed-integer cut

Gomory (1960) derived another valid inequality from a row in the optimal simplex tableau (1.5) using the integrality of decision variables x. Here, we regard the slack variables s as non-negative continuous variables and $A_j \in \mathbb{Z}^r$ be each column of A

for $j \in N$. Then, the inequality is described as

$$\sum_{j \in N} \min \left\{ f'_j, \frac{f'_0(1-f'_j)}{1-f'_0} \right\} x_j + \sum_{k \in R} \max \left\{ v_k, -\frac{f'_0 v_k}{1-f'_0} \right\} s_k \geq f'_0, \tag{1.7}$$

where $f'_0 = v^T \mathbf{b} - \lfloor v^T \mathbf{b} \rfloor$ and $f'_j = v^T A_j - \lfloor v^T A_j \rfloor$ for each $j \in N$. The above inequality is called a *Gomory mixed-integer cut*. As the name suggests, the cuts were originally defined for mixed-integer programs. However, we present the results in the context of binary integer linear programs for consistency.

The Gomory mixed-integer cut can also be derived from the single-constraint relaxation $\mathcal{X}_S$. The constraint in the relaxation can be rewritten as an equality by introducing a slack variable $s_0 \in \mathbb{R}_+$ as

$$u^T A x + s_0 = u^T \mathbf{b}.$$

The Gomory mixed-integer cut (1.7) derived from this equality is expressed as

$$\sum_{j \in N} \min \left\{ f_j, \frac{f_0(1-f_j)}{1-f_0} \right\} x_j + s_0 \geq f_0,$$

where $f_0 = u^T \mathbf{b} - \lfloor u^T \mathbf{b} \rfloor$ and $f_j = u^T A_j - \lfloor u^T A_j \rfloor$ for each $j \in N$. Since $s_0 = u^T \mathbf{b} - u^T A x$, the cut can be rewritten as follows.

$$\sum_{j \in N} \left(\lfloor u^T A_j \rfloor + \max \left\{ 0, \frac{f_j - f_0}{1-f_0} \right\} \right) x_j \leq \lfloor u^T \mathbf{b} \rfloor \tag{1.8}$$

Burdet & Johnson (1975) showed that this inequality is valid for $\mathcal{X}_S$ and dominates the CG cut derived from the same single-constraint relaxation. From here, the Go-

mory mixed-integer cut refers to the inequality of the form (1.8), unless otherwise stated. It is worth noting that the Gomory mixed-integer cut is also valid for $\mathcal{X}_S$ and its relaxation $\mathcal{X}_{SR}$, as its derivation only requires the integrality of decision variables.

A closure associated with Gomory mixed-integer cuts can be defined similarly to the first Chvátal closure, known as the Gomory mixed-integer closure. As the Gomory mixed-integer cut dominates the CG cut derived from the same single-constraint relaxation (Burdet & Johnson, 1975), the Gomory mixed-integer closure is a tighter relaxation for $\mathcal{X}$ than the Chvátal closure. Moreover, the Gomory mixed-integer closure is equivalent to the split closure and mixed-integer rounding closure defined by split cuts (Cook et al., 1990) and mixed-integer rounding cuts (Nemhauser & Wolsey, 1990), respectively, which are other general-purpose cuts proposed more than 30 years after the introduction of Gomory mixed-integer cuts.

The separation problem for Gomory mixed-integer cuts is proven to be NP-hard in a strong sense for both general integer linear programs (Caprara & Letchford, 2003) and binary integer linear programs Lee (2019). Therefore, Gomory mixed-integer cuts are derived from the optimal simplex tableau in practice.

According to Bixby, Fenelon, et al. (2004), Gomory mixed-integer cuts were the most effective general-purpose cuts, and now they have become a crucial component of modern optimization software. However, the usefulness of Gomory mixed-integer cuts was not recognized from the beginning. The cuts were considered mathematically elegant but impractical right after their invention by Gomory (1960). It took about 30 years for them to be revived by the successful incorporation into the branch-and-cut framework (Balas, Ceria, et al., 1996). For a detailed history of Gomory

mixed-integer cuts, we refer the reader to Cornuéjols et al. (2007).

Cut-generating function

The success of Gomory mixed-integer cuts in practice has prompted studies on other families of general-purpose cuts that can be derived from the single-constraint relaxation. Conforti, Cornuéjols, et al. (2015) introduced the term *cut-generating function*, which describes the studies as a whole.

Let us consider a set $\mathcal{X}^p_{SE}(\mathbf{a}, f, K)$ for some $p \in \mathbb{Z}_+$, $\mathbf{a} = (a_1, ..., a_p)$, $0 < f < 1$, and $K \in \mathbb{Z}$ where $a_j \in \mathbb{R}$ for each $j \in \{1, \ldots, p\}$, which is defined as

$$\mathcal{X}^p_{SE}(\mathbf{a}, f, K) = \{(x, s_0) \in \mathbb{Z}^p_+ \times \mathbb{R}_+ : \sum_{j=1}^{p} a_j x_j + s_0 = f + K\},$$

Definition 1.6. $\pi : \mathbb{R} \to \mathbb{R}$ *is a cut-generating function if the inequality*

$$\sum_{j=1}^{p} \pi(a_j) x_j + s_0 \geq f$$

is valid for $\mathcal{X}^p_{SE}(\mathbf{a}, f, K)$ *for all* $p \in \mathbb{Z}_+$, $\mathbf{a} \in \mathbb{R}^n$, $0 < f < 1$, *and* $K \in \mathbb{Z}$.

$\mathcal{X}_{SR}$ can be reformulated as $\mathcal{X}^n_{SE}(u^T A, f_0, \lfloor u^T \mathbf{b} \rfloor)$ with a slack variable $s_0 \in \mathbb{R}_+$, where $f_0 = u^T \mathbf{b} - \lfloor u^T \mathbf{b} \rfloor$. Therefore, a cut-generating function π can derive a valid inequality for $\mathcal{X}_{SR}$, which is also valid for $\mathcal{X}_S$.

A valid inequality for $\mathcal{X}^n_{SE}(u^T A, f_0, \lfloor u^T \mathbf{b} \rfloor)$, obtained using π, is described as follows.

$$\sum_{j \in N} \pi(u^T A_j) x_j + s_0 \geq f_0$$

Since $s_0 = u^T\mathbf{b} - u^T Ax$, this inequality can be rewritten as

$$\sum_{j \in N} (u^T A_j - \pi(u^T A_j)) x_j \leq \lfloor u^T \mathbf{b} \rfloor,$$

which is a valid inequality for $\mathcal{X}_{SR}$. Then, this inequality is also valid for $\mathcal{X}_S$, which we call the *CGF cut* defined by π.

Depending on the choice of cut-generating function, CGF cuts can represent various general-purpose cuts. For example, let us consider the function π_{CG} and π^f_G defined as

$$\pi_{CG}(z) = z - \lfloor z \rfloor$$

and

$$\pi^f_G(z) = \min \left\{ \pi_{CG}(z), \frac{f(1 - \pi_{CG}(z))}{1 - f} \right\},$$

for some $0 < f < 1$. Then, the CG and Gomory mixed-integer cuts for $\mathcal{X}_S$ are equivalent to CGF cuts defined by π_{CG} and $\pi^{f_0}_G$, respectively.

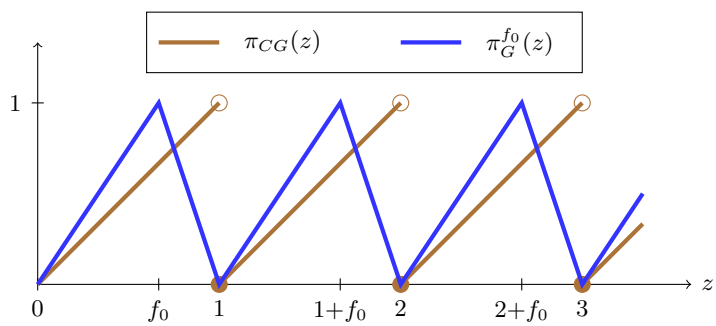


Figure 1.2: Example of $\pi_{CG}(z)$ and $\pi^{f_0}_G(z)$

As shown in the relationship between CG and Gomory mixed-integer cuts, the strength of CGF cuts depends on cut-generating functions, and some CGF cuts can

be dominated by the other CGF cuts. Therefore, studies on cut-generating functions have focused on those generating non-dominated CGF cuts.

A cut-generating function π is said to be *minimal* if there is no cut-generating function $\pi' \neq \pi$ such that $\pi'(z) \leq \pi(z)$ for all $z \in \mathbb{R}$. Hence, minimal cut-generating functions yield non-dominated CGF cuts. The necessary and sufficient conditions for minimality were derived from studies on the infinite relaxation, introduced by Gomory & Johnson (1972a) and Gomory & Johnson (1972b). The derivation of each condition is technical and theoretical. Hence we present them without further explanations.

Theorem 1.2 (Gomory & Johnson, 1972)**.** *π is a minimal cut-generating function if and only if $\pi(0) = 0$, π is subadditive, periodic, and satisfies the symmetry condition defined as follows.*

1. *Subadditivity:* $\pi(z_1) + \pi(z_2) \geq \pi(z_1 + z_2),\ \forall z_1, z_2 \in \mathbb{R}$
2. *Periodicity:* $\pi(z) = \pi(z + k),\ \forall z \in \mathbb{R},\ \forall k \in \mathbb{Z}$
3. *Symmetry:* $\pi(z) + \pi(f - z) = f,\ \forall 0 < f < 1$

By the symmetry condition, if π is a minimal cut-generating function, then $\pi(f) = f$ and $\pi(0) = 0$ for all $0 < f < 1$. Furthermore, the combination of the symmetry condition and periodicity implies that $\pi(z) \leq f$ for all $z \in \mathbb{R}$ if π is a minimal cut-generating function.

π_{CG} is not minimal because the function does not satisfy the symmetry condition, while π_G^f is a minimal cut-generating function. This result also explains why CG cuts are dominated by Gomory mixed-integer cuts derived from the same single-constraint relaxation.

Various cut-generating functions have been proposed in the literature. However, for the sake of brevity, we omit the introduction of other minimal cut-generating functions as they are not directly addressed in this thesis. Instead, we recommend readers to refer to comprehensive surveys on cut-generating functions by Basu et al. (2016a) and Basu et al. (2016b).

1.2.5 Chance-constrained programming approach for optimization problems under uncertainty

Real-world optimization problems often involve uncertain input data. One common approach is to handle this uncertainty deterministically by estimating the input data. However, even small changes to the input can significantly affect the quality of the optimal solution, and in some cases, the solution may become infeasible. Therefore, it is crucial to address uncertainty in optimization problems carefully. This section briefly introduces conventional modeling frameworks for optimization problems under uncertainty and explains the chance-constrained programming approach considered in this thesis.

The conventional modeling frameworks for dealing with uncertainty can be categorized based on the information utilized about the uncertainty. A robust optimization approach (Ben-Tal et al., 2009; Gabrel et al., 2014; Bertsimas, Brown, et al., 2011) requires only the uncertainty set of input data from which realizations are drawn. This modeling framework aims to find the best solution in the worst-case scenario when the input data changes within the given uncertainty set. Hence, the feasibility of the solution is immune to the realization of uncertain input data. Let ξ be the realization of uncertain inputs where $\xi \in \Xi$, and $\mathcal{X}(\xi)$ be the feasible

solution set of the considered optimization problem when ξ is realized. Then, the optimization problem can be modeled using the robust optimization framework as

$$\min_{\xi \in \Xi} \max_{x \in \mathcal{X}(\xi)} c^T x,$$

which is a special case of bilevel optimization problems. Alternatively, the above problem can be expressed as a single-level optimization problem as

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t} \quad & x \in \mathcal{X}(\xi),\ \xi \in \Xi, \end{aligned}$$

which is called the robust counterpart. The robust counterpart may have infinitely many constraints when $|\Xi| = \infty$, which makes it difficult to solve. However, some robust counterparts can be reformulated into tractable optimization problems by exploiting the properties of Ξ. Examples of such reformulations can be found in Bertsimas & Sim (2004) and Ben-Tal et al. (2009).

On the other hand, the stochastic programming approach, initiated by Dantzig (1955), utilizes the uncertainty set and the probabilistic description of input data. This modeling framework typically aims to optimize the expected value of the objective function over the realizations of uncertain input data. Let $\mathcal{D}$ be the probability distribution of ξ, and $x(\xi)$ be the feasible solution when ξ is realized. Then, the optimization problem can be modeled using the stochastic programming approach as follows.

$$\max_{x(\xi) \in \mathcal{X}(\xi)} \mathbb{E}_{\xi \sim \mathcal{D}} \{c^T x(\xi)\}.$$

For numerical computations, the above problem is often approximated using the sample approximation approach, which considers a finite number of samples for the realization of ξ. The quality of the approximation is significantly affected by the number of samples. However, an enormous number of samples is required for the tight approximation as the number of uncertain inputs increases. Hence, scalable optimization techniques for the sample approximation approach have been extensively studied. We refer the interested reader to several textbooks (Birge & Louveaux, 2011; Kall et al., 1994; Shapiro et al., 2021) and a survey by Powell (2019).

The chance-constrained programming approach is a type of stochastic programming approach introduced in Charnes et al. (1958), Miller & Wagner (1965), and Prékopa (1970). This approach aims to find a solution that guarantees feasibility with a certain probability for the realization of uncertain input data. By utilizing the chance-constrained programming approach, the optimization problem can be formulated as follows:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t} \quad & \mathbb{P}\{x \in \mathcal{X}(\xi)\} \geq \rho, \end{aligned}$$

where $\mathbb{P}\cdot$ represents the probability and ρ is a desired probability. The constraint related to the probability is called the *chance constraint*. It should be noted that the above model is equivalent to the robust counterpart when $\rho = 1$. Hence, the chance-constrained programming approach can be viewed as a generalization of the robust optimization approach.

Despite its theoretical advantages, the chance-constrained programming approach

presents several challenges in practical applications. Firstly, the approach requires the joint probability distribution function of random variables across the constraints, which can be impractical to estimate. Secondly, even if the distribution function is well-estimated, the numerical processing of chance constraints may take more work. For example, if the cumulative distribution function is not provided as a closed form, the chance-constrained program may be difficult to reformulate deterministically, leading to computational intractability. In this case, even checking the feasibility of a given solution may not be possible, and the only viable option may be to check it approximately using the Monte-Carlo simulation. Additionally, even with well-estimated distribution functions, chance-constrained programs are often formulated as non-convex optimization problems, posing significant challenges.

One possible way to address those issues is to approximate the chance-constrained program by assuming the constraint-wise independence between random variables (Ahmed, 2014). Let us consider an optimization problem under uncertainty, which can be formulated as a binary integer linear program in the deterministic case. Then, the chance-constrained program is formulated as

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \mathbb{P}\left\{ \sum_{j \in N} \tilde{a}_{kj} x_j \leq b_k,\ k \in R \right\} \geq \rho \\ & x \in \{0,1\}^n, \end{aligned}$$

where $\tilde{a}_{kj}$'s are random variables. Then, the approximation can be formulated as

follows.

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \mathbb{P}\left\{ \sum_{j \in N} \tilde{a}_{kj} x_j \leq b_k \right\} \geq \rho, \ k \in R \\ & x \in \{0,1\}^n, \end{aligned}$$

Estimating the distribution functions associated with each constraint might be easier than estimating those for all uncertain input data. In addition, if each chance constraint can be represented as a convex constraint, the above problem becomes computationally tractable. Convex approximation techniques (Nemirovski & Shapiro, 2007; Ahmed, 2014) can also be used if necessary.

In the above approximation, a single-constraint relaxation defined by each chance constraint can be represented as the feasible solution set of the chance-constrained binary knapsack problem, which is described as

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \mathbb{P}\left\{ \sum_{j \in N} \tilde{a}_j x_j \leq b \right\} \geq \rho \\ & x \in \{0,1\}^n, \end{aligned}$$

where $\tilde{a}_j$'s are random variables. We note that the knapsack cuts introduced in Section 1.2.3 are not applicable to the chance-constrained binary knapsack problem because the chance constraint is often represented as a nonlinear inequality.

Therefore, it is necessary to study the chance-constrained binary knapsack problem to derive cuts and their generation methods that can be used in solving the approximations of chance-constrained programs, as we do in Chapter 3.

Another approach to utilize chance-constrained programs in practice is sampling-based approximations (Luedtke et al., 2010; Ruszczyński, 2002; Küçükyavuz, 2012), similar to the stochastic programming approach. While this approach falls outside the scope of our study, readers interested in approximations of chance-constrained programs, including the methods mentioned above, can refer to several textbooks (Prékopa, 2013; Shapiro et al., 2021) and a survey by Nemirovski (2012).

1.3 Research objectives and contributions

The main objective of this thesis is to enhance the capability of solving binary integer programs using efficient cut-generation methods for two variants of the binary knapsack problem: the binary knapsack problem with generalized upper bounds (GKP) and the chance-constrained binary knapsack problem (CKP).

Firstly, we investigate the GKP, which can derive stronger cuts for binary integer linear programs than the binary knapsack problem. Specifically, we consider rank-1 Chvátal-Gomory (CG) cuts and their separation problem for the GKP, which have not been studied in the literature. By exploiting the properties of non-dominated rank-1 CG cuts for the GKP, we present exact and heuristic algorithms for the separation problem, along with computational complexity analysis. Through extensive computational experiments, we compare the efficiency of the proposed separation algorithms with the mixed-integer programming approach and demonstrate the effectiveness of CG cuts compared to existing cuts for the GKP.

Secondly, we present a novel method for strengthening rank-1 CG cuts for binary integer linear programs to improve the formulation-enhancing effect of the CG cuts. We first reveal the connection between rank-1 CG cuts for binary integer linear programs and lifted cover inequalities for binary knapsack problems. Based on this result, our method strengthens a given rank-1 CG cut using a lifting function of cover inequalities for binary knapsack problems. We compare the strength of the obtained cut with those obtained through existing strengthening methods from the given CG cut. Subsequently, we extend this method to rank-1 CG cuts for binary integer linear programs with generalized upper bounds. We evaluate the effectiveness of the proposed strengthening methods through computational experiments.

Thirdly, we consider the CKP, which arises in the chance-constrained programming approach for optimization problems under uncertainty. We assume that the item weights are independently normally distributed. Then the CKP is formulated as a binary integer nonlinear program where the cuts for the binary knapsack problem are not applicable. For the CKP, a family of valid inequalities has been introduced by Atamtürk & Narayanan (2009), known as probabilistic cover inequalities. We propose an efficient sequential lifting heuristic for probabilistic cover inequalities based on a continuous relaxation for the CKP. We first introduce a non-convex continuous relaxation of the CKP, represented as a non-convex optimization problem, and show that the relaxation provides tighter upper bounds than the other continuous relaxations presented in the literature. In general, non-convex optimization problems are computationally hard to solve; however, we show that the non-convex relaxation can be solved in polynomial time. Our lifting heuristic utilizes the algorithm to solve the non-convex relaxations for the lifting problem of probabilistic

cover inequalities, represented as another CKP. We compare the performance of the lifting heuristic with existing methods, including Atamtürk & Narayanan (2009), through computational tests.

The contributions of the thesis are given as follows.

1. Separation of rank-1 Chvátal-Gomory cuts for the binary knapsack problem with generalized upper bounds.

    - We characterize the non-dominated rank-1 CG cuts for the GKP and, based on their properties, show that the separation problem can be decomposed into sub-problems.
    - We propose a pseudo-polynomial time exact separation algorithm for rank-1 CG cuts for the GKP. This result implies that the separation problem for the GKP is not strongly NP-hard, while it is known to be NP-hard in a strong sense for general integer linear programs. In addition, we devise an efficient heuristic separation algorithm based on the decomposition property of the separation problem.
    - The computational experiment results demonstrate that the proposed separation algorithms can efficiently generate rank-1 CG cuts compared to the mixed-integer programming approach proposed by Fischetti & Lodi (2007). Furthermore, the rank-1 CG cuts generated by the proposed algorithms outperform existing cuts for the GKP in terms of formulation enhancement and computational efficiency.

2. Chvátal-Gomory cut strengthening method for binary integer linear programs and its extension to generalized upper bounds.

- We propose a novel method for strengthening rank-1 CG cuts for binary integer linear programs, which derives a stronger cut that we call the SCG cut.
- We demonstrate that the SCG cut can have a Chvátal rank higher than 1, and it is at least as strong as the Gomory mixed-integer cut derived from the single-constraint relaxation corresponding to the given CG cut. Furthermore, we specify the condition where the SCG cut dominates CGF cuts defined by any cut-generating functions.
- We extend the method to binary integer linear programs with generalized upper bounds and generalize the strength comparison results.
- The computational experiment results indicate that the proposed strengthening method can yield more enhanced formulations using fewer cuts compared to using CG cuts or Gomory mixed-integer cuts. Furthermore, using the proposed method decreased the computation time to enhance the formulations because the number of iterations in the cutting plane algorithm is reduced.

3. Lifting heuristic for probabilistic cover inequalities for the chance-constrained binary knapsack problem.
   - We show that the non-convex continuous relaxation for the CKP provides a tighter upper bound than the other continuous relaxations in the literature.
   - Despite the non-convexity, we propose a polynomial-time exact algorithm for the non-convex continuous relaxation.

- We devise an efficient lifting heuristic of probabilistic cover inequalities based on the non-convex continuous relaxation for the lifting problem. The computational experiments show that our lifting heuristic outperforms the existing methods in terms of computational efficiency, while the strength of the resulting lifted probabilistic cover inequality remains competitive compared to the existing lifting heuristics.

1.4 Organization of the thesis

The remainder of this thesis is organized as follows.

- In Chapter 2, we focus on the separation problem of rank-1 Chvátal-Gomory cuts for the binary knapsack problem with generalized upper bounds. We begin by characterizing the non-dominated rank-1 CG cuts for the problem and then propose a decomposition of the separation problem based on the result. We develop an exact separation algorithm using a dynamic programming approach and discuss the algorithm's computational complexity. Additionally, we propose an efficient heuristic based on the decomposition property of the separation problem. We present computational test results demonstrating the proposed algorithms' efficiency and the effectiveness of rank-1 CG cuts.

- In Chapter 3, we propose a strengthening method for rank-1 Chvátal-Gomory cuts for binary integer linear programs. We compare the strength of the resulting cut with those obtained through existing strengthening methods from the given Chvátal-Gomory cut. We also extend the results to binary integer linear programs with generalized upper bounds. Finally, we present computational

experiment results that demonstrate the effectiveness of the proposed method.

- In Chapter 4, we discuss three formulations for the chance-constrained binary knapsack problem found in the literature and compare the continuous relaxations obtained from the formulations. We propose a polynomial-time algorithm for the non-convex continuous relaxation which yields a tighter upper bound compared to the other continuous relaxations. We then present a lifting heuristic using the non-convex relaxation and the proposed algorithm, along with corresponding experimental results.

- In Chapter 5, we summarize the results of the thesis and discuss possible future study directions.

Chapter 2

Separation of the rank-1 Chvátal-Gomory cuts for the knapsack problem with generalized upper bounds

In this chapter, we investigate rank-1 CG cuts for the binary knapsack problem with generalized upper bounds, GKP for short, and propose efficient solution approaches for their separation problem. Firstly, we explore the properties of non-dominated rank-1 CG cuts for the GKP and analyze the computational complexity of the corresponding separation problem. We demonstrate that the separation problem can be solved in pseudo-polynomial time, whereas it is known to be strongly NP-hard for general integer linear programs. We also devise an efficient heuristic algorithm for the separation problem, along with with an analysis of its computational complexity. Finally, we conduct computational experiments to evaluate the performance of the proposed algorithms. The test results indicate that the proposed separation algorithms generate CG cuts efficiently, and the generated CG cuts outperform the existing valid inequalities for the GKP in terms of formulation enhancement and computational efficiency.

2.1 Introduction

The knapsack problem with generalized upper bounds, GKP for short, is defined as the binary knapsack problem with additional constraints where at most, one item should be selected among some of the given items. For a given set of items $N = \{1, \ldots, n\}$ and a finite index set $I = \{1, \ldots, m\}$, a generalized upper bound (GUB) set is defined as $K_i \subseteq N$ with $|K_i| = n_i$ for each $i \in I$ such that $N = \cup_{i \in I} K_i$ and $K_i \cap K_j = \emptyset$ whenever $i \neq j$. In addition, positive integers b, c_j and a_j for all $j \in N$, and d_i for all $i \in I$, are given. Then, the GKP can be formulated as a binary integer linear program as follows.

$$
\begin{aligned}
\text{(GKP)} \quad \max \quad & \sum_{i \in I} \sum_{j \in K_i} c_j x_j \\
\text{s.t} \quad & \sum_{i \in I} \sum_{j \in K_i} a_j x_j \leq b && (2.1) \\
& \sum_{j \in K_i} x_j \leq 1, \ \forall i \in I && (2.2) \\
& x \in \{0,1\}^n.
\end{aligned}
$$

Constraints (2.2) is called GUB constraints. We denote $\mathcal{X}_G$ as the feasible solution set of the GKP, and the GKP polytope is defined as $\text{conv}(\mathcal{X}_G)$. We note that if GUB sets are singletons where $K_i = \{i\}$ for all $i \in I$ with $m = n$, the GKP polytope is equivalent to the binary knapsack polytope, that is, the GKP is the generalization of the binary knapsack problem.

GUB constraints arise in various applications of binary integer linear programs, such as representing transmitter selection for each receiver in wireless network design

problems (D'Andreagiovanni et al., 2013), gate allocation for each flight in airport gate assignment problems (Kim et al., 2023), and choice of the beginning period for each job in machine scheduling problems (Sousa & Wolsey, 1992), among others (Balintfy et al., 1978; Sankaran et al., 1999; Cavalcante et al., 2001). For such binary integer linear programs, each constraint or an implied constraint obtained by aggregating the constraints with non-negative multipliers defines a tighter relaxation, together with GUB constraints, than the single-constraint relaxation. This relaxation can be represented as a GKP polytope by complementing the variables if necessary (Johnson & Padberg, 1981). Therefore, cuts derived from the GKP polytope can be more effective than those for the binary knapsack polytope in solving binary integer linear programs. Several studies have been conducted on valid inequalities and their generation methods for the GKP polytope (Wolsey, 1990; Nemhauser & Vance, 1994; Sherali & Lee, 1995; Gu et al., 1998).

We consider rank-1 CG cuts (Chvátal, 1973), which we refer to as CG cuts for brevity, for the GKP. As introduced in Section 1.2.4, CG cuts are general-purpose cuts that can be defined for general integer linear programs. However, despite their generality and effectiveness (Fischetti & Lodi, 2007), the use of CG cuts has been restricted in practice due to the absence of efficient generation methods.

This study mainly focuses on the CG cut separation problem for the GKP, which has not been studied in the literature before. We characterize non-dominated CG cuts for the GKP and analyze the computational complexity of the separation problem. Based on the results, we present efficient separation algorithms with corresponding computational experiment results.

The remainder of this chapter is organized as follows. We review the relevant

literature in Section 2.2. In Section 2.3, we discuss the relationship between the existing valid inequalities and the CG cuts for the GKP polytope. The properties of non-dominated CG cuts are also presented. Based on the result, we propose an exact pseudo-polynomial time algorithm for the separation problem in Section 2.4. An efficient heuristic algorithm is also devised in Section 2.5. The computational efficiency of the proposed separation algorithms, as well as the effectiveness of the generated CG cuts, are demonstrated in Section 2.6. Finally, the concluding remarks are given in Section 2.7.

2.2 Literature review

In this section, we introduce valid inequalities and their generation methods for the GKP polytope in the literature and present the studies on CG cuts associated with the GKP.

For the GKP polytope, Wolsey (1990) first introduced a family of valid inequalities, GUB cover inequalities, by generalizing the concept of the cover for the binary knapsack polytope (Wolsey, 1975; Hammer et al., 1975; Balas, 1975), which is described as follows.

$$\sum_{j \in C_G} x_j \leq |C_G| - 1, \tag{2.3}$$

where $C_G \subseteq N$ is a GUB cover such that $\sum_{j \in C_G} a_j > b$ and $|C_G \cap K_i| \leq 1$ for all $i \in I$. The author also investigated the extensions of GUB cover inequalities.

GUB cover inequalities can be strengthened through lifting techniques that improve the coefficients of variables not present in the GUB cover, resulting in lifted GUB cover inequalities. If the GUB cover is minimal, the lifted GUB cover inequality

can define a facet for the GKP polytope. Nemhauser & Vance (1994) generalized the result in the binary knapsack polytope by Balas & Zemel (1978) by characterizing the lifting coefficients required to define a facet for the GKP polytope.

In Sherali & Lee (1995), an efficient sequential lifting algorithm was proposed that simultaneously improves the coefficients of the variables in a GUB set. The authors also devised a simultaneous lifting method for GUB cover inequalities by deriving bounds on lifting coefficients to define a facet for the GKP polytope, independently of Nemhauser & Vance (1994).

While the above lifting techniques are known as up-lifting, lifted GUB cover inequalities can be generalized by modifying the coefficients of the variables in the GUB cover and the right-hand side. Such generalized inequalities, called general lifted cover inequalities, can be generated through down-lifting techniques and represent more diverse facet-defining inequalities for the GKP polytope. Gu et al. (1998) proposed an efficient down-lifting technique and extensively examined the practical aspects and implementation of the resulting general lifted GUB cover inequalities generated together with up-lifting techniques.

In contrast to the valid inequalities for the GKP polytopes mentioned above, CG cuts (Chvátal, 1973) are valid inequalities that can be defined for general integer linear programs. Although negative results have been reported on the separation problem for general integer linear programs (Eisenbrand, 1999), CG cuts for some specific problems can be generated efficiently (Edmonds, 1965; Padberg, 1973), as discussed in Section 1.2.4. Building on the review of these studies, we now narrow the scope to those associated with GKP.

Park & Lee (2011) characterized non-dominated CG cuts for the fixed-charge

binary knapsack problem. The authors proposed a pseudo-polynomial time exact separation algorithm by decomposing the separation problem using properties of the non-dominated CG cuts. The authors also showed that the separation problem for the binary knapsack problem can be solved in pseudo-polynomial time. Their results imply that the separation problem for the (fixed-charge) binary knapsack problem is at least weakly NP-hard while the problem for general integer linear programs is strongly NP-hard (Eisenbrand, 1999).

Based on the result by Park & Lee (2011), Lee (2012) devised an efficient heuristic to separate CG cuts for the binary knapsack problem. The author demonstrated the effectiveness of the CG cuts compared to the general-purpose cuts generated from commercial optimization software through extensive computational tests.

For the GKP, Glover, Sherali, et al. (1997) analyzed the separation problem for a family of CG cuts in which the right-hand side is fixed to some integers less than or equal to the number of items. Subsequently, the authors devised a polynomial-time algorithm for the separation problem. However, to the best of our knowledge, the computational complexity of the separation problem for general CG cuts for the GKP has not yet been settled.

2.3 Non-dominated CG cuts for the GKP polytope

The CG cut for the GKP defined with $(u_0, u) \in \mathbb{R}_+^{m+1}$ is described as follows.

$$\sum_{i \in I} \sum_{j \in K_i} \lfloor u_0 a_j + u_i \rfloor x_j \leq \left\lfloor u_0 b + \sum_{i \in I} u_i \right\rfloor \tag{2.4}$$

Here, u_0 is the multiplier associated with the constraint (2.1) and u_i is associated with each of GUB constraints (2.2) for each $i \in I$. We note that the variable bound constraints, $x_j \leq 1$ for each $j \in N$, can be considered negligible when defining the CG cut because they are redundant with the GUB constraints.

We first show that the CG cuts include some of existing valid inequalities in the literature.

Proposition 2.1. *A GUB cover inequality is a CG cut for the GKP.*

Proof. Let C_G be the GUB cover such that $\sum_{j \in C_G} a_j > b$ and $a(C_G) = \sum_{j \in C_G} a_j$. Because the GKP implicitly has additional constraints, $x_j \leq 1$ for each $j \in N$. Hence, the CG cut can be rewritten as

$$\sum_{i \in I} \sum_{j \in K_i} \lfloor u_0 a_j + u_i + v_j \rfloor x_j \leq \left\lfloor u_0 b + \sum_{i \in I} u_i + \sum_{j \in N} v_j \right\rfloor,$$

where v_j is the multiplier associated with $x_j \leq 1$ for each $j \in N$. Let $u_0 = 1/a(C_G)$, $u_i = 0$ for all $i \in I$, and $v_j = (a(C_G) - a_j)/a(C_G)$ for all $j \in C_G$ while $v_j = 0$ for all $j \in N \setminus C_G$. Then, the CG cut is defined as

$$\sum_{j \in C_G} x_j \leq \left\lfloor |C_G| - 1 + \frac{b}{a(C_G)} \right\rfloor,$$

where the right-hand is equivalent to $|C_G| - 1$ because $b/a(C_G) < 1$. Therefore, the result follows. □

We note that (general) lifted GUB cover inequalities may not be a rank-1 CG cut, that is, they can be a CG cut with a higher rank. Nevertheless, the rank-1 CG cuts may still derive useful valid inequalities for the GKP polytope.

Let $\mathcal{P}_G$ be the linear relaxation of $\mathcal{X}_G$ provided by the formulation defined in Section 2.1 and $\mathcal{P}_G^1$ be the Chvátal closure of P described with the CG cuts of the form (2.4). Recall that the Chvátal closure of a rational polytope is also a rational polytope (Chvátal, 1973). Therefore, $\mathcal{P}_G^1$ is a polytope, and there exist dominated CG cuts of the form (2.4), which are ineffective in improving the relaxation $\mathcal{P}_G$. In the subsequent discussion, we characterize the properties of multipliers $(u_0, u) \in \mathbb{R}_+^{m+1}$ that may yield non-dominated CG cuts.

Proposition 2.2. *$\mathcal{P}_G^1$ is the set of all points in $\mathcal{P}_G$ satisfying the CG cuts* (2.4) *for all $(u_0, u) \in \mathbb{R}_+^{m+1}$ such that $u_0 < 1$ and $u_i < 1$ for all $i \in I$.*

Proof. For a given CG cut $\beta^T x \leq \beta_0$ defined with $(\hat{u}_0, \hat{u}) \in \mathbb{R}_+^{m+1}$, let $(u_0^1, u^1) = (\hat{u}_0 - \lfloor \hat{u}_0 \rfloor, \hat{u} - \lfloor \hat{u} \rfloor)$ and $(u_0^2, u^2) = (\lfloor \hat{u}_0 \rfloor, \lfloor \hat{u} \rfloor)$. It is clear that $\beta^T x \leq \beta_0$ is the sum of the two CG cuts of the form (2.4) defined with (u_0^1, u^1) and (u_0^2, u^2), respectively. Let $\gamma^T x \leq \gamma_0$ be the CG cut defined with (u_0^1, u^1). By definition, $u_0^1 < 1$ and $u_i^1 < 1$ for all $i \in I$. In addition, since the cut defined with (u_0^2, u^2) is valid for $\mathcal{P}_G$, $\{x \in \mathcal{P}_G : \gamma^T x \leq \gamma_0\} \subseteq \{x \in \mathcal{P}_G : \beta^T x \leq \beta_0\}$. Therefore, the result follows. □

The above proposition implies that the multipliers (u_0, u) with some of their components being greater than or equal to 1 do not need to be considered in the description of $\mathcal{P}_G^1$. Let $U = \{(u_0, u) \in \mathbb{R}_+^{m+1} : u_0 < 1,\ u_i < 1,\ \forall i \in I\}$. The following

proposition further characterize the properties of multipliers $(u_0, u) \in U$ that may yield non-dominated CG cuts.

Proposition 2.3. *Suppose that $\beta^T x \leq \beta_0$ is a CG cut defined with $(\hat{u}_0, \hat{u}) \in U$ such that $\hat{u}_0 > 0$. If $\beta^T x \leq \beta_0$ is non-dominated, then there exists $(u_0^\star, u^\star) \in U$, which yields $\beta^T x \leq \beta_0$, such that $u_0^\star = t/(a_l - a_k)$ for some $\{k, l\} \subseteq K_i \cup \{0\}$ and $i \in I$ such that $a_l > a_k$, where $a_0 = 0$ and t is a positive integer less than $a_l - a_k$.*

Proof. Let us consider the following optimization problem which is the dual problem of the linear program, $\max\{\beta^T x : x \in \mathcal{P}_G\}$.

$$\begin{aligned} \min \quad & u_0 b + \sum_{i \in I} u_i \\ \text{s.t} \quad & \beta_j \leq u_0 a_j + u_i, \ \forall j \in K_i, \ i \in I \\ & u_0 \geq 0, u_i \geq 0, \ i \in I \end{aligned}$$

Here, we denote (u_0, u) as the decision variables of the above problem. By definition of the given CG cut, $\beta_j = \lfloor \hat{u}_0 a_j + \hat{u}_i \rfloor$ for each $j \in K_i$ and $i \in I$. Hence, we can see that $(\hat{u}_0, \hat{u})$ is a feasible solution for the above problem.

On the other hand, let $(u_0^\star, u^\star)$ be the optimal solution of the above problem. Then, a CG cut, $\gamma^T x \leq \gamma_0$, can be defined with the optimal solution where $\gamma_0 = \lfloor u_0^\star b + \sum_{i \in I} u_i^\star \rfloor$ and $\gamma_j = \lfloor u_0^\star a_j + u_i^\star \rfloor$ for each $j \in K_i$ and $i \in I$. Due to the constraints of the above problem,

$$\beta_j \leq u_0^\star a_j + u_i^\star, \ j \in K_i, \ i \in I,$$

hence, $\beta_j \leq \gamma_j$ for each $j \in K_i$ and $i \in I$. In addition, because $(\hat{u}_0, \hat{u})$ is the feasible

solution of the above problem,

$$u_0^\star b + \sum_{i \in I} u_i^\star \leq \hat{u}_0 b + \sum_{i \in I} \hat{u}_i,$$

which implies that $\gamma_0 \leq \beta_0$. Suppose that there exists $j \in K_i$ for some $i \in I$ such that $\beta_j < \gamma_j$ or $\gamma_0 < \beta_0$. Then, $\beta^T x \leq \beta_0$ is clearly dominated by $\gamma^T x \leq \gamma_0$, which contradicts to the assumption that $\beta^T x \leq \beta_0$ is non-dominated. Therefore, $\gamma_j = \beta_j$ for each $j \in N$ and $\gamma_0 = \beta_0$. In other words, $(u_0^\star, u^\star)$ also yields CG cut $\beta^T x \leq \beta_0$. $(u_0^\star, u^\star) \in U$ is induced by Proposition 2.2. If $(u_0^\star, u^\star) \notin U$, $\beta^T x \leq \beta_0$ is a dominated CG cut, which contradicts the assumption that it is non-dominated.

Now, we characterize the form of $u_0^\star$. By Glover & Klingman (1979), it has been shown that there exists an optimal solution $(u_0^\star, u^\star)$ to the above program such that $u_0^\star = \beta_j / a_j$ for some $j \in N$ or $u_0^\star = (\beta_l - \beta_k)/(a_l - a_k)$ for some $\{k, l\} \subseteq K_i$ and $i \in I$. Therefore, the result follows. □

The converse of Proposition 2.3 may not be true. However, the proposition implies that it is sufficient to consider a finite number of possible values for u_0 to generate non-dominated CG cuts. Now, let us consider the following CG cut defined with $(u_0^\star, u) \in U$ such that $u_0^\star = t/(a_l - a_k)$ where $\{k, l\} \subseteq K_i \cup \{0\}$ and $i \in I$ with $a_l > a_k$, $a_0 = 0$, and t is a positive integer less than $a_l - a_k$.

$$\sum_{i \in I} \sum_{j \in K_i} \lfloor u_0^\star a_j + u_i \rfloor x_j \leq \left\lfloor u_0^\star b + \sum_{i \in I} u_i \right\rfloor \tag{2.5}$$

The following proposition characterizes the property of $u \in \mathbb{R}_+^m$ in which (2.5) may be a non-dominated CG cut.

Proposition 2.4. *If the CG cut* (2.5) *is non-dominated, then there exists* $u^\star \in \mathbb{R}^m_+$*, which yields the same inequality, such that, for each* $i \in I$*,* $u^\star_i = 0$ *or* $u^\star_i = (1 - f_j)$ *for some* $j \in K_i$ *with* $f_j > 0$*, where* $f_j = u^\star_0 a_j - \lfloor u^\star_0 a_j \rfloor$ *for all* $j \in N$*.*

Proof. Let $\beta_j \in \mathbb{Z}_+$ be the coefficient of x_j in the inequality (2.5) for each $j \in N$. Then, we have $\beta_j \leq u^\star_0 a_j + u_i$ and $\beta_j = \lfloor u^\star_0 a_j \rfloor + \lfloor f_j + u_i \rfloor$ for each $j \in N$. For simplicity, we define $f_{n+1} = 1$. For each $i \in I$, let us define $K^L_i = \{j \in K_i \cup \{n+1\} : (1 - f_j) \leq u_i\}$ and $K^U_i = \{j \in K_i : (1 - f_j) > u_i\}$. Note that K^L_i is non-empty by definition. Then, for each $i \in I$ and $j \in K_i$, if $j \in K^L_i$, then $\lfloor f_j + u_i \rfloor = 1$, and $\lfloor f_j + u_i \rfloor = 0$ otherwise.

Let us now consider $u^\star \in \mathbb{R}^m_+$ such that

$$u^\star_i = \max_{j \in K^L_i} \{1 - f_j\},\ \forall i \in I.$$

Then, for each $i \in I$, we have $u^\star_i \leq u_i$. Moreover, by the definition of K^L_i and $u^\star_i$, we have $(1 - f_j) \leq u^\star_i$ for all $j \in K^L_i$. Therefore, we can see that

$$\lfloor f_j + u_i \rfloor = \lfloor f_j + u^\star_i \rfloor,\ \forall j \in K_i,\ \forall i \in I.$$

In addition, since the cut is non-dominated, we have $\lfloor u^\star_0 b + \sum_{i \in I} u_i \rfloor = \lfloor u^\star_0 b + \sum_{i \in I} u^\star_i \rfloor$. Therefore, the result follows. □

2.4 Exact separation algorithm for CG cuts

We first analyze the separation problem associated with CG cuts and then show that it can be solved in pseudo-polynomial time. The separation problem checks whether

$\hat{x} \in \mathcal{P}_G^1$ for a given $\hat{x} \in \mathcal{P}_G$. If $\hat{x} \notin \mathcal{P}_G$, it generates a CG cut that cuts off $\hat{x}$. From Proposition 2.2, the separation problem, which we call SEP, can be formulated as an optimization problem as follows.

$$\begin{aligned} \text{SEP:} \quad \max \quad & \sum_{i \in I} \sum_{j \in K_i} \lfloor u_0 a_j + u_i \rfloor \hat{x}_j - \left\lfloor u_0 b + \sum_{i \in I} u_i \right\rfloor \\ \text{s.t} \quad & (u_0, u) \in U. \end{aligned}$$

If the optimal objective value of SEP is less than or equal to 0, it implies that $\hat{x} \in \mathcal{P}_G^1$. Otherwise, an optimal solution to SEP exists, which yields a non-dominated CG cut. Let us consider the special case of SEP where u_0 is fixed to a given positive number $q_0 < 1$, denoted as SEP(q_0). Then, SEP(q_0) can be stated as

$$\begin{aligned} \text{SEP}(q_0): \quad \max \quad & \sum_{i \in I} \sum_{j \in K_i} \lfloor q_0 a_j + u_i \rfloor \hat{x}_j - \left\lfloor q_0 b + \sum_{i \in I} u_i \right\rfloor \\ \text{s.t} \quad & 0 \leq u_i < 1, \forall\, i \in I. \end{aligned}$$

Now, let $D := \bigcup_{i \in I} \{|a_l - a_k| : l, k \in K_i \cup \{0\},\ l \neq k\}$. The following theorem states that SEP can be decomposed into a finite number of sub-problems.

Theorem 2.1. *Let Q_0 be the set of rational numbers, t/d, for all pair of positive integers d and t such that $d \in D$ and $t < d$. Then, SEP can be solved by solving SEP(q_0) for all $q_0 \in Q_0$, whose number is $O(n^2 \bar{a})$, where $\bar{a} = \max_{j \in N}\{a_j\}$.*

Proof. There exists an optimal solution $(u_0^\star, u^\star) \in U$ to SEP, which yields a non-dominated CG cut. This observation implies that, by Proposition 2.3, there exists $(u_0^\star, u^\star)$ such that $u_0^\star = t/d$, for some $d \in D$ and a positive integer $t < d$. Such an

optimal solution is also optimal for SEP(q_0) for some $q_0 \in Q_0$ by the definition of Q_0. It means that SEP can be solved by solving SEP(q_0) for all $q_0 \in Q_0$.

The number of elements in D is $O(n^2)$ while the number of possible choices of t for each $d \in D$ is $O(\bar{a})$ since $d \leq \bar{a}$ for each $d \in D$. Therefore, the number of elements in Q_0 is $O(n^2\bar{a})$, which follows the result. □

In the proof of Theorem 2.1, the optimal solution to SEP, $(u_0^\star, u^\star)$, yields a non-dominated CG cut while it is obtained from SEP(q_0) for some $q_0 \in Q_0$. Therefore, it is sufficient to consider $u \in [0,1)^n$ that can yield a non-dominated CG cut in SEP(q_0) for each $q_0 \in Q_0$. Proposition 2.4 allows us to narrow down the choice of u, and the resulting reformulation of SEP(q_0) is described as follows.

$$\begin{aligned} \text{SEP}(q_0): \quad \max \quad & \sum_{i\in I}\sum_{j\in K_i^+} \lfloor f_j + u_i \rfloor \hat{x}_j - \left\lfloor f_0 + \sum_{i\in I} u_i \right\rfloor + C \\ \text{s.t} \quad & u_i \in \{0\} \cup \{1 - f_j : j \in K_i^+\}, \forall\, i \in I, \end{aligned}$$

where $f_0 = q_0 b - \lfloor q_0 b \rfloor$ and $f_j = q_0 a_j - \lfloor q_0 a_j \rfloor$ for each $j \in N$. Here, $C = \sum_{i\in I}\sum_{j\in K_i} \lfloor q_0 a_j \rfloor \hat{x}_j - \lfloor q_0 b \rfloor$ and $K_i^+ = \{j \in K_i : f_j > 0 \text{ and } \hat{x}_j > 0\}$ for all $i \in I$. Additionally, we define $N^+ = \cup_{i\in I} K_i^+$. Since C is a constant for a given q_0, we omit it in the subsequent discussion.

Before presenting an exact algorithm for SEP, we show a negative result on the computational complexity of SEP(q_0) for a given $q_0 \in Q_0$.

Theorem 2.2. *SEP(q_0) for a given $q_0 \in Q_0$ is NP-hard.*

Proof. Let us show that a special case of SEP(q_0) where $I = N$ and $K_i = \{i\}$ for all $i \in I$, which we call SP(q_0), is NP-hard by showing that the corresponding decision

problem, DSP(q_0), is NP-complete. DSP(q_0) is stated as follows: For given integers b and a_j for all $j \in N$, $\hat{x} \in \mathcal{P}_G$, $q_0 \in Q_0$, and an integer L, the problem is to determine whether there exists a subset $J \subseteq N^+$ such that $\sum_{j\in J} \hat{x}_j - \lfloor f_0 + \sum_{j\in J}(1-f_j)\rfloor \geq L$. In this special case of SEP(q_0), $q_0 = t/a_k$ for some $k \in N$ and a positive integer $t < a_k$ by the definition of Q_0.

It is clear that DSP(q_0) is in NP. We show that every instance of the well-known PARTITION problem, which is NP-complete (Garey & Johnson, 1979), can be polynomially transformed into an instance of DSP(q_0). An instance of PARTITION consists of a finite set $A = \{1, \ldots, n\}$, a positive integer B, and positive integers r_a for all $a \in A$ such that $\sum_{a\in A} r_a = 2B$. The question is whether or not there exists a subset $S \subseteq A$ such that $\sum_{a\in S} r_a = B$. For a given instance of PARTITION, the corresponding instance of DSP(q_0) can be constructed as follows. First, set $N = A \cup \{n+1\}$, $a_{n+1} = 2B+1$, $a_j = (2B+1) - 2r_j$ for all $j \in N \setminus \{n+1\}$, $q_0 = 1/a_{n+1}$, and $L = 1$. Next, set $\hat{x}_1 = r_1/2B$, $\hat{x}_j = r_j/B$ for all $j \in \{2, \ldots, n\}$, and $\hat{x}_{n+1} = 1$. Finally, set $b = \beta(2B+1)$, where β is the smallest positive integer such that $\sum_{j\in N} a_j\hat{x}_j \leq \beta(2B+1)$. Note that $\beta < n+1$ because $\hat{x}_{n+1} = 1$, $\hat{x}_j < 1$ for all $j \in N \setminus \{n+1\}$, and $a_j \leq (2B+1)$ for all $j \in N$. In addition, from the construction, $f_0 = 0$, $f_{n+1} = 0$, $f_j = 2r_j$ for all $j \in N \setminus \{n+1\}$, and $N^+ = N \setminus \{n+1\}$.

Now, suppose that there exists $S \subseteq A$ such that $\sum_{a\in A} r_a = B$. Without loss of generality, we can assume that $1 \notin S$. Then, if we set $J = S$, it follows that $\sum_{j\in J\cap N^+} \hat{x}_j = 1$ and $f_0 + \sum_{j\in J\cap N^+}(1-f_j) < 1$. Therefore, J is a solution to the instance of DSP(q_0). Conversely, suppose that $J \subseteq N^+$ is a solution to the instance of DSP(q_0). Since $\sum_{j\in N^+} \hat{x}_j < 2$ and $\lfloor f_0 + \sum_{h\in N^+}(1-f_j)\rfloor = 1$, it is clear that $\sum_{j\in J} \hat{x}_j \geq 1$ and $\lfloor f_0 + \sum_{h\in J}(1-f_j)\rfloor = 0$. Now, we show that $\sum_{j\in J} \hat{x}_j = 1$

and $1 \notin J$, which means $\sum_{j\in J} 2r_j = 2B$. If $\sum_{j\in J} \hat{x}_j > 1$, then it can be easily verified that $\sum_{j\in J} 2r_j \geq 2B+1$ and $\lfloor f_0 + \sum_{h\in J}(1-f_j)\rfloor = 1$. If $\sum_{j\in J} \hat{x}_j = 1$ but $1 \in J$, then $r_1 + \sum_{j\in J\setminus\{1\}} 2r_j = 2B$, which means $\lfloor f_0 + \sum_{h\in J}(1-f_j)\rfloor = 1$ because $\sum_{j\in J} 2r_j = 2B + r_j \geq 2B+1$. Hence, by setting $S = J$, $\sum_{a\in S} r_a = B$, which means S is a solution to the instance of PARTITION. Therefore, the result follows. □

We show that SEP(q_0) is weakly NP-hard in the following proposition by presenting a pseudo-polynomial time algorithm for the problem. To this end, let us reformulate SEP(q_0) as a nonlinear integer program.

Recall that $q_0 \in Q_0$ is t/d for some $d \in D$ and a positive integer $t < d$. Then, $f_0 = 1 - p_0/d$ for some integer $0 < p_0 \leq d$ and $f_j = 1 - p_j/d$ for some integer $0 < p_j < d$ for all $j \in K_i^+$ for all $i \in I$. Now, let us define a binary variable y_j for each $j \in K_i^+$ and $i \in I$ such that $u_i = 1 - f_j$ if $y_j = 1$. For each $i \in I$, if $u_i = 1 - f_j$ for some $j \in K_i^+$, then

$$\sum_{k\in K_i^+} \lfloor f_k + u_k \rfloor \hat{x}_j = \sum_{\{k\in K_i^+ : f_k \geq f_j\}} \hat{x}_k$$

because $\lfloor f_k + u_i \rfloor = 1$ if $f_k \geq f_j$, and 0, otherwise for each $k \in K_i^+$. Then, the second term of the objective function can be written as

$$\Big\lfloor f_0 + \sum_{i\in I} u_i \Big\rfloor = \Big\lfloor f_0 + \sum_{i\in I} \sum_{j\in K_i^+} (1-f_j) y_j \Big\rfloor,$$

which in turn represented as $\lfloor (d - p_0 + w)/d \rfloor$ by introducing an integer variable w

such that

$$w = \sum_{i \in I} \sum_{j \in K_i^+} p_j y_j.$$

Hence, SEP(q_0) can be reformulated as the following nonlinear integer program, NIP(q_0), where $q_0 = t/d$ for some $d \in D$ and a positive integer $t < d$.

$$\begin{aligned}
\text{NIP}(q_0): \quad \max \quad & \sum_{i \in I} \sum_{j \in K_i^+} s_j y_j - \lfloor (d - p_0 + w)/d \rfloor + \mathcal{C} \\
\text{s.t} \quad & \sum_{i \in I} \sum_{j \in K_i^+} p_j y_j = w && (2.6) \\
& \sum_{j \in K_i^+} y_j \leq 1, \forall\, i \in I && (2.7) \\
& y_j \in \{0,1\}, \forall\, j \in N^+ \\
& w \in \mathbb{Z}_+,
\end{aligned}$$

where $s_j = \sum_{\{k \in K_i^+ : f_k \geq f_j\}} \hat{x}_k$ for all $j \in K_i^+$ and $i \in I$.

Proposition 2.5. *For a given $q_0 \in Q_0$ such that $q_0 = t/d$ for some $d \in D$ and a positive integer $t < d$, SEP(q_0) can be solved in $O(nm\bar{a})$ where $\bar{a} = \max_{j \in N} a_j$.*

Proof. We establish the result by showing that NIP(q_0) can be solved in pseudo-polynomial time. Let $F(\kappa, \nu)$ be the optimal objective value of the following integer program for each pair of $\kappa = 1, \ldots, m$ and $\nu = 0, 1, \ldots, m\bar{a}$.

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{\kappa} \sum_{j \in K_i^+} s_j y_j \\
\text{subject to} \quad & \sum_{i=1}^{\kappa} \sum_{j \in K_i^+} p_j y_j = \nu
\end{aligned}$$

$$\sum_{j \in K_i^+} y_j \leq 1, \forall\, i = 1, \ldots, \kappa$$

$$y_j \in \{0, 1\}, \forall\, j \in N^+.$$

Let η be the optimal objective value of NIP(q_0). Because $\sum_{j \in K_i^+} y_j \leq 1$ for all $i \in I$ and $p_j < d \leq \bar{a}$, it holds that $\sum_{i \in I} \sum_{j \in K_i^+} p_j y_j \leq m\bar{a}$. Therefore, it is clear that

$$\eta = \max_{0 \leq \nu \leq m\bar{a}} \{F(m, \nu) - \lfloor (d - p_0 + \nu)/d \rfloor + C\}.$$

Now, we show that $F(\kappa, \nu)$ for each pair of κ and ν can be computed recursively as follows. For the recursion, define $F(0, 0) = 0$ and $F(0, \nu) = -\infty$ for $r = 1, \ldots, m\bar{a}$. Then, for each $\kappa = 1, \ldots, m$ and $\nu = 0, 1, \ldots, m\bar{a}$,

$$F(\kappa, \nu) = \max \left\{ F(\kappa - 1, \nu), \max_{\{j \in K_\kappa^+ : p_j \leq \nu\}} \{F(\kappa - 1, \nu - p_j) + s_j\} \right\},$$

which takes at most $(n_\kappa + 1)$ operations because $|K_\kappa^+| \leq |K_\kappa| = n_\kappa$. Hence, for a given κ, it requires $O(n_\kappa m\bar{a})$ time to compute $F(\kappa, \nu)$ for all $\nu = 0, 1, \ldots, m\bar{a}$, which implies that computing $F(\kappa, \nu)$ for all pairs of κ and ν can be performed in $O(\sum_{i \in I} n_i m\bar{a}) = O(nm\bar{a})$ operations. In addition, η can be computed in $O(m\bar{a})$. Therefore, there exists a pseudo-polynomial time algorithm for SEP(q_0) whose computational complexity is $O(nm\bar{a})$. □

The detailed algorithm for the SEP(q_0) where $q_0 = t/d$ for some $d \in D$ and an integer t such that $t < d$ is described in Algorithm 3. For the sake of brevity, we describe the construction of the SEP(q_0) for each $q_0 \in Q_0$ separately with Algorithm 2.

Algorithm 2 Construction of the SEP(q_0)

procedure CONSTRUCT($t, d, \hat{x}$)
  $p_0 \leftarrow d - (tb \mod d)$, $p_j \leftarrow d - (ta_j \mod d)$ for all $j \in N$;
  $f_0 \leftarrow 1 - p_0/d$, $f_j \leftarrow 1 - p_j/d$ for all $j \in N$;
  $K_i^+ \leftarrow \{j \in K_i : f_j > 0 \text{ and } \hat{x}_j > 0\}$ for all $i \in I$;
  $s_j = \sum_{k \in K_i^+ : f_k \geq f_j} \hat{x}_k$ for all $j \in K_i^+$ and $i \in I$;
  return $\mathbf{p} \leftarrow (p_0, \ldots, p_n)$, $\mathbf{s} \leftarrow (s_1, \ldots, s_n)$, $K^+ = \cup_{i \in I} K_i^+$;
end procedure

Algorithm 3 Exact algorithm for the SEP(q_0)

procedure SUB($\mathbf{p}, \mathbf{s}, K^+$)
  $F(0,0) \leftarrow 0$, $F(0,\nu) \leftarrow -\infty$ for all $\nu = 1, \ldots, m\bar{a}$;
  for $1 \leq \kappa \leq m$ and $0 \leq \nu \leq m\bar{a}$ **do**
    $F(\kappa,\nu) \leftarrow F(\kappa-1,\nu)$, $P(\kappa,\nu) \leftarrow 0$;
    for $j \in K_\kappa^+$ **do**
      if $p_j \leq \nu$ and $F(\kappa,\nu) < F(\kappa-1,\nu-p_j) + s_j$ **then**
        $F(\kappa,\nu) \leftarrow F(\kappa-1,\nu-p_j) + s_j$;
        $P(\kappa,\nu) \leftarrow p_j$;
      end if
    end for
  end for
  return F,P ;
end procedure

By Proposition 2.5 together with Theorem 2.1 that says SEP can be solved by solving SEP(q_0) for all $q_0 \in Q_0$, there exists a pseudo-polynomial time algorithm for SEP.

Theorem 2.3. *SEP can be solved in* $O(n^3 m\bar{a}^2)$.

Proof. By Theorem 2.1, the number of possible values of $q_0 \in Q_0$ is $O(n^2\bar{a})$. In addition, SEP(q_0) for a given $q_0 \in Q$ can be solved in $O(nm\bar{a})$ by Proposition 2.5. Therefore, the result follows. □

The details of an exact algorithm for SEP based on Theorem 2.3 is described in Algorithm 4 which returns the most violated CG cut and the violation $\eta^\star$ for a

Algorithm 4 Exact algorithm for the SEP

procedure EXT($\hat{x}$)
  $D \leftarrow \bigcup_{i \in I} \{|a_l - a_k| : l, k \in K_i \cup \{0\}, l \neq k\}$;
  $\eta^\star \leftarrow -\infty$, $u_0^\star \leftarrow 0$, $u_i^\star \leftarrow 0$ for all $i \in I$;
  $\bar{a} \leftarrow \max_{j \in N} a_j$
  for $d \in D$ and $1 \leq t < d$ **do**
    $q_0 \leftarrow t/d$, $C \leftarrow \sum_{i \in I} \sum_{j \in K_i} \lfloor q_0 a_j \rfloor \hat{x}_j - \lfloor q_0 b \rfloor$;
    $\mathbf{p}, \mathbf{s}, K^+ \leftarrow$CONSTRUCT$(t, d, \hat{x})$;
    $F, P \leftarrow$SUB$(\mathbf{p}, \mathbf{s}, K^+)$;
    $\eta \leftarrow \max_{0 \leq \nu \leq m\bar{a}} \{F(m, \nu) - \lfloor (d - p_0 + \nu)/d \rfloor + C\}$;
    $\nu^\star \leftarrow \arg\max_{0 \leq \nu \leq m\bar{a}} \{F(m, \nu) - \lfloor (d - p_0 + \nu)/d \rfloor + C\}$;
    if $\eta^\star < \eta$ **then**
      $\eta^\star \leftarrow \eta$, $u_0^\star \leftarrow q_0$;
      for $i = m, \ldots, 1$ **do**
        $u_i^\star \leftarrow P(i, \nu^\star)/d$;
        $\nu^\star \leftarrow \nu^\star - P(i, \nu^\star)$;
      end for
    end if
  end for
  return $\eta^\star$, $u_0^\star$, and $u_i^\star$ for all $i \in I$;
end procedure

given $\hat{x} \in \mathcal{P}_G$. If $\eta^\star > 0$, then the obtained CG cut separates $\hat{x}$. Otherwise, $\hat{x} \in \mathcal{P}_G^1$, that is, $\hat{x}$ satisfies all the CG cuts of the form (2.4).

We note that the computational complexity of the CG cut separation problem for the GKP is not established in this study. The problem may be NP-hard or even P. However, Theorem 2.3 implies that, at least, the problem is not strongly NP-hard, whereas the separation problem for general integer linear programs has been proven to be NP-hard in a strong sense (Eisenbrand, 1999).

2.5 Heuristic separation algorithm for CG cuts

Even though the separation problem can be solved in pseudo-polynomial time, it may require a significant amount of computation, especially for large values of a_j. Therefore, we propose an efficient separation heuristic based on the decomposition

property of the problem.

The proposed heuristic consists of two steps. First, we restrict the range of t of $q_0 = t/d$, where $q_0 \in Q_0$, to reduce the number of sub-problems that need to be solved in the heuristic. We only focus on the sub-problems where the most violated CG cuts may be derived. Next, we obtain the feasible solution of each sub-problem in a greedy manner, following several variable ordering strategies.

We discuss the selection of sub-problems in Section 2.5.1, and present the greedy algorithm for each sub-problem in Section 2.5.2.

2.5.1 Selection of sub-problems to be solved

The computational complexity of the exact algorithm presented in Section 2.4 is affected by the number of sub-problems, which depends on the values of a_j's. While the exact algorithm solves all the sub-problems, the proposed heuristic considers only some with a high possibility of yielding the most violated CG cuts.

Let us consider a non-dominated CG cut with $u_0 = q_0$ for some $q_0 \in Q_0$ and $u_i = 1 - f_i^\star$ for each $i \in I$ where $0 < f_i^\star \leq 1$ for each $i \in I$. In addition, let $\bar{K}_i = \{j \in K_i : f_j \geq f_i^\star\}$ where $f_j = q_0 a_j - \lfloor q_0 a_j \rfloor$ for each $j \in N$. Then, the CG cut can be expressed as follows.

$$\sum_{i \in I} \sum_{j \in \bar{K}_i} (q_0 a_j + 1 - f_j) x_j + \sum_{i \in I} \sum_{j \in K_i \setminus \bar{K}_i} (q_0 a_j - f_j) x_j \leq \left\lfloor q_0 b + \sum_{i \in I} (1 - f_i^\star) \right\rfloor$$

For given $\hat{x} \in \mathcal{P}_G$, the violation $\eta^\star$ by the above cut is as follows.

$$\eta^\star = \sum_{i \in I} \sum_{j \in \bar{K}_i} (q_0 a_j + 1 - f_j) \hat{x}_j + \sum_{i \in I} \sum_{j \in K_i \setminus \bar{K}_i} (q_0 a_j - f_j) \hat{x}_j - \left\lfloor q_0 b + \sum_{i \in I} (1 - f_i^\star) \right\rfloor$$

or, equivalently

$$\eta^\star = -q_0 \left(b - \sum_{i \in I} \sum_{j \in K_i} a_j \hat{x}_j \right) + \sum_{i \in I} \sum_{j \in \bar{K}_i} (1 - f_j) \hat{x}_j - \sum_{i \in I} (1 - f_i^\star) - \sum_{i \in I} \sum_{j \in \bar{K}_i} f_j \hat{x}_j + f_0, \tag{2.8}$$

where $f_0 = q_0 b + \sum_{i \in I} (1 - f_i^\star) - \lfloor q_0 b + \sum_{i \in I} (1 - f_i^\star) \rfloor$.

We observed that $b - \sum_{i \in I} \sum_{j \in K_i} a_j \hat{x}_j \geq 0$ because $\hat{x} \in \mathcal{P}_G$. In addition,

$$\sum_{i \in I} \sum_{j \in \bar{K}_i} (1 - f_j) \hat{x}_j - \sum_{i \in I} (1 - f_i^\star) \leq 0$$

since

$$\sum_{i \in I} \sum_{j \in \bar{K}_i} (1 - f_j) \hat{x}_j - \sum_{i \in I} (1 - f_i^\star) \leq \sum_{i \in I} (1 - f_i^\star) (\sum_{j \in \bar{K}_i} \hat{x}_j - 1) \leq 0.$$

From these observations, f_0 is the only term that makes the violation (2.8) can have a positive value. However, since $f_0 < 1$, the rest terms in the violation (2.8) should have small values to make the CG cut separate $\hat{x}$. We note that $q_0 = t/d$ for some $d \in D$ and an integer t such that $t < d$. As t gets large, corresponding CG cuts may fail to separate $\hat{x}$ because it increases $q_0 \left(b - \sum_{i \in I} \sum_{j \in K_i} a_j \hat{x}_j \right)$. Moreover, $b - \sum_{i \in I} \sum_{j \in K_i} a_j \hat{x}_j$ also increases as CG cuts are added to the formulation of the GKP. These observations lead to exploiting the sub-problems with small t has a high chance of identifying the most violated CG cut.

The CG cut obtained from the sub-problem with small t has an additional benefit in the perspective of depth. Depth of a given valid inequality $\beta^T x \leq \beta$ separating $\hat{x}$ is defined as the shortest distance from $\hat{x}$ to the inequality, i.e., $|\beta^T \hat{x} - \beta| / \|\beta\|_2$, and it can be used to estimate the effectiveness of valid inequalities. Because the depth implicitly represents the region's volume separated by the inequality, deeper valid

inequalities may improve the LP relaxation more than shallow ones. In this point of view, the CG cut obtained from the sub-problem with a small t may be more effective than the others derived from sub-problems with larger t's and the same d if the violations are comparable.

For these reasons, the heuristic for the separation problem solves only sub-problems with $t \leq T$ for the given parameter T, i.e., $O(Tn^2)$ sub-problems are solved. Hence, if T is polynomially bounded, the heuristic solves the polynomial number of sub-problems.

2.5.2 Greedy algorithm for each sub-problem

While the number of sub-problems is reduced in the heuristic, solving each sub-problem is still time-consuming. Even though it can be solved in pseudo-polynomial time by Proposition 2.5, each sub-problem may require a huge amount of computation for large a_j's. Therefore, using a greedy algorithm, the proposed heuristic efficiently obtains a solution for each sub-problem, which may not be optimal.

The sub-problem SEP(q_0) is formulated as a nonlinear integer program, NIP(q_0). When the values of y_j's are given while satisfying the constraint (2.7), the value of w is uniquely determined by the constraint (2.6), and the corresponding objective value can be evaluated. Our greedy algorithm generates a sequence of feasible solutions for NIP(q_0) following a given sequence of variables y_j's and compares their objective values. Specifically, let $\tau = (\tau_1, ..., \tau_{n^+})$ be the sequence where $n^+ = |N^+|$, and let $G(j)$ denote $i \in I$ such that $j \in K_i^+$. For each $k \in N^+$, a solution for NIP(q_0),

$(y^k, w^k) \in \mathbb{R}^{n^+ + 1}$, is recursively constructed as

$$y_j^k = \begin{cases} 1, & j = \tau_k \\ 0, & j = L_k(G(\tau_k)) \\ y_j^{k-1}, & \text{otherwise.} \end{cases},$$

and

$$w^k = w^{k-1} + p_{\tau_k} - p_{L_k(G(\tau_k))},$$

where $y^0 = (0, ..., 0)$, and $L_k(i) = l$ for some $l \in K_i^+$ such that $y_l^{k-1} = 1$ if $\sum_{j \in K_i^+} y^{k-1} = 1$, otherwise 0.

The above construction ensures the feasibility of (y^k, w^k) for NIP(q_0). Therefore, n^+ different feasible solutions for NIP(q_0) can be obtained. The objective values of NIP(q_0) for each feasible solution are compared, and our greedy algorithm returns the corresponding $(y^{k^\star}, w^{k^\star})$ that yields the maximum value. The feasible solution of SEP(q_0) is then recovered from $(y^{k^\star}, w^{k^\star})$, which yields a CG cut with the same violation as the objective value of NIP(q_0) corresponding to $(y^{k^\star}, w^{k^\star})$. The greedy algorithm can be implemented with $O(n)$ complexity. Details on the algorithm can be found in Algorithm 5.

We note that each sub-problem can be solved exactly using the greedy algorithm if a proper sequence τ is given. Of course, identifying such a sequence may be as difficult as solving the sub-problem exactly. Therefore, we propose three variable ordering strategies to obtain τ, based on the relationship between NIP(q_0) and the GKP.

The goal of sub-problems is to find a violated CG cut for given $\hat{x}$, i.e., to find the

Algorithm 5 Greedy algorithm for SEP(q_0)

procedure GREEDY($\tau, \mathbf{p}, \mathbf{s}, K^+$)
  $\eta^\star_{sub} \leftarrow -\infty$, $k^* \leftarrow 0$, $\hat{u}_i \leftarrow 0$ for all $i \in I$;
  $\eta_{sub} \leftarrow 0$, $\hat{w} \leftarrow 0$, $n^+ \leftarrow \sum_{i \in I} |K_i^+|$, $L(i) \leftarrow 0$ for all $i \in I$;
  for $k = 1, \ldots, n^+$ **do**
    $\hat{w} \leftarrow \hat{w} + p_{\tau_k}$, $\eta_{sub} \leftarrow \eta_{sub} + s_{\tau_k}$;
    if $L(G(\tau_k)) > 0$ **then**
      $\hat{w} \leftarrow \hat{w} - p_{L(G(\tau_k))}$, $\eta_{sub} \leftarrow \eta_{sub} - s_{L(G(\tau_k))}$;
    end if
    $L(G(\tau_k)) \leftarrow \tau_k$;
    if $\eta^\star_{sub} < \eta_{sub} + \lfloor (d - p_0 + \hat{w})/d \rfloor$ **then**
      $\eta^\star_{sub} \leftarrow \eta_{sub} + \lfloor (d - p_0 + \hat{w})/d \rfloor$, $k^* \leftarrow k$;
    end if
  end for
  $L^\star(G(\tau_k)) \leftarrow \tau_k$ for all $k = 1, \ldots, k^\star$;
  $\hat{u}_i \leftarrow 1 - f_{L^\star(i)}$ for all $i \in I$;
  return $\eta^\star_{sub}$, $\hat{u} = (\hat{u}_1, \ldots, \hat{u}_m)$;
end procedure

values of y_j's in NIP(q_0) where the corresponding objective value is greater than 0. Hence, it seems natural to select y_j's with large s_j's by priority. The first ordering strategy (`S1`) sorts the variables in descending order of s_j. Such an order β satisfying

$$s_{\tau_1} \geq s_{\tau_2} \geq \cdots \geq s_{\tau_{n^+}},$$

which can be constructed in $O(n \log n)$ computations.

Example 2.1. *Let* $N^+ = \{1, 2, 3, 4, 5\}$, $I = \{1, 2\}$, $K_1^+ = \{1, 2\}$ *and* $K_2^+ = \{3, 4, 5\}$. *In addition, let* $(s_1, s_2, s_3, s_4, s_5) = (0.4, 0.7, 0.4, 0.5, 0.9)$ *and* $(p_1, p_2, p_3, p_4, p_5) = (4, 6, 2, 8, 10)$. *Let us consider NIP*($q_0$) *for some* $q_0 \in Q_0$, *which is defined with these inputs as follows.*

$$\begin{aligned} \max \quad & 0.4y_1 + 0.7y_2 + 0.4y_3 + 0.5y_4 + 0.9y_5 - \lfloor (d - p_0 + w)/d \rfloor + C \\ s.t \quad & 4y_1 + 6y_2 + 2y_3 + 8y_4 + 10y_5 = w \end{aligned}$$

$$y_1 + y_2 \leq 1$$

$$y_3 + y_4 + y_5 \leq 1$$

$$y_j \in \{0, 1\}, \forall\, j \in N^+$$

$$w \in \mathbb{Z}_+,$$

Then, `S1` *constructs* $\tau = (5, 2, 4, 1, 3)$.

We note that the formulation of NIP(q_0) is valid even if the constraint (2.6) is replaced with an inequality,

$$\sum_{i \in I} \sum_{j \in K_i^+} d_i p_j y_j \leq w. \tag{2.9}$$

If variable w is fixed to $\hat{w}$, and the constraints (2.7) are ignored, the rest of the modified formulation represents a binary knapsack problem, where c_j and a_j of each variable $j \in N^+$ are s_j and p_j, respectively while $b = \hat{w}$. For the binary knapsack problem, a 1/2-approximation solution can be obtained by examining the variables in descending order of the marginal profit, c_j/a_j, which is independent of b. Our second ordering strategy (`S2`) adopts the sequence and sorts the variables in descending order of s_j/p_j. Then, τ can be obtained in $O(n \log n)$, satisfying

$$\frac{s_{\tau_1}}{p_{\tau_1}} \geq \frac{s_{\tau_2}}{p_{\tau_2}} \geq \cdots \geq \frac{s_{\tau_{n^+}}}{p_{\tau_{n^+}}}.$$

If p_j's are the same for all $j \in N^+$, the second strategy is equivalent to the first one.

Example 2.1 (Continued)**.**

$$\frac{s_3}{p_3} \geq \frac{s_2}{p_2} \geq \frac{s_1}{p_1} \geq \frac{s_5}{p_5} \geq \frac{s_4}{p_4}.$$

Therefore, `S2` *constructs* $\tau = (3, 2, 1, 5, 4)$.

Suppose w is fixed to $\hat{w}$ in NIP(q_0) defined with the constraint (2.9). In that case, the formulation represents another GKP. Johnson & Padberg (1981) proposed an algorithm to solve the LP relaxation of the GKP, which examines variables of the problem that can have non-zero values sequentially. Our third ordering strategy adopts the sequence of variables examined in the algorithm. Specifically, the variables are sorted in descending order of s_j/p_j, and let k be the variable with the maximum value. Then, the variable in $G(k)$ with the minimum p_j, which we denote as l, is eliminated from $G(k)$, and p_j's and s_j's of the rest variables in $G(k)$ are replaced with $p_j - p_l$ and $s_j - s_l$, respectively. This procedure is iterated until all variables are eliminated, and τ is the sequence of eliminated variables through the iteration. Under this strategy, τ can be constructed in $O(n \log n)$ time using the method proposed by Glover & Klingman (1979). If K_i^+'s are singletons, `S3` is equivalent to `S2`.

Example 2.1 (Continued)**.** `S3` *constructs* τ *as follows.*

- *Iteration 1:* y_2 *has the maximum value of* $s_j/p_j = 0.7/6$*. Because* y_1 *has the minimum* p_j *in* K_1^+*,* y_1 *is eliminated from* K_1^+*, and* $\tau_1 = 1$*. Then,* s_2 *and* p_2 *are replaced with* 0.3 *and* 2*, respectively.*

- *Iteration 2:* y_3 *has the maximum value of* $s_j/p_j = 0.4/2$ *and it has the minimum* p_j *in* K_2^+*. Therefore,* y_3 *is eliminated from* K_2^+*, and* $\tau_2 = 3$*.* s_4 *and*

p_4 are replaced with 0.1 *and* 6*, respectively while s_5 and p_5 become* 0.5 *and* 8*, respectively.*

- *Iteration 3: y_2 has the maximum value of $s_j/p_j = 0.3/2$ and it has the minimum p_j in K_1^+. Hence, y_1 is eliminated from K_1^+, i.e., $K_1^+ = \emptyset$, and $\tau_3 = 2$.*

The iteration is continued until y_4 and y_5 are eliminated from K_2^+. The constructed τ is $(1, 3, 2, 4, 5)$.

With the result presented in Section 2.5.1, the overall heuristic is described in Algorithm 6. Because each sub-problem in the heuristic can be solved in $O(n \log n)$ regardless of the ordering strategy while $O(Tn^2)$ sub-problems are solved, the computational complexity of Algorithm 6 is $O(Tn^3 \log n)$ for a given parameter T. Through computational tests in Section 2.6, we compare the performance of the heuristics using different variable ordering strategies presented above.

Algorithm 6 Heuristic algorithm for the SEP

procedure HEU$_T(\hat{x})$
  $D \leftarrow \bigcup_{i \in I} \{|a_l - a_k| : l, k \in K_i \cup \{0\}, l \neq k\}$;
  $\eta^\star \leftarrow -\infty$, $u_0^\star \leftarrow 0$, $u_i^\star \leftarrow 0$ for all $i \in I$; $u_i^\star \leftarrow 0$ for all $i \in I$;
  for $d \in D$ and $1 \leq t \leq T$ **do**
    $q_0 \leftarrow t/d$, $C \leftarrow \sum_{i \in I} \sum_{j \in K_i} \lfloor q_0 a_j \rfloor \hat{x}_j - \lfloor q_0 b \rfloor$;
    $\mathbf{p}, \mathbf{s}, K^+ \leftarrow$CONSTRUCT$(t, d, \hat{x})$;
    $\tau \leftarrow$Apply `S1`, `S2`, or `S3` ;
    η_{sub}, $\hat{u} \leftarrow$GREEDY$(\tau, \mathbf{p}, \mathbf{s}, K^+)$;
    $\eta \leftarrow \eta_{sub} + C$;
    if $\eta^\star > \eta$ **then**
      $\eta^\star \leftarrow \eta$ and $u_0^\star \leftarrow q_0$;
      $u_i^\star \leftarrow \hat{u}_i$ for all $i \in I$;
    end if
  end for
  return $\eta^\star$, $u_0^\star$, and $u_i^\star$ for all $i \in I$;
end procedure

2.6 Computational experiment results

This section provides the computational test results on the performance of the proposed separation algorithms. In addition, we evaluate the effectiveness of CG cuts through comparison with general lifted GUB cover inequalities.

Computational tests used two types of instances. First, the GKP instances were randomly generated in the same manner with Sinha & Zoltners (1979). Each instance has the same number of items in GUB sets, and $a_j \in [1, \bar{a}]$ for all $j \in N$ where $\bar{a}$ denotes the data range. Second, we used benchmark instances for multi-dimensional multiple-choice knapsack problems (MMKP) and general binary integer linear programs to prove the usefulness of the CG cuts in practice. The feasible solution sets of the benchmark instances are defined with multiple GKP polytopes. The MMKP is described as

$$\begin{aligned} \max \quad & \sum_{i \in I} \sum_{j \in K_i} c_j x_j \\ \text{s.t} \quad & \sum_{i \in I} \sum_{j \in K_i} a_{kj} x_j \leq b_k, \ \forall k \in R^\star \\ & \sum_{j \in K_i} x_j \leq 1, \ \forall i \in I \\ & x \in \{0,1\}^n, \end{aligned}$$

where $|R^\star| = r^\star$. Additionally, general binary integer linear programs can be described as follows.

$$\max \quad \sum_{j \in N} c_j x_j$$

$$\text{s.t} \quad a_{kj}x_j \leq b_k,\ k \in R^\star$$

$$x \in \{0,1\}^n$$

We obtained the benchmark instances for the MMKP and general binary integer linear programs from OR-Library (Beasley, 1990) and MIPLIB 3.0 (Bixby, Ceria, et al., 1998), respectively. We refer to them as the MMKP instances and MIPLIB instances, respectively.

While GUB sets are explicitly given in the MMKP instances, MIPLIB instances do not provide GUB sets explicitly. To define GUB sets in MIPLIB instances, we employed the GUB set identification strategy proposed by Gu et al. (1998). With the number of identified GUB sets, the summary of both instances is given in Appendix A.

Cuts were generated using the following cutting plane algorithms.

(Step 1) Solve the LP relaxation of the problem.

(Step 2) If the optimal solution is integral, then stop.

(Step 3) For each GKP in the problem, generate a cut using a separation algorithm. If a violated cut is identified, then add it to the original problem, then go to Step 1. Otherwise, stop.

After the cutting plane algorithm terminates, the following measures are reported.

- Integrality gap closed (IGC (%)): $100 * (z_{LP} - z_{CUT})/(z_{LP} - z_{OPT})$
- Number of generated cuts (#Cut)
- Time spent on the cutting plane algorithm (Cutting plane time (s))

- Time spent on generating each cut (Separation time (s))

where z_{OPT} represent the optimal objective value of the given problem while z_{CUT} denotes the upper bound improved by cutting plane algorithms. We note that the IGC represents the formulation-enhancing effect achieved by cutting plane algorithms.

Cutting plane algorithms may require significant computation time to complete the iteration. Hence, we set the time limit for the experiments. All algorithms were implemented using C++ with the linear programming solver provided by Xpress (Guéret et al., 2002). All tests were performed using a machine with an Intel Core i7, 3.10GHz CPU, and 16GB RAM.

2.6.1 Performance of exact and heuristic separation algorithms for CG cuts

We evaluated the performance of the exact and heuristic separation algorithms on small-sized GKP instances. For each m, n_i, and $\bar{a}$, we generated 30 instances, and averages of performance measures were reported. The results obtained using exact and heuristic separation algorithms are presented as "ECG" and "HCG", respectively. In the HCG, `S1`, `S2`, and `S3` refer to the results using each variable ordering strategy, respectively. The "MIP" indicates the result using another exact separation method for CG cuts proposed by Fischetti & Lodi (2007), which generates CG cuts by solving the mixed-integer program representing SEP using the commercial optimization solver.

We have set 600 seconds time limit for cutting plane algorithms, and the parameter of the HGC was set to $T = \min\{m, n_i\}$.

Table 2.1 to 2.4 show the results where "$*$" indicates that the cutting plane algorithm did not terminate in the time limit for some instances of the type. For this experiment, we only present results of the HCG using `S3` because there were no significant differences between variable ordering strategies. The overall experiment results can be found in Appendix B.1.

Table 2.1: IGC (%) by each separation algorithm

n	m	n_i	$\bar{a} = 100$			$\bar{a} = 200$		
			MIP	ECG	HCG (`S3`)	MIP	ECG	HCG (`S3`)
50	5	10	99.83	99.72	99.60	100.00	100.00	100.00
	10	5	98.86	98.86	98.30	99.56	99.56	98.46
100	5	20	100.00	100.00	100.00	99.93	99.93	99.66
	10	10	*98.56	98.81	98.21	98.83	98.83	98.76
	20	5	*63.97	99.38	98.60	*66.20	98.43	97.98
Average			92.24	99.35	98.94	92.91	99.35	98.97

Table 2.2: Cutting plane time (s) by each separation algorithm

n	m	n_i	$\bar{a} = 100$			$\bar{a} = 200$		
			MIP	ECG	HCG (`S3`)	MIP	ECG	HCG (`S3`)
50	5	10	1.01	0.80	0.04	1.11	2.27	0.03
	10	5	4.11	2.23	0.03	3.31	9.62	0.03
100	5	20	5.06	2.37	0.05	8.94	14.81	0.07
	10	10	*32.24	5.84	0.05	12.39	22.94	0.05
	20	5	*547.56	23.38	0.06	*599.26	73.38	0.05
Average			134.36	6.93	0.05	125.00	24.60	0.05

Table 2.1 and 2.2 show the integrality gap closed and cutting plane times achieved by each separation algorithm. Because the MIP solves the separation problem exactly, the gap-closing effects of the MIP and ECG will be the same if both algorithms complete within the time limit. However, as shown in Table 2.2, the MIP is unfin-

ished within the time limit for some instances of $n = 100$. Hence, the ECG could close more gaps than the MIP. On the other hand, the formulation-enhancing effect of the proposed heuristic separation algorithm was comparable with exact separation algorithms.

Table 2.3: #Cut by each separation algorithm

n	m	n_i	$\bar{a} = 100$			$\bar{a} = 200$		
			MIP	ECG	HCG (S3)	MIP	ECG	HCG (S3)
50	5	10	7.7	8.2	8.0	7.2	7.2	6.3
	10	5	9.4	8.4	7.6	8.8	9.1	7.9
100	5	20	13.2	10.7	11.8	14.7	15.1	14.4
	10	10	*26.0	11.1	10.9	10.8	10.8	10.4
	20	5	*5.1	14.6	12.9	*5.3	10.4	10.0
Average			12.3	10.6	10.2	9.4	10.5	9.8

Table 2.3 provides the number of cuts generated by each separation algorithm. As previously mentioned, the MIP and ECG utilize exact separation algorithms for CG cuts, which result in the same gap-closing effect once they terminate within the time limit. However, the number of generated cuts may differ in the MIP and ECG. Because many CG cuts may have the maximum violation for a given incumbent solution $\hat{x}$, different CG cuts can be generated from the MIP and ECG. In our experiment, the difference in the number of generated cuts was negligible, and the HCG also generated a similar number of CG cuts. This result implied that the difference between cutting plane times depended on the separation times shown in Table 2.4

The excessive cutting plane time of the MIP can be explained with Table 2.4 that presents the computation time spent on generating each CG cut, i.e., separation time. When $n = 100$ and $m = 20$, the separation time of the MIP increased rapidly

Table 2.4: Separation time (s) by each separation algorithm

n	m	n_i	$\bar{a} = 100$			$\bar{a} = 200$		
			MIP	ECG	HCG (S3)	MIP	ECG	HCG (S3)
50	5	10	0.127	0.084	0.000	0.180	0.302	0.000
	10	5	0.397	0.264	0.000	0.447	1.046	0.000
100	5	20	0.273	0.169	0.000	0.410	0.717	0.000
	10	10	*1.079	0.497	0.001	1.081	1.976	0.001
	20	5	*122.930	1.580	0.001	*123.790	7.219	0.001
Average			28.041	0.519	0.001	25.182	2.252	0.001

compared to the other algorithms. The separation time of the ECG was influenced by the parameter R as we analyzed in Section 2.4. For the HCG, a value of 0.000 indicates a time of less than 0.001. The proposed heuristic separation algorithm generated CG cuts much faster than the exact algorithms, regardless of the variable ordering strategy employed. While the computational complexity of each variable ordering strategy is theoretically equivalent, in practice, using `S3` required slightly more computation time than the others.

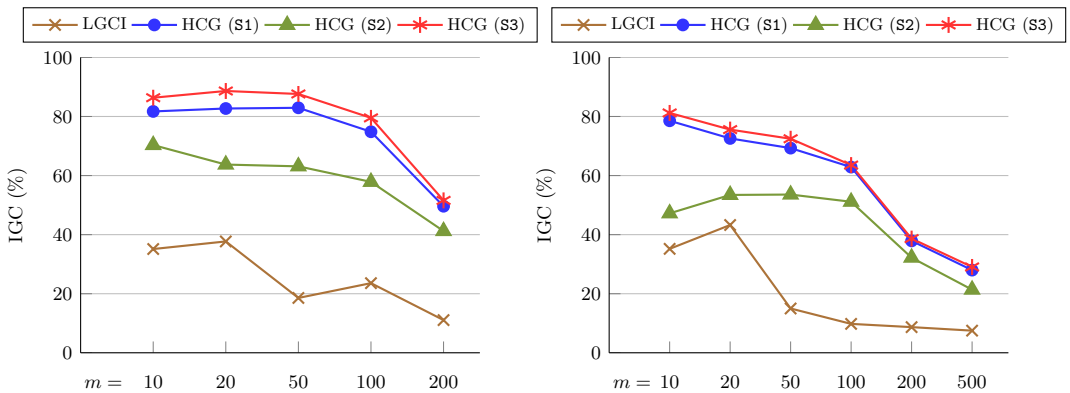
In conclusion, although the proposed exact separation algorithm was efficient compared to the MIP, it also incurred a considerable amount of time in generating CG cuts. However, our heuristic separation algorithm improved the LP relaxation as much as the exact separation algorithms in significantly less computation time.

2.6.2 Effectiveness of CG cuts compared with general lifted GUB cover inequalities

We compared the effectiveness of CG cuts generated by the proposed heuristic separation algorithm with the well-known valid inequalities, general lifted GUB cover inequalities. General lifted GUB cover inequalities were generated using the "default

algorithm" devised by Gu et al. (1998), and the "LGCI" represents the result.

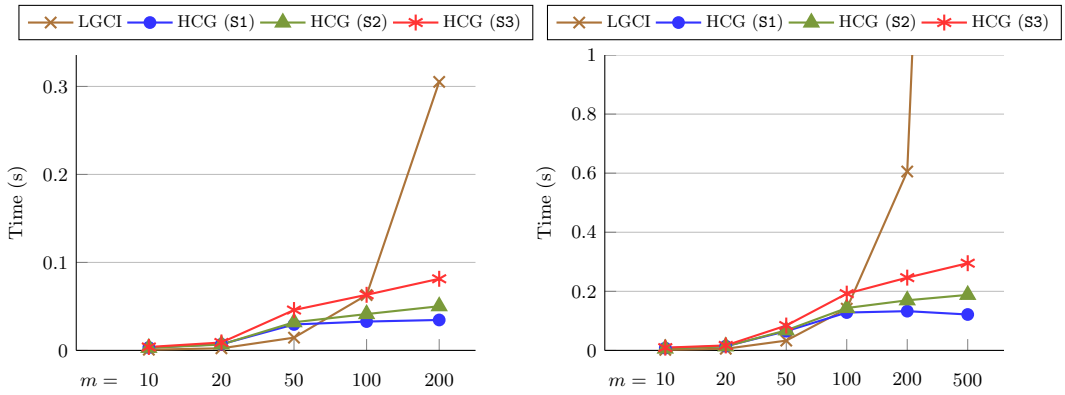
The time limit for cutting plane algorithms was 60 seconds, while the parameter of the heuristic separation algorithm was kept the same as in the previous experiment. The tests were performed on large-sized GKP instances with $n \in \{200, 500, 1000, 2000, 5000\}$. However, in this section, we only present the result of $n = 2000, 5000$ and $\bar{a} = 200$. The overall results can be found in Appendix B.2.



(a) $n = 2000$ (b) $n = 5000$

Figure 2.1: IGC (%) by each separation algorithm for $n = 2000, 5000$

Figure 2.1 illustrates the integrality gap closed by generated cuts. As mentioned in Section 2.3, general lifted GUB cover inequalities can have a Chvátal rank higher than 1, that is, using them can be more effective to enhance the given formulations in theory. However, using CG cuts could significantly improve the integrality gap rather than general lifted GUB cover inequalities, regardless of the choice of the variable ordering strategies. Specifically, the effect of the HCG depended on the choice of the variable ordering strategy. `S1` and `S3` could reduce a similar amount of the integrality gap while the effect slightly deteriorated in `S2`. The formulation-enhancing effects of the LGCI and HCG decreased as m increased. However, the



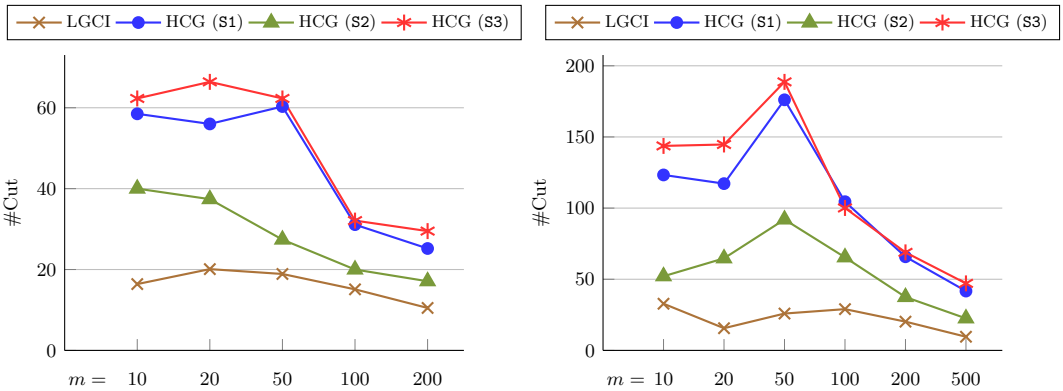
(a) $n = 2000$ (b) $n = 5000$

Figure 2.2: Separation time (s) by each separation algorithm for $n = 2000, 5000$

difference was still considerable.

Figure 2.2 shows the separation times by the separation algorithms. Each CG cut was generated within a comparable time with general lifted GUB cover inequalities. More precisely, the default algorithm to generate general lifted GUB cover inequalities required far more computation time compared to the heuristic separation algorithm for CG cuts when the problem has many GUB sets. In particular, when $n = 5000$ and $m = 500$, it took more than 5 seconds to generate general lifted GUB cover inequalities.

However, as shown in Figure 2.3, the number of generated CG cuts was significantly larger than that of general lifted GUB cover inequalities. It is worth noting that a general lifted GUB cover inequality is derived from a GUB cover inequality, which is a special case of CG cuts as proven in Section 2.3. While general lifted GUB cover inequalities may have greater effectiveness in theory, they cannot be generated if violated GUB cover inequalities are not identified. This property explains the disparity in the number of generated cuts between the two algorithms. Since CG cuts

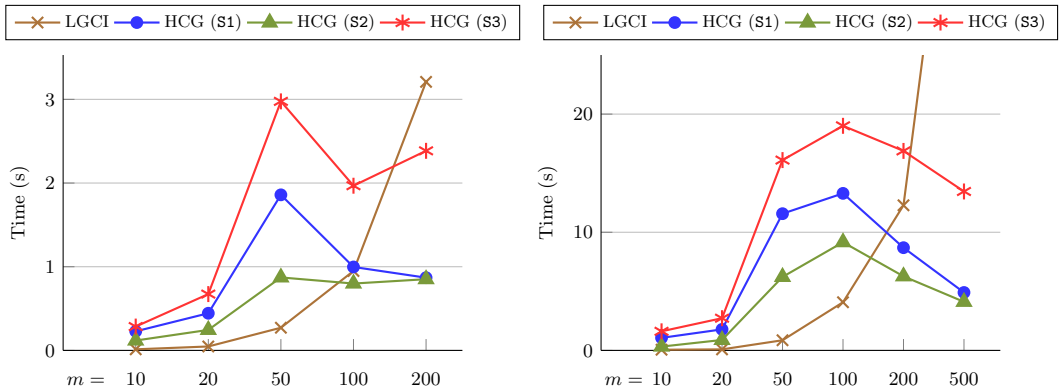


(a) $n = 2000$ (b) $n = 5000$

Figure 2.3: #Cut by each separation algorithm for $n = 2000, 5000$

dominate GUB cover inequalities, the HCG can generate violated CG cuts for a given incumbent solution, while the LGCI may fail to do so. Consequently, while the LGCI was unable to find any further violated GUB cover inequalities during the iterations of the cutting plane algorithm and terminated early, the HCG continued to identify violated CG cuts and was able to strengthen the formulation further.

A large number of generated cuts implied many iterations in the cutting plane algorithm. Therefore, even though the separation times for the HCG and the LGCI were comparable, the HCG spent much computation time on the cutting plane algorithm compared to the LGCI, as shown in Figure 2.4. Furthermore, the HCG exceeded the time limit in some instances where the LGCI terminated within the time limit. In detail, the LGCI required more computation time when the problem had many GUB sets despite generating fewer cuts, as it spent a considerable amount of separation time. Particularly, when $n = 5000$ and $m = 500$, the cutting plane time of the LGCI averaged over 50 seconds. However, in most instances, the HCG required much more cutting plane time than the LGCI due to generating many cuts. It might



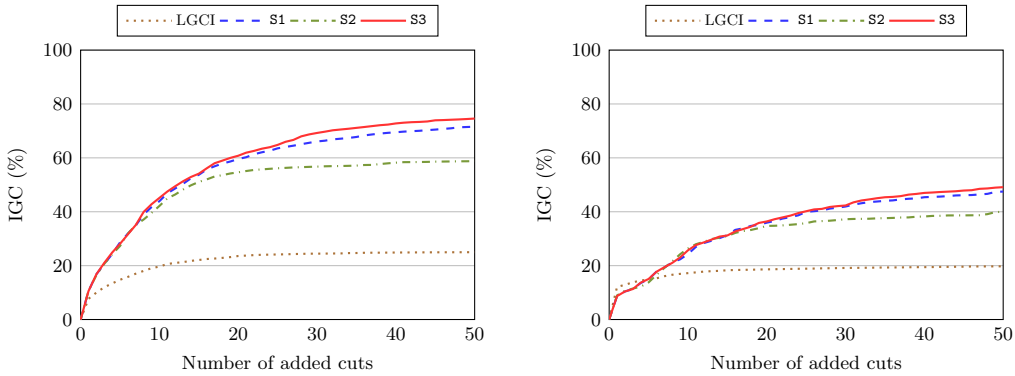
(a) $n = 2000$ (b) $n = 2000$

Figure 2.4: Cutting plane time by each separation algorithm for $n = 2000, 5000$

appear that the HCG tightened the LP relaxations using many cuts with additional computation time. However, the following experimental results show that the CG cuts are more effective in improving the integrality gap than general lifted GUB inequalities, even though the same number of cuts are used.

We investigated the integrality gap closed, varying the number of added cuts during each cutting plane algorithm. For all instances of $n = 2000, 5000$ and $\bar{a} = 200$, Figure 2.5 represents the average integrality gap closed after adding the generated cuts. `S1`, `S2`, and `S3` denote the HCG using variable ordering strategies 1, 2, and 3, respectively.

The results showed that each generated CG cut was more effective at improving the integrality gap than each general lifted GUB cover inequality, regardless of the choice of variable ordering strategies. Specifically, when $n = 5000$, the general lifted GUB cover inequality generated at the beginning of the cutting plane algorithm was initially more effective than the CG cut. However, this difference was reversed with only a few additional CG cuts. These results imply that the gap-closing effect of



(a) $n = 2000$ (b) $n = 5000$

Figure 2.5: Change in integrality gap closed depending on the number of added cuts

the HCG remains promising even when the number of generated CG cuts is limited. Therefore, using only CG cuts generated by the heuristic separation algorithm can enhance the formulations more effectively and efficiently than general lifted GUB cover inequalities by appropriately limiting the number of added cuts.

2.6.3 Effectiveness of CG cuts for GKPs on benchmark instances of binary integer linear programs

For each benchmark instance of binary integer linear programs, we applied the cutting plane algorithm using the heuristic CG separation algorithm with variable ordering strategy 3 (`S3`), which is the most effective strategy to enhance the formulation. Then, it was compared to the cutting plane algorithm using the default algorithm that generates general lifted GUB cover inequalities. We set the parameter for the heuristic separation algorithm as $T = \min\{r^\star, m\}$, and the time limit was 60 seconds for the cutting plane algorithms.

The results on MMKP instances are provided in Table 2.5. In contrast to previous

experiments, we approximately compute the integrality gap closed as

$$\text{IGC } (\%) = \frac{z_{LP} - z_{CUT}}{z_{LP} - z_B} * 100,$$

using best solutions (z_B) given in Table A.1 because optimal solutions for several instances are unknown. However, since the obtained best solutions were almost optimal (Gap$<$ 0.02% in Table A.1), the approximate integrality gap may be close to the exact one. "Time (s)" denote the cutting plane times. Since each instance has $r^\star$ different GKP polytopes as its substructure, the cutting plane algorithm generates at most $r^\star$ cuts for each iteration.

Table 2.5: Results of cutting plane algorithms for MMKP instances

Name	LGCI			HCG (S3)		
	IGC (%)	#Cut	Time (s)	IGC (%)	#Cut	Time (s)
I01	42.77	5	0.001	52.65	12	0.001
I02	36.59	5	0.001	100.00	12	0.003
I03	0.75	5	0.002	6.02	23	0.013
I04	0.74	3	0.002	4.42	22	0.017
I05	100.00	3	0.004	100.00	3	0.005
I06	36.64	11	0.022	64.13	14	0.019
I07	0.03	3	0.181	1.02	21	0.07
I08	0.00	1	0.235	0.67	15	0.066
I09	0.00	0	0.008	0.04	17	0.09
I10	0.01	2	2.245	0.22	25	0.15
I11	0.03	4	7.388	0.13	22	0.218
I12	0.28	4	11.506	0.64	17	0.207
I13	0.11	4	16.288	0.33	19	0.235
Average	16.76	3.85	2.91	25.41	17.08	0.08

Similar to the results obtained for GKP instances, the HCG further reduced the integrality gap compared to the LGCI for MMKP instances. Moreover, the cutting plane times for the LGCI and HCG were comparable in instances I01 to I06, de-

spite generating more CG cuts than general lifted cover inequalities. However, for instances with many GUB sets, such as I07 to I13, the LGCI required more cutting plane time than the HCG, even though its gap-closing effect was weaker, because the longer computation time was required to generate each general lifted GUB cover inequality compared to the CG cuts, as observed in the results for 0-1 GKP instances. However, for the difficult instances I07 to I13, the cutting plane algorithm was ineffective in reducing the integrality gap, while it showed promise for the relatively easier instances I01 to I06.

Table 2.6: Results of cutting plane algorithms for MIPLIB instances

Name	LGCI			HCG (`S3`)		
	IGC (%)	#Cut	Time (s)	IGC (%)	#Cut	Time (s)
bm23	16.18	6	0.003	20.23	26	0.024
l152lav	0.00	0	0.295	1.02	4	0.914
lp4l	2.04	2	0.299	3.40	3	0.192
lseu	62.53	17	0.021	75.65	49	0.249
mitre	9.07	258	60.000	12.56	426	60.000
mod008	14.66	17	2.027	83.14	75	5.413
mod010	18.32	3	2.290	18.32	5	1.175
p0033	72.72	23	0.008	87.42	26	0.009
p0040	100.00	2	0.002	100.00	3	0.003
p0201	33.78	14	0.048	33.78	30	0.083
p0282	96.59	164	5.238	97.60	208	4.193
p0291	96.49	45	4.662	98.03	67	33.462
p0548	86.90	244	9.636	87.81	310	19.103
p2756	0.88	86	60.000	0.95	245	60.000
pipex	74.21	22	0.014	80.11	50	0.103
sentoy	13.34	26	0.031	24.79	63	1.931
sp97ar	0.00	24	60.000	3.47	122	16.779
Average	41.04	56.1	12.034	48.72	100.7	11.978

The results on MIPLIB instances are provided in Table 2.6. The HCG could also provide more enhanced formulations compared to the LGCI. In detail, for the instance `l152lav`, the HCG could enhance the formulation while the LGCI failed to

generate any general lifted GUB cover inequalities. In `sp97ar`, the LGCI could generate several cuts. However, the cuts were ineffective in enhancing the formulation, while the CG cuts generated by the HCG could improve the integrality gap. Overall, the HCG generated more cuts than the LGCI. Hence, the HCG required more cutting plane time compared to the LGCI. Nonetheless, using CG cuts could further enhance the formulations in a reasonable time. These results show the potential of CG cuts to improve the LP relaxation of general binary integer linear programs.

2.7 Conclusion

In this study, we present the properties of non-dominated CG cuts associated with the knapsack problem with generalized upper bounds (GKP). We propose an exact and heuristic separation algorithm for CG cuts and evaluate their performance through computational tests on the GKP and the multi-dimensional multiple-choice knapsack problem.

Although the CG cut separation problem for general integer linear programs is known to be strongly NP-hard, we demonstrate that the separation problem for the GKP can be solved in pseudo-polynomial time through the proposed exact separation algorithm. Computational results show that the proposed heuristic can improve the comparable integrality gap with the exact separation algorithm within significantly less time. Furthermore, generated CG cuts outperform general lifted GUB cover inequalities in strengthening the linear programming relaxations within a comparable time. This result implies that CG cuts associated with GKP can be used in practice as a pre-process to enhance the formulation of various problems.

In the computational tests, our heuristic separation algorithm shows promise

even though we selected the parameter arbitrarily. Further research involving a more careful parameter selection may improve the algorithm's performance. Additionally, investigations into the characteristics of effective CG cuts for the GKP could enhance separation algorithms and identify a new family of strong valid inequalities for the GKP. Finally, research on CG cuts associated with specific combinatorial problems is lacking. This line of study may provide a useful clue to tackle such problems.

Chapter 3

Strengthening Chvátal-Gomory cuts for binary integer linear programs and its extension to generalized upper bounds

This chapter presents a novel method to strengthen CG cuts for binary integer linear programs. We first establish the relationship between CG cuts for binary integer linear programs and lifted cover inequalities for binary knapsack problems. Based on this result, we propose a method for strengthening a given CG cut, utilizing a lifting function of cover inequalities for binary knapsack problems. We then show that the strengthened cut can dominate the other cuts derived from the given CG cut through existing strengthening methods. We also present the extension of the method to CG cuts for binary integer linear programs with generalized upper bounds. Computational test results indicate that the strengthened cuts obtained from the proposed methods can yield more enhanced formulations defined with fewer cuts compared to CG cuts.

3.1 Introduction

In the previous chapter, we demonstrated that rank-1 CG cuts derived from the GKP effectively enhance the formulations of binary integer linear programs. We refer to rank-1 CG cuts as CG cuts for brevity. However, the resulting formulations often require many cuts, leading to longer computational times to solve the relaxations. This large-sized formulation can deteriorate the performance of the branch-and-bound algorithm, even though tighter upper bounds can be obtained from the formulations.

Cuts that dominate the CG cuts can further enhance the formulations, while the same level of enhancement with the CG cuts can be achieved using fewer cuts. The impact of stronger cuts was observed in the experiment results using general lifted GUB cover inequalities (LGCIs) in Section 2.6. Figure 3.1 illustrates changes in the integrality gap closed by varying the number of cuts added during cutting plane algorithms using CG cuts and LGCIs for a GKP instance with $n = 5000$ and $m = 20$.

As discussed in Section 2.3, some LGCIs can dominate the CG cuts because they can have a Chvátal rank higher than 1. Therefore, as shown in Figure 3.1, fewer LGCIs may be necessary than the CG cuts until they both achieve the same enhancement level. However, the resulting formulations obtained by using LGCIs may not provide relaxations as tight as those obtained by the CG cuts because the generation method cannot always yield LGCIs that dominate the CG cuts. Therefore, to reduce the size of the enhanced formulation while preserving the tightness of the relaxation provided by CG cuts for the GKP, only cuts that dominate the CG cuts should be considered.

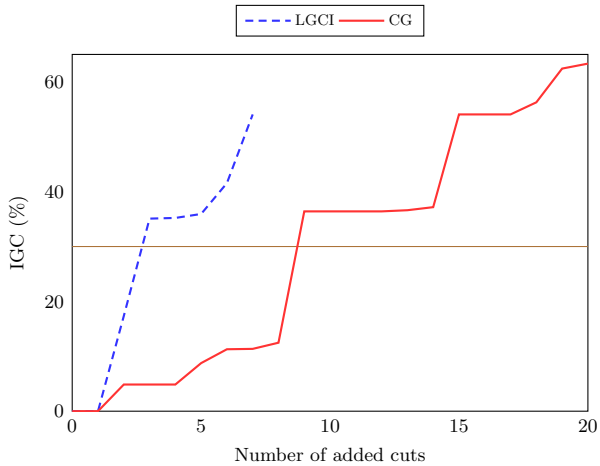


Figure 3.1: Changes in IGC (%) depending on the number of added cuts

One possible strategy to obtain such cuts is to strengthen the generated CG cuts using cut-strengthening methods. In general, for a given cut for a problem, a strengthened cut can be obtained by solving the separation problem defined with the family of cuts that dominate the given one. Cut strengthening methods refer to heuristic solution approaches for the separation problem, emphasizing computational efficiency.

Such strengthening methods for CG cuts have typically been studied for general integer linear programs. Although there has been only one study that directly refers to CG cut strengthening methods (Letchford & Lodi, 2002), numerous studies on cut-generating functions (Conforti, Cornuéjols, et al., 2015) can be utilized as CG cut strengthening methods. Recall that, for a single-constraint relaxation for integer linear programs, the CG cut is derived from the constraint using the cut-generating function π_{CG}. If another cut-generating function that dominates π_{CG} is applied to the constraint, one can obtain a stronger cut, such as the Gomory mixed-integer cut Gomory (1960). These methods can also be applied to binary integer linear programs

such as the GKP.

Let us consider the single-constraint relaxation $\mathcal{X}_S$ of a binary integer linear program, defined in Section 1.2.4 as follows.

$$\mathcal{X}_S = \{x \in \{0,1\} : u^T Ax \leq u^T \mathbf{b}\}$$

Then, a CG cut derived from the above relaxation is

$$\lfloor u^T A \rfloor x \leq \lfloor u^T \mathbf{b} \rfloor.$$

The CG cut can be strengthened to a CGF cut

$$\sum_{j \in N} (u^T A_j - \pi(u^T A_j)) x_j \leq \lfloor u^T \mathbf{b} \rfloor,$$

with a cut-generating function π that dominates π_{CG}. However, as we discussed in Section 1.2.4, both the CG and CGF cuts are valid for the further relaxed single-constraint relaxation $\mathcal{X}_{SR}$ where the variable bounds are ignored because the binarity of decision variables is not used to derive them. Therefore, for binary integer linear programs, stronger cuts may be derived by incorporating the binarity of decision variables into their derivation.

In this study, we propose a novel method to strengthen CG cuts for general binary integer linear programs by exploiting the variable bounds. Specifically, based on the observation that the single-constraint relaxations are represented as the binary knapsack polytopes, our CG cut strengthening method utilizes lifting techniques of cover inequalities. We then compare the strengths of obtained cuts with those

derived by cut-generating functions. The proposed method is extended to binary integer linear programs with generalized upper bounds, which can be applied to CG cuts for the GKP.

The remainder of this chapter is organized as follows. We review the relevant literature in Section 3.2. In Section 3.3, we characterize non-dominated CG cuts for binary integer linear programs. Then, we propose a CG cut strengthening method along with a theoretical comparison of its strength with others in Section 3.4. We extend the results to binary integer linear programs with generalized upper bounds in Section 3.5. In Section 3.6, we present the computational test results. Concluding remarks for this chapter are given in Section 3.7.

3.2 Related works

Several studies have incorporated variable bounds to derive strong valid inequalities from single-constraint relaxations of general mixed-integer linear programs with variable bounds.

Marchand & Wolsey (2001) induced mixed-integer rounding cuts from the single-constraint relaxations after complementing some of the integer variables. That is, they replaced some of the integral decision variables x_j satisfying $0 \leq x_j \leq d_j$ for some $d_j \in \mathbb{Z}+$ with $\bar{x}_j = d_j - x_j$ by appropriately adjusting the coefficients of the constraint. The authors showed that the obtained cuts, c-MIR inequalities, were more effective in solving mixed-integer linear programs than mixed-integer rounding cuts.

Atamtürk & Günlük (2010) presented a reformulation of the single-constraint relaxations using the relationship between the coefficients of the variables and the

right-hand side, together with complementing some of the integer variables. From the constraint of the relaxation, the authors derived the so-called mingling inequalities using the mixed-integer rounding procedure, which yields a mixed-integer rounding cut from an inequality. The authors showed that mingling inequalities dominate c-MIR inequalities when the complemented variables define a cover of the mixed-integer knapsack polytope representing the single-constraint relaxation. Atamtürk & Kianfar (2012) further generalized mingling inequalities into n-step mingling inequalities by applying mixed-integer rounding procedures iteratively and derived a new family of facet-defining inequalities for mixed-integer knapsack polytopes.

Those studies focused on deriving a useful cut from the given mixed-integer rounding cut by incorporating variable bounds. However, their methods did not strengthen the original mixed-integer rounding cuts. Instead, they derived mixed-integer rounding cuts from other single-constraint relaxations reformulated from the original relaxation. While we focus on CG cuts for binary integer linear programs, our method ensures that the obtained cuts are at least as strong as the given CG cuts. Moreover, our method exploits the properties of non-dominated CG cuts and uses lifting techniques for cover inequalities rather than CG procedures or mixed-integer rounding procedures.

Alternatively, the sequential lifting technique proposed by Zemel (1978) can be used to strengthen the given CG cut for binary integer linear programs. For example, for the given CG cut derived from $\mathcal{X}_S$, which can be represented as a binary knapsack polytope, the coefficients of CG cuts with zero value can be improved by solving the lifting problem for $\mathcal{X}_S$, which is also represented as a binary knapsack problem. However, the lifting problem may be challenging to solve because the coefficients of

CG cuts are not polynomial to the input size in general.

Dey & Richard (2009) proposed a heuristic for the sequential lifting technique that solves the lifting problems based on its LP relaxation. However, their heuristic can be too costly to use as a strengthening method for a given CG cut because a considerable amount of computation may already be used to obtain the CG cut. Instead, we adopt the simultaneous lifting technique for cover inequalities in our CG cut strengthening method, where additional computational efforts are minimal.

3.3 Non-dominated CG cuts for the single-constraint relaxation of binary integer linear programs

In this section, we present properties of non-dominated CG cuts for binary integer linear programs, which are the cornerstone of our CG cut strengthening method.

Let us consider a CG cut for a binary integer linear program, described as

$$\lfloor u^T A \rfloor x \leq \lfloor u^T \mathbf{b} \rfloor, \tag{3.1}$$

and the corresponding single-constraint relaxation $\mathcal{X}_S$ again, which can be rewritten as follows.

$$\mathcal{X}_S = \left\{ x \in \{0,1\}^n : \sum_{j \in N} u^T A_j x_j \leq u^T \mathbf{b} \right\}$$

Recall that the system $Ax \leq \mathbf{b}$ includes the variable bounds, $x_j \leq 1$ for each $j \in N$. Without loss of generality, let u_j be the multipliers corresponding to the jth variable bound for each $j \in N$. Then, we can define another single-constraint relaxation, $\mathcal{X}'_S$,

which is described as follows.

$$\mathcal{X}_{SB} = \left\{ x \in \{0,1\} : \sum_{j \in N} (u^T A_j - u_j) x_j \leq u^T \mathbf{b} - \sum_{j \in N} u_j \right\}$$

It is clear that $\mathcal{X}_{SB} \subseteq \mathcal{X}_S$ because the constraint in $\mathcal{X}_S$ is dominated by the constraint in $\mathcal{X}_{SB}$. It can be easily shown that the CG cut (3.1) can be derived from $\mathcal{X}_{SB}$.

Let $N' = \{j \in N : u^T A_j - u_j < 0\}$. By complementing variables in N', $\mathrm{conv}(\mathcal{X}'_S)$ can be reformulated as a binary knapsack polytope $\mathrm{conv}(\mathcal{X}_B)$ where

$$\mathcal{X}_B = \left\{ y \in \{0,1\}^n : \sum_{j \in N} |u^T A_j - u_j| y_j \leq u^T \left(\mathbf{b} - \sum_{j \in N'} A_j \right) - \sum_{j \in N \setminus N'} u_j \right\},$$

where $y_j = 1 - x_j$ if $j \in N'$, and $y_j = x_j$, otherwise. Then, the CG cut (3.1) can be rewritten with y as the following valid inequality for $\mathcal{X}_B$.

$$\sum_{j \in N \setminus N'} \lfloor u^T A_j \rfloor y_j - \sum_{j \in N'} \lfloor u^T A_j \rfloor y_j \leq \lfloor u^T \mathbf{b} \rfloor - \sum_{j \in N'} \lfloor u^T A_j \rfloor, \tag{3.2}$$

Proposition 3.1. *The inequality* (3.2) *is a CG cut for* $\mathcal{X}_B$.

Proof. Let us consider a CG cut for $\mathcal{X}_B$ in the following form.

$$\sum_{j \in N} \left\lfloor |u^T A_j - u_j| + \theta_j - v_j \right\rfloor y_j \leq \left\lfloor u^T \left(\mathbf{b} - \sum_{j \in N'} A_j \right) - \sum_{j \in N \setminus N'} u_j + \sum_{j \in N} \theta_j \right\rfloor$$

Here, θ_j is the multiplier for the jth variable bound, $y_j \leq 1$, and v_j corresponds to $-y_j \leq 0$ for each $j \in N$. Let $\hat{\theta}_j = u_j$ for each $j \in N \setminus N'$ while $\hat{\theta}_j = u^T A_j - \lfloor u^T A_j \rfloor$ for each $j \in N'$. Additionally, we define $\hat{v}_j$ as u_j for each $j \in N'$ and 0, otherwise.

Then, the CG cut of the above form with $(\hat{\theta}, \hat{v})$ can be written as

$$\sum_{j \in N \setminus N'} \lfloor u^T A_j \rfloor y_j - \sum_{j \in N'} \lfloor u^T A_j \rfloor y_j \leq \left\lfloor u^T \mathbf{b} - \sum_{j \in N'} \lfloor u^T A_j \rfloor \right\rfloor, \tag{3.3}$$

where

$$\left\lfloor u^T \mathbf{b} - \sum_{j \in N'} \lfloor u^T A_j \rfloor \right\rfloor = \lfloor u^T \mathbf{b} \rfloor - \sum_{j \in N'} \lfloor u^T A_j \rfloor.$$

Therefore, the CG cut (3.3) is equivalent to the inequality (3.2), and the result follows. □

Proposition 3.1 states that the CG cut for $\mathcal{X}_{SB}$ can be derived from a CG cut for $\mathcal{X}_B$. Furthermore, it can be easily shown that the dominance between two valid inequalities for $\mathcal{X}_{SB}$ is preserved in the corresponding valid inequalities for $\mathcal{X}_B$, and the converse also holds. These results imply that a cut for $\mathcal{X}_B$ that dominates the CG cut (3.2) can yield a cut for $\mathcal{X}_{SB}$ that dominates the given CG cut (3.1). Therefore, we only focus on CG cuts for the binary knapsack polytope $\mathcal{X}_B$ and their strengthening method in the subsequent discussion.

3.3.1 Non-dominated CG cuts for binary knapsack polytopes

For the sake of simplicity, we redefine $\mathcal{X}_B$ as

$$\mathcal{X}_B = \left\{ x \in \{0,1\} : \sum_{j \in N} a_j x_j \leq a_0 \right\},$$

where $a_j \in \mathbb{R}_+$ for each $j \in N \cup \{0\}$. One caution for the reader is that, in contrast to Chapter 2, a_j's need not be integers. Then, we can define a CG cut for $\mathcal{X}_B$ as

$$\sum_{j \in N} \lfloor \hat{\theta}_0 a_j + \hat{\theta}_j \rfloor x_j \leq \left\lfloor \hat{\theta}_0 a_0 + \sum_{j \in N} \hat{\theta}_j \right\rfloor, \tag{3.4}$$

for some $(\hat{\theta}_0, \hat{\theta}) \in \mathbb{R}_+^{n+1}$. We first characterize non-dominated CG cuts for $\mathcal{X}_B$.

Proposition 3.2. *If the CG cut* (3.4) *is non-dominated, then* $\hat{\theta}_j < 1$ *for each* $j \in N$ *and* $\frac{1}{2} \leq f_0 < 1$ *where* $f_0 = \hat{\theta}_0 a_0 + \sum_{j \in N} \hat{\theta}_j - \lfloor \hat{\theta}_0 a_0 + \sum_{j \in N} \hat{\theta}_j \rfloor$.

Proof. Suppose that there exists $k \in N$ such that $\hat{\theta}_k \geq 1$. We define $\theta_0^\star = \hat{\theta}_0$ and $\theta_j^\star = \hat{\theta}_j - \lfloor \hat{\theta}_j \rfloor$ for each $j \in N$. Then, the CG cut (3.4) can be represented as a linear combination of the CG cut defined with $(\theta_0^\star, \theta^\star)$ and variable bounds with non-negative multipliers. Therefore, the CG cut (3.4) is dominated by the CG cut defined with $(\theta_0^\star, \theta^\star)$, which contradicts the assumption that (3.4) is non-dominated.

Now, suppose that $f_0 = 0$. Then, the CG cut (3.4) is clearly dominated by

$$\sum_{j \in N} (\hat{\theta}_0 a_j + \hat{\theta}_j) x_j \leq \hat{\theta}_0 a_0 + \sum_{j \in N} \hat{\theta}_j.$$

If $0 < f_0 < \frac{1}{2}$, let us define $(\theta_0^\star, \theta^\star) = (t\hat{\theta}_0, t\hat{\theta})$ where

$$t = \left\lceil \frac{1}{2 f_0} \right\rceil.$$

Then, a CG cut defined with $(\theta_0^\star, \theta^\star)$ is

$$\sum_{j \in N} \lfloor t(\hat{\theta}_0 a_j + \hat{\theta}_j) \rfloor x_j \leq \left\lfloor t(\hat{\theta}_0 a_0 + \sum_{j \in N} \hat{\theta}_j) \right\rfloor. \tag{3.5}$$

On the other hand, the CG cut (3.4) can be rewritten by scaling the coefficients as follows.

$$\sum_{j\in N} t\lfloor \hat{\theta}_0 a_j + \hat{\theta}_j \rfloor x_j \leq t\Big\lfloor \hat{\theta}_0 a_0 + \sum_{j\in N} \hat{\theta}_j \Big\rfloor \tag{3.6}$$

By the superadditivity of the round-down function, it is clear that

$$t\lfloor \hat{\theta}_0 a_j + \hat{\theta}_j \rfloor \leq \lfloor t(\hat{\theta}_0 a_j + \hat{\theta}_j) \rfloor,\ \forall j \in N. \tag{3.7}$$

Additionally, the right-hand side of (3.5) is equivalent to

$$\left\lfloor t\left(\hat{\theta}_0 a_0 + \sum_{j\in N} \hat{\theta}_j\right)\right\rfloor = t\Big\lfloor \hat{\theta}_0 a_0 + \sum_{j\in N} \hat{\theta}_j \Big\rfloor + \lfloor t f_0 \rfloor. \tag{3.8}$$

Because $\frac{1}{2} \leq t f_0 < f_0 + \frac{1}{2}$ and $f_0 < \frac{1}{2}$, $\lfloor t f_0 \rfloor = 0$. Therefore, the right-hand sides of (3.5) and (3.6) are equivalent. Therefore, the CG cut (3.5) dominates the CG cut (3.4), which contradicts the assumption that (3.4) is non-dominated. Therefore, the result follows. □

Now, let $\alpha^T x \leq \alpha_0$ be the CG cut (3.4), that is, $\alpha_0 = \lfloor \hat{\theta}_0 a_0 + \sum_{j\in N} \hat{\theta}_j \rfloor$ and $\alpha_j = \lfloor \hat{\theta}_0 a_j + \hat{\theta}_j \rfloor$ for each $j \in N$. Then, we can define a linear program

$$\max\{\alpha^T x : x \in \mathcal{P}_B\}, \tag{3.9}$$

where $\mathcal{P}_B$ is the linear relaxation of $\mathcal{X}_B$, defined as follows.

$$\mathcal{P}_B = \left\{ x \in [0,1] : \sum_{j\in N} a_j x_j \leq a_0 \right\}$$

An optimal solution $x^\star$ of the above linear program can be obtained as follows. Let $(j_1, ..., j_n)$ be the sequence of variables sorted in descending order of α_j/a_j. Additionally, let $l = \min\{k \in N : \sum_{i=1}^{k} : a_{j_i} > a_0\}$ and $C_B = \{j_1, ..., j_l\}$. We note that C_B is a cover for $\mathcal{X}_B$ by definition. Then, $x^\star$ is defined as

$$x_j^\star = \begin{cases} 1 & , \ j \in C_B \setminus \{l\} \\ (b - \sum_{j \in C_B \setminus \{l\}} a_j)/a_l & , \ j = l \\ 0 & , \ j \in N \setminus C_B \end{cases}.$$

Proposition 3.3. *If the CG cut $\alpha^T x \leq \alpha_0$ is non-dominated, then there exists $(\theta_0^\star, \theta^\star) \in \mathbb{R}_+^{n+1}$, which yields $\alpha^T x \leq \alpha_0$, such that*

1. $\theta_0^\star = \alpha_l/a_l$
2. $\theta_j^\star = \lceil \theta_0^\star a_j \rceil - \theta_0^\star a_j$ *for each* $j \in C_B$ *and* $\theta_j^\star = 0$ *for each* $j \in N \setminus C_B$
3. $\theta_j^\star \leq f_0^\star$ *for each* $j \in C_B$ *such that* $\lceil \theta_0^\star a_j \rceil = 1$

where $f_0^\star = \theta_0^\star a_0 + \sum_{j \in N} \theta_j^\star - \lfloor \theta_0^\star a_0 + \sum_{j \in N} \theta_j^\star \rfloor$.

Proof. Let us consider the dual problem of the linear program (3.9), which is described as follows.

$$\begin{aligned} \min \quad & \theta_0 a_0 + \sum_{j \in N} \theta_j \\ \text{s.t} \quad & \alpha_j \leq \theta_0 a_j + \theta_j, \ j \in N \\ & (\theta_0, \theta) \in \mathbb{R}_+^{n+1} \end{aligned}$$

By the definition of α, it is clear that $(\hat{\theta}_0, \hat{\theta})$ is a feasible solution for the above

problem. Let $(\theta_0^\star, \theta^\star)$ be an optimal solution to the above problem, defined as $\theta_0^\star = \alpha_l/a_l$ and

$$\theta_j^\star = \begin{cases} \alpha_j - \theta_0^\star a_j & , \ j \in C_B \\ 0 & , \ j \in N \setminus C_B \end{cases}.$$

Let $\sum_{j\in N} \alpha_j^\star x_j \le \alpha_0^\star$ be the CG cut defined with $(\theta_0^\star, \theta^\star)$. Since $(\theta_0^\star, \theta^\star)$ is optimal for the above problem, it is clear that $\alpha_j \le \alpha_j^\star$ for each $j \in N$ and $\alpha_0 \ge \alpha_0^\star$. If there exists $k \in N$ such that $\alpha_k < \alpha_k^\star$, or $\alpha_0 > \alpha_0^\star$, then $\sum_{j\in N} \alpha_j^\star x_j \le \alpha_0^\star$ dominates $\sum_{j\in N} \alpha_j x_j \le \alpha_0$, which conflicts to the assumption that $\sum_{j\in N} \alpha_j x_j \le \alpha_0$ is non-dominated. Therefore, $\alpha_j = \alpha_j^\star$ for each $j \in N$ and $\alpha_0 = \alpha_0^\star$.

By the definition of $(\theta_0^\star, \theta^\star)$, $\alpha_j = \theta_0^\star a_j + \theta_j^\star$ for each $j \in C_B$. On the other hand, if there exists $k \in C_B$ such that $\theta_k^\star \ge 1$, the CG cut $\sum_{j\in N} \alpha_j x_j \le \alpha_0$ is a dominated one by Proposition 3.2. Therefore, for each $j \in C_B$, $\theta_j^\star$ should be less than 1, which implies that $\theta_0^\star a_j \le \alpha_j < \theta_0^\star a_j + 1$. Since α_j's are integers, $\alpha_j = \lceil \theta_0^\star a_j \rceil$, that is, $\theta_j^\star = \lceil \theta_0^\star a_j \rceil - \theta_0^\star a_j$ for each $j \in C_B$.

Now, suppose that there exists $k \in C_B$ such that $\alpha_k = 1$ and $\theta_k^\star > f_0^\star$. Then,

$$\lfloor \alpha_0 + f_0^\star - \theta_k^\star \rfloor = \alpha_0 - 1.$$

If $\theta_k^\star$ is replaced with 0, the CG cut defined with $(\theta_0^\star, \theta^\star)$ can be described as follows.

$$\sum_{j\in N\setminus\{k\}} \alpha_j x_j \le \alpha_0 - 1$$

The CG cut $\alpha^T x \le \alpha_0$ is dominated by the above CG cut because it is a non-negative linear combination of the above CG cut and $x_k \le 1$. This result contradicts the assumption that $\alpha^T x \le \alpha_0$ is non-dominated. Therefore, the result follows. $\square$

The above propositions state that a non-dominated CG cut $\alpha^T x \leq \alpha_0$ can be rewritten as

$$\sum_{j \in C_B} \lceil \theta_0^\star a_j \rceil x_j + \sum_{j \in N \setminus C_B} \lfloor \theta_0^\star a_j \rfloor \leq \left\lfloor \theta_0^\star a_0 + \sum_{j \in C_B} \theta_j^\star \right\rfloor, \tag{3.10}$$

with some $\theta_0^\star \in \mathbb{R}_+$ and a cover $C_B \subseteq N$ such that $\sum_{j \in C_B} a_j > a_0$, where $\theta_j^\star = \lceil \theta_0^\star a_j \rceil - \theta_0^\star a_j$ for each $j \in C_B$ and $\theta_j^\star = 0$ for each $j \in N \setminus C_B$.

3.3.2 Maximal CG cuts for binary knapsack polytopes

A CG cut of the form (3.10) may be dominated because the above propositions are necessary conditions for non-dominated CG cuts. Nonetheless, the CG cuts of the form (3.10) are sufficient to describe the Chvátal closure of $\mathcal{P}_B$. We call such CG cuts as *maximal* CG cuts in perspective that no more CG cuts are necessary to describe the Chvátal closure.

Definition 3.1. *The inequality of the form* (3.10) *is called a maximal CG cut for* $\mathcal{X}_B$ *if*

1. $C_B \subseteq N$ *is a cover for* $\mathcal{X}_B$*, i.e.,* $\sum_{j \in C_B} a_j > a_0$.
2. $\theta_0^\star = p/a_l$ *for some* $p \in \mathbb{Z}_+$ *with* $l \in C_B$*, and* $\theta_j^\star = \lceil \theta_0^\star a_j \rceil - \theta_0^\star a_j$ *for each* $j \in C_B$ *while* $\theta_j^\star = 0$ *for each* $j \in N \setminus C_B$.
3. $\frac{1}{2} \leq f_0^\star$ *and* $\theta_j^\star \leq f_0^\star$ *if* $\lceil \theta_0^\star a_j \rceil = 1$ *for each* $j \in C_B$.

where $f_0^\star = \theta_0^\star a_0 + \sum_{j \in C_B} \theta_j^\star - \lfloor \theta_0^\star a_0 + \sum_{j \in C_B} \theta_j^\star \rfloor$.

In the following discussion, we show that, from any given CG cut, a maximal CG cut can be easily obtained, which dominates the given one. Let a CG cut $\alpha^T x \leq \alpha_0$ be given. By solving the linear program (3.9), an optimal solution $x^\star$, C_B and l can be

obtained in $O(n \log n)$ computation time, such that $\sum_{j \in C_B} a_j > a_0$ and $0 \leq x_l^\star < 1$. Then, we define $(\bar{\theta}_0, \bar{\theta})$ as $\bar{\theta} = \alpha_l / a_l$ and

$$\bar{\theta}_j = \begin{cases} \lceil \bar{\theta}_0 a_j \rceil - \bar{\theta}_0 a_j & , \ j \in C_B \\ 0 & , \ j \in N \setminus C_B \end{cases}.$$

Additionally, let $\sum_{j \in N} \bar{\alpha}_j x_j \leq \bar{\alpha}_0$ be the CG cut defined with $(\bar{\theta}_0, \bar{\theta})$. We note that the CG cut can be rewritten as

$$\sum_{j \in C_B} (\bar{\theta}_0 a_j + \bar{\theta}_j) x_j + \sum_{j \in N \setminus C_B} \lfloor \bar{\theta}_0 a_j \rfloor x_j \leq \left\lfloor \bar{\theta}_0 a_0 + \sum_{j \in C_B} \bar{\theta}_j \right\rfloor,$$

by the definition of $(\bar{\theta}_0, \bar{\theta})$. Then, as shown in the proof of Proposition 3.3, $\sum_{j \in N} \bar{\alpha}_j x_j \leq \bar{\alpha}_0$ is at least as strong as the given CG cut. Furthermore, the CG cut satisfies the first and second conditions in Definition 3.1.

Now, let $\bar{f}_0 = \bar{\theta}_0 a_0 + \sum_{j \in C_B} \bar{\theta}_j - \lfloor \bar{\theta}_0 a_0 + \sum_{j \in C_B} \hat{\theta}_j \rfloor$. We only consider $\bar{f}_0 > 0$ because, if $\bar{f}_0 = 0$, the CG cut is redundant for $\mathcal{P}_B$ as shown in the proof of Proposition 3.2. Let us define $(\theta_0^\star, \theta^\star)$ as $\theta_0^\star = \bar{\theta}_0$ and $\theta_j^\star = \bar{\theta}_j$ for each $j \in N$ if $\frac{1}{2} \leq \bar{f}_0$. If $0 < \bar{f}_0 < \frac{1}{2}$, we define $(\theta_0^\star, \theta^\star)$ as $\theta_0^\star = t\bar{\theta}_0$ and $\theta_j^\star = t\bar{\theta}_j - \lfloor t\bar{\theta}_j \rfloor$ for each $j \in N$ where

$$t = \left\lceil \frac{1}{2\bar{f}_0} \right\rceil.$$

Subsequently, let $\sum_{j \in N} \alpha_j^\star x_j \leq \alpha_0^\star$ be the CG cut defined with $(\theta_0^\star, \theta^\star)$. Then, $\alpha_j^\star = \lceil \theta_0^\star a_j \rceil$ for each $j \in C_B$ and $\alpha_j^\star = \lfloor \theta_0^\star a_j \rfloor$ for each $j \in N \setminus C_B$. Additionally, $\alpha_0^\star = \lfloor \theta_0^\star a_0 + \sum_{j \in C_B} \theta_j^\star \rfloor$. The CG cut $\sum_{j \in N} \alpha_j^\star x_j \leq \alpha_0^\star$ is at least as strong as $\sum_{j \in N} \bar{\alpha}_j x_j \leq \bar{\alpha}_0$ by the proof of Proposition 3.2 while it still satisfies the first and second conditions in Definition 3.1.

Now, let $f_0^\star = \theta_0^\star a_0 + \sum_{j\in C_B}\theta_j^\star - \lfloor \theta_0^\star a_0 + \sum_{j\in C_B}\theta_j^\star \rfloor$. Suppose that there exists $k \in C_B$ such that $\alpha_k^\star = 1$ and $\theta_k^\star > f_0^\star$.

Proposition 3.4. *$C_B \setminus \{k\}$ is a cover for $\mathcal{X}_B$.*

Proof. We show that $\theta_0^\star(\sum_{j\in C_B\setminus\{k\}} a_j - a_0) > 0$ to prove that $C_B\setminus\{k\}$ is a cover. By the definition of $(\theta_0^\star, \theta^\star)$, $\theta_0^\star a_j = \alpha_j^\star - \theta_j^\star$ for each $j \in C_B$. Let $\lambda_B = \sum_{j\in C_B} a_j - a_0$. Then, $\lambda_B > 0$ because C_B is a cover. From the definition of $f_0^\star$,

$$\begin{aligned}
\theta_0^\star\left(\sum_{j\in C_B\setminus\{k\}} a_j - a_0\right) &= \sum_{j\in C_B\setminus\{k\}}(\alpha_j^\star - \theta_j^\star) - f_0^\star - \left\lfloor \theta_0^\star a_0 + \sum_{j\in C_B}\theta_j^\star \right\rfloor + \sum_{j\in C_B}\theta_j^\star \\
&= \sum_{j\in C_B\setminus\{k\}} \alpha_j^\star + \theta_k^\star - f_0^\star - \left\lfloor \theta_0^\star a_0 + \sum_{j\in C_B}\theta_j^\star \right\rfloor \\
&= \theta_k^\star - f_0^\star - \alpha_k - \lfloor -\theta_0^\star \lambda_B \rfloor,
\end{aligned}$$

where the last equality holds due to $a_0 = \sum_{j\in C_B} a_j - \lambda_B$. Because $\lfloor -\theta_0^\star\lambda_B\rfloor = -\lceil \theta_0^\star\lambda_B\rceil$ and $\alpha_k = 1$ by the assumption,

$$\theta_k^\star - f_0^\star - \alpha_k - \lfloor -\theta_0^\star\lambda_B\rfloor = \theta_k^\star - f_0^\star - 1 + \lceil\theta_0^\star\lambda_B\rceil > 0,$$

where the last inequality holds because $\lceil\theta_0^\star\lambda_B\rceil \geq 1$ and $\theta_k^\star - f_0^\star > 0$ by the assumption. Therefore, $C_B\setminus\{k\}$ is a cover. □

Let us replace $\theta_k^\star$ with 0. Then, the CG cut defined with the modified $(\theta_0^\star, \theta^\star)$ is described as follows.

$$\sum_{j\in C_B\setminus\{k\}} \lceil\theta_0^\star a_j\rceil x_j + \sum_{j\in N\setminus C_B} \lfloor\theta_0^\star a_j\rfloor x_j \leq \left\lfloor \theta_0^\star a_0 + \sum_{j\in C_B\setminus\{k\}}\theta_j^\star \right\rfloor. \tag{3.11}$$

The CG cut (3.11) is at least as strong as the CG cut $\sum_{j\in N}\alpha_j^\star x_j \le \alpha_0^\star$ as shown in the proof of Proposition 3.3. We replace C_B and the CG cut $\sum_{j\in N}\alpha_j^\star x_j \le \alpha_0^\star$ with $C_B\setminus\{k\}$ and the CG cut (3.11), respectively. By repeatedly eliminating $k \in C_B$ that violates the third condition of Definition 3.1 until no such k exists, we can obtain a maximal CG cut that is at least as strong as the given CG cut. Since one variable is removed from C_B in each iteration, the process terminates within at most $O(n)$ iterations.

We describe the overall algorithm to derive the maximal CG cut from the given one in Algorithm 7.

Algorithm 7 Construct a maximal CG cut from the given one

procedure MAXIMAL($\hat{\theta}_0, \hat{\theta}$)
  $\alpha_0 \leftarrow \hat{\theta}_0 a_0 + \sum_{j\in N}\hat{\theta}_j$, $a_j \leftarrow \lfloor \hat{\theta}_0 a_j + \hat{\theta}_j \rfloor$ for each $j \in N$;
  $C_B, l \leftarrow$solve $\max\{\alpha^T x : x \in \mathcal{P}_B\}$;
  $\theta_0^\star \leftarrow \alpha_l/a_l$, $\theta_j^\star \leftarrow \lceil \theta_0^\star a_j \rceil - \theta_0^\star a_j$ for each $j \in C_B$;
  $f_0^\star \leftarrow \theta_0^\star a_0 + \sum_{j\in N}\theta_j^\star - \lfloor \theta_0^\star a_0 + \sum_{j\in N}\theta_j^\star \rfloor$;
  if $f_0^\star < \frac{1}{2}$ **then**
    $t \leftarrow \lceil 1/(2f_0^\star) \rceil$;
    $\theta_0^\star \leftarrow t\theta_0^\star$, $\theta_j^\star \leftarrow t\theta_j^\star - \lfloor t\theta_j^\star \rfloor$;
  end if
  repeat
    $f_0^\star \leftarrow \theta_0^\star a_0 + \sum_{j\in N}\theta_j^\star - \lfloor \theta_0^\star a_0 + \sum_{j\in N}\theta_j^\star \rfloor$;
    Find $k \in C_B$ such that $\lceil \theta_0^\star a_k \rceil = 1$ and $\theta_k^\star > f_0^\star$;
    $C_B \leftarrow C_B \setminus \{k\}$, $\theta_k^\star \leftarrow 0$;
  until $\nexists k$;
  return $(\theta_0^\star, \theta^\star)$;
end procedure

We emphasize that a maximal CG cut can be obtained from any given CG cut in $O(n \log n)$ and is guaranteed to be at least as strong as the given cut. The following section presents a method further to strengthen maximal CG cuts for binary knapsack polytopes. We first reveal a link between the maximal CG cuts and cover inequalities for binary knapsack polytopes. The key idea of our strengthening

method is the use of a lifting function for cover inequalities to derive a stronger cut from the maximal CG cuts.

3.4 Strengthening maximal CG cuts for binary knapsack polytopes

3.4.1 Extended knapsack polytope and lifted cover inequalities

Let $\beta^T x \leq \beta_0$ be a given maximal CG cut for $\mathcal{X}_B$, which is defined with (θ_0, θ) for some $\theta_0 \in \mathbb{R}_+$, cover $C_B \subseteq N$ and $l \in C_B$, where $\theta_0 a_l$ is an integer, $\theta_j = \lceil \theta_0 a_j \rceil - \theta_0 a_j$ for each $j \in C_B$ and $\theta_j = 0$ for each $j \in N \setminus C_B$. $\beta^T x \leq \beta_0$ can be rewritten as

$$\sum_{j \in C_B} \lfloor \theta_0 a_j + \theta_j \rfloor + \sum_{j \in N \setminus C_B} \lfloor \theta_0 a_j \rfloor x_j \leq \left\lfloor \theta_0 a_0 + \sum_{j \in C_B} \theta_j \right\rfloor,$$

Additionally, let $\lambda_B = \sum_{j \in C_B} a_j - a_0$ and $f_0 = \theta_0 a_0 + \sum_{j \in C_B} \theta_j - \lfloor \theta_0 a_0 + \sum_{j \in C_B} \theta_j \rfloor$. Then, $\theta_j \leq f_0$ for all $j \in C_B$ by definition. For the sake of simplicity, let $\beta(C_B) = \sum_{j \in C_B} \beta_j$.

We introduce an extended formulation for $\mathcal{X}_B$ using C_B described as follows.

$$\mathcal{X}_E = \left\{ (w, x) \in \{0,1\}^{\beta(C_B)+n} : \begin{array}{l} \displaystyle\sum_{j \in C_B} \sum_{k \in [\beta_j]} \frac{a_j}{\beta_j} w_{jk} + \sum_{j \in N \setminus C_B} a_j x_j \leq a_0 \\ x_j \leq w_{jk}, \ \forall k \in [\beta_j], \ \forall j \in C_B \end{array} \right\}$$

where $[\beta_j]$ denotes $\{1, ..., \beta_j\}$. $\mathcal{X}_E$ has additional decision variables, w_{jk}'s, compared with $\mathcal{X}_B$.

Proposition 3.5. *$Proj_x(\mathcal{X}_E) = \mathcal{X}_B$.*

Proof. Let $(\hat{w}, \hat{x})$ be a feasible solution for $\mathcal{X}_E$. Since $\hat{x}_j \leq \hat{w}_{jk}$ for each $k \in [\beta_j]$ and

$j \in C_B$,

$$\sum_{j \in C_B} a_j \hat{x}_j + \sum_{j \in N \setminus C_B} a_j \hat{x}_j \leq \sum_{j \in C_B} \sum_{k \in [\beta_j]} \frac{a_j}{\beta_j} \hat{w}_{jk} + \sum_{j \in N \setminus C_B} a_j \hat{x}_j \leq a_0.$$

Therefore, $\hat{x} \in \mathcal{X}_B$. This result implies that $Proj_x(\mathcal{X}_E) \subseteq \mathcal{X}_B$.

Conversely, let $\hat{x} \in \mathcal{X}_B$. For each $j \in C_B$, we set $\hat{w}_{jk} = 1$ for all $k \in [\beta_j]$ if $\hat{x}_j = 1$ while $\hat{w}_{jk} = 0$ for all $k \in [\beta_j]$ such that $\hat{x}_j = 0$. Since $\hat{x}_j = \sum_{k \in [\beta_j]} \hat{w}_{jk}/\beta_j$,

$$\sum_{j \in C_B} \sum_{k \in [\beta_j]} \frac{a_j}{\beta_j} \hat{w}_{jk} + \sum_{j \in N \setminus C_B} a_j \hat{x}_j = \sum_{j \in C_B} a_j \hat{x}_j + \sum_{j \in N \setminus C_B} a_j \hat{x}_j \leq a_0,$$

which implies that $(\hat{w}, \hat{x}) \in \mathcal{X}_E$. Therefore, $\mathcal{X}_B \subseteq Proj_x(\mathcal{X}_E)$ and the result follows.

□

We call $\mathcal{X}_E$ as the *extended knapsack polytope*. Proposition 3.5 implies that all valid inequalities for $\mathcal{X}_B$ can be obtained from those for the extended knapsack polytope. We first provide a family of valid inequalities for the extended knapsack polytope, which includes the given maximal CG cut for $\mathcal{X}_B$.

The extended knapsack polytope is not a binary knapsack polytope due to the additional constraints, $x_j \leq w_{jk}$ for all $k \in [\beta_j]$ and $j \in C_B$. However, several valid inequalities for the extended knapsack polytope can be obtained from its relaxation, described as follows.

$$\mathcal{X}_{ER} = \left\{ (w, x) \in \{0,1\}^{\beta(C_B)+n} : \sum_{j \in C_B} \sum_{k \in [\beta_j]} \frac{a_j}{\beta_j} w_{jk} + \sum_{j \in N \setminus C_B} a_j x_j \leq a_0 \right\}.$$

We note that $\mathcal{X}_{ER}$ is a binary knapsack polytope. A cover inequality can be obtained

from the definition of $\mathcal{X}_{ER}$. Let C_E be the subset of variable w's such that

$$C_E = \{(j,k) : k \in [\beta_j],\ j \in C_B \setminus \{l\}\} \cup \{(l,k) : k \in \{1,...,\bar{\beta}_l\}\},$$

where $\bar{\beta}_l = \beta_l - \lfloor \theta_0 \lambda_B \rfloor$. Additionally, let $\lambda_E = \sum_{(j,k)\in C_E} a_j/\beta_j - a_0$.

Proposition 3.6. *C_E is a minimal cover for $\mathcal{X}_{ER}$.*

Proof. From the definition of C_E,

$$\sum_{(j,k)\in C_E} \frac{a_j}{\beta_j} = \sum_{j\in C_B\setminus\{l\}} a_j + \bar{\beta}_l \frac{a_l}{\beta_l}.$$

Because $\bar{\beta}_l = \beta_l - \lfloor \theta_0 \lambda_B \rfloor$,

$$\sum_{j\in C_B\setminus\{l\}} a_j + a_l \frac{\bar{\beta}_l}{\beta_l} = \sum_{j\in C_B\setminus\{l\}} a_j + a_l \left(1 - \frac{\lfloor \theta_0\lambda_B \rfloor}{\beta_l}\right) > \sum_{j\in C_B\setminus\{l\}} a_j + a_l \left(1 - \frac{\theta_0\lambda_B}{\beta_l}\right).$$

Then, since $\theta_0 = \beta_l/a_l$,

$$\sum_{j\in C_B\setminus\{l\}} a_j + a_l \left(1 - \frac{\theta_0\lambda_B}{\beta_l}\right) = \sum_{j\in C_B} a_j - \lambda_B = a_0.$$

Therefore, $\sum_{(j,k)\in C_E} a_j/\beta_j > a_0$, that is, C_E is a cover for $\mathcal{X}_{ER}$.

We show that $a_j/\beta_j \geq \lambda_E$ for all $j \in C_B$ to prove the minimality of C_E. Suppose that there exists $k \in C_B$ such that $a_k/\beta_k < \lambda_E$. By definition, λ_E can be rewritten as follows.

$$\lambda_E = \lambda_B - (\beta_l - \bar{\beta}_l)\frac{a_l}{\beta_l} = \lambda_B - \frac{\beta_l - \bar{\beta}_l}{\theta_0}$$

In addition, f_0 can be rewritten as follows.

$$f_0 = \lceil \theta_0 \lambda_B \rceil - \theta_0 \lambda_B = (\beta_l - \bar{\beta}_l + 1) - \theta_0 \lambda_B.$$

These results imply that $\lambda_E = (1 - f_0)/\theta_0$. Because $a_k/\beta_k < \lambda_E$,

$$\frac{\theta_0 a_k}{\beta_k} < 1 - f_0 \quad \Rightarrow \quad \frac{\beta_k - \theta_k}{\beta_k} < 1 - f_0,$$

which implies that $f_0 < \theta_k/\beta_k$. By definition of the maximal CG cut, $f_0 \geq \theta_k$ if $\beta_k = 1$. In addition, $f_0 \geq \theta_k/\beta_k$ if $\beta_k \geq 2$ because $\frac{1}{2} \leq f_0$. This result implies that no such k exists. Therefore, C_E is a minimal cover. □

From the definition of C_E, it can be easily shown that $|C_E| = \beta_0 + 1$. Then, a cover inequality for $\mathcal{X}_{ER}$ can be defined as follows.

$$\sum_{(j,k) \in C_E} w_{jk} \leq \beta_0 - 1, \tag{3.12}$$

The cover inequality is described with only the variables in C_E. However, it can be written in a general form using the variables not in C_E, the so-called lifted cover inequalities, as follows.

$$\sum_{(j,k) \in C_E} w_{jk} + \sum_{k \in [\beta_l] \setminus [\bar{\beta}_l]} \nu_k w_{lk} + \sum_{j \in N \setminus C_B} \gamma_j x_j \leq \beta_0 \tag{3.13}$$

where $\nu_k \geq 0$ for all $k \in [\beta_l] \setminus [\bar{\beta}_l]$ and $\gamma_j \geq 0$ for all $j \in N \setminus C_B$. The lifted cover inequalities can define some facets of $\mathcal{X}_{ER}$ because C_E is a minimal cover.

The lifted cover inequalities for $\mathcal{X}_{ER}$ are also valid for $\mathcal{X}_E$ because $\mathcal{X}_E \subseteq \mathcal{X}_{ER}$.

The following proposition states that the converse is also true, that is, if an inequality of the form (3.13) is valid for $\mathcal{X}_E$, it is also valid for $\mathcal{X}_{ER}$.

Proposition 3.7. *The inequality* (3.13) *is valid for* $\mathcal{X}_E$ *if and only if it is valid for* $\mathcal{X}_{ER}$.

Proof. We only show the "only if" direction because the "if" direction is clearly true. Let $\mathcal{W}_E = \{(w, x) \in \mathcal{X}_E : x_j = 0,\ j \in C_B\}$. Then,

$$\sum_{(j,k)\in C_E} \hat{w}_{jk} + \sum_{k\in[\beta_l]\setminus[\bar{\beta}_l]} \nu_k \hat{w}_{lk} + \sum_{j\in N\setminus C_B} \gamma_j \hat{x}_j \le \beta_0,\ (\hat{w}, \hat{x}) \in \mathcal{W}_E \tag{3.14}$$

by the assumption on the validity of the inequality (3.13) for $\mathcal{X}_E$. Let us consider the following optimization problem.

$$\begin{aligned} \max \quad & \sum_{(j,k)\in C_E} w_{jk} + \sum_{k\in[\beta_l]\setminus[\bar{\beta}_l]} \nu_k w_{lk} + \sum_{j\in N\setminus C_B} \gamma_j x_j \\ \text{s.t} \quad & (w, x) \in \mathcal{X}_{ER} \end{aligned}$$

It is clear that there exists an optimal solution $(w^\star, x^\star)$ for this problem, such that $x^\star_j = 0$ for all $j \in C_B$. We note that a solution for $\mathcal{X}_{ER}$ where $x_j = 0$ for all $j \in C_B$ is feasible for $\mathcal{X}_E$. Therefore, $(w^\star, x^\star) \in \mathcal{W}_E$, and

$$\sum_{(j,k)\in C_E} w^\star_{jk} + \sum_{k\in[\beta_l]\setminus[\bar{\beta}_l]} \nu_k w^\star_{lk} + \sum_{j\in N\setminus C_B} \gamma_j x^\star_j \le \beta_0$$

due to (3.14). These results imply that

$$\sum_{(j,k)\in C_E} \hat{w}_{jk} + \sum_{k\in[\beta_l]\setminus[\bar{\beta}_l]} \nu_k \hat{w}_{lk} + \sum_{j\in N\setminus C_B} \gamma_j \hat{x}_j \le \beta_0,\ (\hat{w}, \hat{x}) \in \mathcal{X}_{ER}.$$

Therefore, the inequality (3.13) is also valid for $\mathcal{X}_{ER}$. □

Proposition 3.7 implies that valid inequalities for $\mathcal{X}_E$ of the form (3.13) are lifted cover inequalities for $\mathcal{X}_{ER}$. We call the inequalities of the form (3.13) as lifted cover inequalities without distinction for $\mathcal{X}_E$ or $\mathcal{X}_{ER}$.

We can obtain a valid inequality for $\mathcal{X}_B$ from a lifted cover inequality for $\mathcal{X}_E$ using the relationship between the variables w and x.

Proposition 3.8. *Suppose a lifted cover inequality of the form* (3.13) *is given. Then, the inequality*

$$\sum_{j \in C_B \setminus \{l\}} \beta_j x_j + \left(\bar{\beta}_l + \sum_{k \in [\beta_l] \setminus [\bar{\beta}_l]} \nu_k \right) x_l + \sum_{j \in N \setminus C_B} \gamma_j x_j \leq \beta_0. \tag{3.15}$$

is valid for $\mathcal{X}_B$.

Proof. Because $x_j \leq w_{jk}$ for each $k \in [\beta_j]$ and $j \in C_B$ in $\mathcal{X}_E$, the lifted cover inequality is valid although w_{jk}'s are replaced with x_j for each $k \in [\beta_j]$ and $j \in C_B$. Therefore, the result follows. □

We call the inequality of the form (3.15) as the *x-inequality* of the lifted cover inequality of the form (3.13) because the inequality (3.15) is defined with only x.

Now, we show that the given maximal CG cut is a x-inequality for some lifted cover inequality. One possible way to generate a lifted cover inequality is using lifting functions ψ for cover inequalities described in Section 1.2.3. Then, the resulting lifted cover inequality is described as follows.

$$\sum_{(j,k) \in C_E} w_{jk} + \sum_{k \in [\beta_l] \setminus [\bar{\beta}_l]} \psi \left(\frac{a_l}{\beta_l} \right) w_{lk} + \sum_{j \in N \setminus C_B} \psi(a_j) x_j \leq \beta_0.$$

Here, a_l/β_l can be rewritten as $1/\theta_0$ by definition.

Proposition 3.9. *$\psi_{CG}(z) = \lfloor \theta_0 z \rfloor$ is a lifting function.*

Proof. By Theorem 1.1, we show that $\psi_{CG}(z)$ is superadditive, and $\psi_{CG}(z) \leq \psi^\star(z)$ for all $z \in [0, a_0]$. Since the former is clear, we only show the latter. Let us consider the following linear program,

$$\max \left\{ \sum_{(j,k) \in C_E} w_{jk} : (w, x) \in \mathcal{P}_{ER} \right\}, \tag{3.16}$$

where $\mathcal{P}_{ER}$ is the linear relaxation of $\mathcal{X}_{ER}$. This problem is the LP relaxation of a binary knapsack problem. Therefore, an optimal solution $(w^\star, x^\star)$ can be defined as $x^\star_j = 0$ for all $j \in N$, $w^\star_{jk} = 1$ for all $(j,k) \in C_E \setminus \{(l, \bar{\beta}_l)\}$, $w^\star_{lk} = 1$ for all $k \in [\beta_l] \setminus [\bar{\beta}_l]$, and $w^\star_{l\bar{\beta}_l} = 1 - \theta_0 \lambda_E$. The corresponding optimal objective value is $|C_E| - \theta_0 \lambda_E$ where $|C_E| = \beta_0 + 1$ by definition. Let us consider the dual problem of the inner optimization problem presented in the definition of $\psi^\star$, which is described as follows.

$$\begin{aligned}
\min \quad & v_0(a_0 - z) + \sum_{(j,k) \in C_E} v_{jk} \\
\text{s.t} \quad & 1 \leq v_0 \frac{a_j}{\beta_j} + v_{jk}, \ (j,k) \in C_E \\
& 0 \leq v_0 \frac{a_l}{\beta_l} + v_{lk}, \ k \in [\beta_l] \setminus [\bar{\beta}_l] \\
& 0 \leq v_0 a_j + v_j, \ j \in N \setminus C_B \\
& (v_0, v) \in \mathbb{R}_+^{\beta(C_B)+n+1}
\end{aligned} \tag{3.17}$$

We note that when $z = 0$, the dual problem (3.17) is equivalent to the dual problem

of the problem (3.16).

By the property of the LP relaxation of a binary knapsack problem, there exists an optimal solution for the problem (3.17) when $z = 0$, such that $v_0 = \theta_0$. We define such an optimal solution as $(\theta_0, v^\star)$. Then, $(\theta_0^\star, v^\star)$ is also feasible for the problem (3.17) when $z \neq 0$. Therefore,

$$\psi^\star(z) \geq \beta_0 - \left\lfloor \theta_0(a_0 - z) + \sum_{(j,k)\in C_E} v_{jk}^\star \right\rfloor \geq \beta_0 - \left\lfloor \theta_0 a_0 + \sum_{(j,k)\in C_E} v_{jk}^\star \right\rfloor + \lfloor \theta_0 z \rfloor,$$

where the last inequality holds due to the superadditivity of the rounding down function. Because $(\theta_0, v^\star)$ is an optimal dual solution for the problem (3.16),

$$\left\lfloor \theta_0 a_0 + \sum_{(j,k)\in C_E} v_{jk}^\star \right\rfloor = \lfloor \beta_0 + 1 - \theta_0 \lambda_E \rfloor = \beta_0.$$

This result implies that $\psi^\star(z) \geq \lfloor \theta_0 z \rfloor$ for all $z \in [0, a_0]$. Therefore, $\lfloor \theta_0 z \rfloor$ is a lifting function. □

By Proposition 3.9, a lifted cover inequality can be obtained through ψ_{CG}, represented as follows.

$$\sum_{(j,k)\in C_E} w_{jk} + \sum_{k\in[\beta_l]\setminus[\bar{\beta}_l]} w_{lk} + \sum_{j\in N\setminus C_B} \lfloor \theta a_j \rfloor x_j \leq \beta_0$$

The x-inequality of the above inequality is equivalent to the given maximal CG cut $\beta^T x \leq \beta_0$. This observation implies that a stronger lifted cover inequality can derive a cut for $\mathcal{X}_B$, which dominates the given maximal CG cut.

3.4.2 CG cut strengthening method using a lifting function for cover inequalities

Our CG cut strengthening method starts by generating a stronger lifted cover inequality than the one obtained through ψ_{CG}. For the sake of simplicity, let us redefine C_E as $C'_E = \{1, \ldots, \beta_0 + 1\}$ where there exists a one-to-one correspondence between each $(j,k) \in C_E$ and each $i \in C'_E$. We also define $\bar{a}_i = a_j/\beta_j$ for each $i \in C'_E$ and the corresponding $(j,k) \in C_E$.

Without loss of generality, we assume that $\bar{a}_1 \geq \cdots \geq \bar{a}_{\beta_0} + 1$. Then, $\bar{a}_1 = 1/\theta_0$. Additionally, let $\mu_0 = 0$, $\mu_j = \sum_{i=1}^{j} \bar{a}_i$ for all $j \in C'_E$. Let us consider the lifting function proposed by Gu et al. (2000), which is defined as

$$h(z) = \begin{cases} k & \text{if } \mu_k - \lambda_E + \rho_k \leq z \leq \mu_{k+1} - \lambda_E \\ k - \dfrac{\mu_k - \lambda_E + \rho_k - z}{\rho_1} & \text{if } \mu_k - \lambda_E < z < \mu_k - \lambda_E + \rho_k \end{cases},$$

for some $k = 0, \ldots, \beta_0$ where $\rho_j = \max\{0, \bar{a}_{j+1} - (\bar{a}_1 - \lambda_E)\}$ for each $j = 0, ..., \beta_0$.

Theorem 3.1. $h(z) \geq \psi_{CG}(z),\ \forall z \in [0, a_0]$.

Proof. We denote the range of z such that $k \leq h(z) < k+1$ as $I(k)$ for each $k \in \{0, \ldots, \beta_0\}$, that is,

$$I(k) = [\mu_k - \lambda_E + \rho_k, \mu_{k+1} - \lambda_E + \rho_{k+1}).$$

We show that $\psi_{CG}(z) \leq k$ for all $z \in I(k)$ for all $k \in \{0, \ldots, \beta_0\}$.

Suppose that there exists $\hat{z} \in I(k)$ for some $k \in \{0, \ldots, \beta_0\}$ such that $\psi_{CG}(\hat{z}) > k$. By the definition of ψ_{CG}, $\hat{z} \geq (k+1)/\theta_0$. On the other hand, the upper bound of

$I(k)$ is

$$\mu_{k+1} - \lambda_E + \rho_{k+1} = \begin{cases} \sum_{j=1}^{k+1} \bar{a}_j - \lambda_E & \text{, if } \rho_{k+1} = 0 \\ \sum_{j=2}^{k+2} \bar{a}_j & \text{, otherwise.} \end{cases}$$

by the definition of μ_j's and ρ_j's. Because $\bar{a}_j \leq 1/\theta_0$ for all $j \in C'_E$,

$$\hat{z} < \mu_{k+1} - \lambda_E + \rho_{k+1} \leq \frac{k+1}{\theta_0}$$

whether $\rho_{k+1} = 0$ or not. This result contradicts the assumption that $\hat{z} \geq (k+1)/\theta_0$. Therefore, the result follows. □

The lifted cover inequality obtained through the function h is

$$\sum_{(j,k)\in C_E} w_{jk} + \sum_{k\in[\beta_l]\setminus[\bar{\beta}_l]} h\left(\frac{1}{\theta_0}\right) w_{lk} + \sum_{j\in N\setminus C_B} h(a_j)x_j \leq \beta_0,$$

where the x-inequality is

$$\sum_{j\in C_B\setminus\{t\}} \beta_j x_j + \left(\bar{\beta}_l + (\beta_l - \bar{\beta}_l)h\left(\frac{1}{\theta_0}\right)\right) x_l + \sum_{j\in N\setminus C_B} h(a_j)x_j \leq \beta_0. \tag{3.18}$$

We call the inequality (3.18) as the *SCG cut*.

Theorem 3.2. *The SCG cut is equivalent to or dominates the given maximal CG cut.*

Proof. By Theorem 3.1, it is clear that the coefficients of the SCG cut are greater than or equal to those of the given maximal CG cut except for x_l. Hence, we only show that

$$\bar{\beta}_l + (\beta_l - \bar{\beta}_l)h\left(\frac{1}{\theta_0}\right) \geq \beta_l.$$

$h(z) \geq 1$ if $z \geq \mu_1 - \lambda_E + \rho_1$ where

$$\mu_1 - \lambda_E + \rho_1 = \begin{cases} \frac{1}{\theta_0} - \lambda_E & \text{, if } \rho_1 = 0 \\ \bar{a}_2 & \text{, otherwise.} \end{cases}$$

Therefore, it is clear that

$$\frac{1}{\theta_0} \geq \mu_1 - \lambda_E + \rho_1,$$

whether $\rho_1 = 0$ or not and the following inequality holds.

$$\bar{\beta}_l + (\beta_l - \bar{\beta}_l) h\left(\frac{1}{\theta_0}\right) \geq \bar{\beta}_l + (\beta_l - \bar{\beta}_l) = \beta_t$$

Therefore, the SCG cut is at least as strong as the given maximal CG cut. □

It may seem difficult to obtain the SCG cut efficiently. Because β_0 can be arbitrarily large, computing μ_j's and identifying the range of z for given $z \in [0, a_0]$ may require a significant amount of time to obtain the value of $h(z)$. However, $h(z)$ can be computed efficiently because the variables in C'_E have at most $|C_B|$ different values of $\bar{a}_j$.

Without loss of generality, let $C_B = \{1, \ldots, |C_B|\}$ where

$$\frac{a_1}{\beta_1} \geq \cdots \geq \frac{a_{|C_B|}}{\beta_{|C_B|}}.$$

We note that $a_1/\beta_1 = 1/\theta_0$. Then, we define $\bar{\mu} = 0$, $\bar{\mu}_1 = \bar{\beta}_l/\theta_0$, and $\bar{\mu}_j = \bar{\mu}_1 + \sum_{i=2}^{j} a_j$ for each $j \in \{2, ..., |C_B|\}$. Additionally, let

$$\bar{\rho}_j = \max\left\{0, \frac{a_{j+1}}{\beta_{j+1}} - \left(\frac{1}{\theta_0} - \lambda_E\right)\right\}, \ \forall j \in \{1, ..., |C_B|\}.$$

Then, for given $z \in [0, a_0]$, we can find q in $O(\log |C_B|)$ such that

$$\bar{\mu}_q - \lambda_E + \bar{\rho}_q \leq z < \bar{\mu}_{q+1} - \lambda_E + \bar{\rho}_{q+1}.$$

Let $\bar{\mu}_q = \mu_{k_1}$ and $\bar{\mu}_{q+1} = \mu_{k_2}$ where $k_1 = \bar{\beta}_l + \sum_{j=2}^{q} \beta_j$ and $k_2 = \bar{\beta}_l + \sum_{j=2}^{q+1} \beta_j$. Then, for each $k_1 + 1 \leq k \leq k_2$,

$$\mu_{k+1} - \mu_k = \frac{a_{q+1}}{\beta_{q+1}},$$

and

$$\rho_k = \max\left\{0, \frac{a_{q+1}}{\beta_{q+1}} - \left(\frac{1}{\theta_0} - \lambda_E\right)\right\}.$$

We can define $q^\star$ as follows.

$$q^\star = \left\lfloor \frac{z - \mu_{k_1} + \lambda - \rho_{k_1}}{\frac{a_{q+1}}{\beta_{q+1}}} \right\rfloor$$

Then, the given z is in the following range.

$$\mu_{k_1+q^\star} - \lambda_E + \rho_{k_1+q^\star} \leq z < \mu_{k_1+q^\star+1} - \lambda_E + \rho_{k_1+q^\star+1}$$

In this range,

$$h(z) = \begin{cases} k + q^\star & \text{, if } z \leq \mu_{k_1+q^\star+1} - \lambda_E \\ k + q^\star + 1 - \frac{\mu_{k_1+q^\star+1} - \lambda_E + \rho_{k_1+q^\star+1} - z}{\rho_1} & \text{, otherwise.} \end{cases}$$

The overall procedure to compute $h(z)$ for given $z \in [0, b]$ is described in Algorithm 8.

If $\bar{\mu}_j$'s and $\bar{\rho}_j$'s are given, $h(z)$ can be computed in $O(\log n)$. Hence, the SCG cut

Algorithm 8 Computation of $h(z)$

$\bar{\mu}_0 \leftarrow 0$, $\bar{\mu}_1 \leftarrow \bar{\beta}_l/\theta_0$, and $\bar{\mu}_j \leftarrow \bar{\mu}_1 + \sum_{i=2}^{j} a_j$ for each $j = 2, ..., |C_B|$;
$\bar{\rho}_j \leftarrow \max\{0, a_{j+1}/\beta_{j+1} - (1/\theta_0 - \lambda_E)\}$ for each $j = 1, ..., |C_B| - 1$;
Find q such that $\bar{\mu}_q - \lambda + \bar{\rho}_q \leq z < \bar{\mu}_{q+1} - \lambda + \bar{\rho}_{q+1}$;
$k \leftarrow \gamma + \sum_{j=2}^{q} \beta_j$;
$q^\star \leftarrow \lfloor (z - \mu_k + \lambda + \rho_k)/(a_{q+1}/\beta_{q+1}) \rfloor$ and $\rho \leftarrow \max\{0, a_{q+1}/\beta_{q+1} - (1/\theta_0 - \lambda_E)\}$;
if $z \leq \bar{\mu}_q + (q^\star + 1)a_{q+1}/\beta_{q+1} - \lambda_E$ **then**
    $h(z) \leftarrow k + q^\star$;
else
    $h(z) \leftarrow k + q^\star + 1 - (\mu_{k_1+q^\star+1} - \lambda_E + \rho_{k_1+q^\star+1} - z)/\rho_1$;
end if
return $h(z)$;

can be obtain in $O(n \log n)$ by computing $h(a_j)$ for each $j \in N \setminus C_B$ and $h(1/\theta_0)$.

3.4.3 Strength of the SCG cut

In this section, we compare the strength of the SCG cut with other cuts strengthened from the given maximal CG cut. We first show that the SCG cut can have a Chvátal rank higher than 1.

Example 3.1. *Let us consider the following binary knapsack polytope.*

$$\mathcal{X}_B = \{x \in \{0,1\}^5 : 3x_1 + 5x_2 + 8x_3 + 7x_4 + 12x_5 \leq 14\}$$

Suppose that a CG cut defined with $(u_0, u_1, u_2, u_3, u_4, u_5) = (\frac{1}{8}, \frac{5}{8}, \frac{3}{8}, 0, 0, 0)$ *is given, which is described as follows.*

$$x_1 + x_2 + x_3 + \lfloor 0.875 \rfloor x_4 + \lfloor 1.5 \rfloor x_5 \leq \lfloor 2.75 \rfloor \quad \Rightarrow \quad x_1 + x_2 + x_3 + x_5 \leq 2$$

The above CG cut is a maximal CG cut where $C_B = \{1, 2, 3\}$*. Then, the function* h *is defined as Figure 3.2 The corresponding SCG cut is defined as follows.*

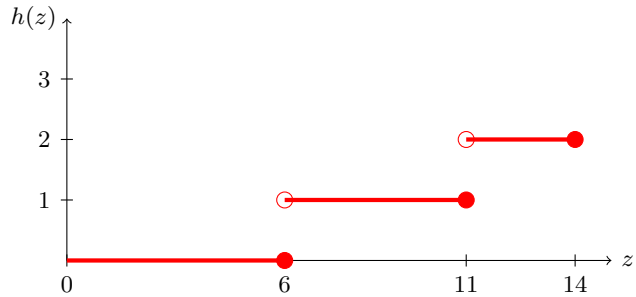


Figure 3.2: Example of $h(z)$

$$\textit{SCG cut: } x_1 + x_2 + x_3 + x_4 + 2x_5 \leq 2$$

It can be easily checked that the above inequality defines a facet for $\mathcal{X}_B$. Furthermore, the SCG cut is not a rank-1 CG cut.

As mentioned in Section 3.1, the given maximal CG cut can be strengthened using cut-generating functions. Recall that the given maximal CG cut $\beta^T x \leq \beta_0$ is equivalent to

$$\sum_{j \in C_B} \lfloor \theta_0 a_j + \theta_j \rfloor + \sum_{j \in N \setminus C_B} \lfloor \theta_0 a_j \rfloor x_j \leq \left\lfloor \theta_0 a_0 + \sum_{j \in C_B} \theta_j \right\rfloor,$$

where $\theta_0 a_j + \theta_j$ is integer for each $j \in C_B$. Then, the maximal CG cut is obtained by applying π_{CG} to

$$\sum_{j \in C_B} (\theta_0 a_j + \theta_j) x_j + \sum_{j \in N \setminus C_B} \theta_0 a_j x_j \leq \theta_0 a_0 + \sum_{j \in C_B} \theta_j. \tag{3.19}$$

Let f_0 be the fractional part of the right-hand side of the inequality (3.19).

A CGF cut derived from (3.19) with a cut-generating function π can be repre-

sented as

$$\sum_{j \in C_B} \beta_j x_j + \sum_{j \in N \setminus C_B} (\theta_0 a_j - \pi(\theta_0 a_j)) x_j \leq \beta_0, \tag{3.20}$$

We compare the strength of the SCG cut with CGF cuts that can dominate the maximal CG cut. We first show that the SCG cut can dominate every CGF cuts under certain condition.

Theorem 3.3. *If $h(a_j) \in \mathbb{Z}_+$ for all $j \in N \setminus C_B$, then the SCG cut is at least as strong as every CGF cut.*

Proof. Let us consider the following implied inequality of $\mathcal{X}_{ER}$.

$$\sum_{(j,k) \in C_E} \left(\theta_0 \frac{a_j}{\beta_j} + v_{jk} \right) w_{jk} + \sum_{k \in [\beta_l] \setminus [\bar{\beta}_l]} \theta_0 \frac{a_t}{\beta_t} w_{tk} + \sum_{j \in N \setminus C_B} \theta_0 a_j x_j \leq \theta_0 a_0 + \sum_{(j,k) \in C_E} v_{jk}, \tag{3.21}$$

where $v_{jk} = \theta_j / \beta_j$ for all $(j,k) \in C_E$. We note that the right-hand sides of the above inequality and (3.19) are equivalent. Then, a cut-generating function π for (3.19) can be applied to (3.21). The obtained CGF cut is expressed as

$$\sum_{(j,k) \in C_E} w_{jk} + \sum_{k \in [\beta_l] \setminus [\bar{\beta}_l]} w_{jk} + \sum_{j \in N \setminus C_B} (\theta_0 a_j - \pi(\theta_0 a_j)) x_j \leq \beta_0.$$

We can see that the CGF cut is a lifted cover inequality of the form (3.13) for $\mathcal{X}_{ER}$, and its x-inequality is equivalent to the CGF cut (3.20) for $\mathcal{X}_B$. Therefore, it is clear that

$$\theta_0 a_j - \pi(\theta_0 a_j) \leq \psi^\star(a_j), \ \forall j \in N \setminus C_B.$$

On the other hand, the assumption that $h(a_j) \in \mathbb{Z}_+$ for all $j \in N \setminus C_B$ implies that $h(a_j) = \psi^\star(a_j)$ for all $j \in N \setminus C_B$ (Gu et al., 2000). Therefore, for any cut-

generating function π,

$$\theta_0 a_j - \pi(\theta_0 a_j) \leq \psi^\star(a_j) = h(a_j),\ \forall j \in N \setminus C_B,$$

which implies that the SCG cut is at least as strong as every CGF cut. □

In general cases, there may exist a CGF cut that dominates the SCG cut. However, we show that the SCG cut is always at least as strong as the Gomory mixed-integer cut derived from (3.19) using $\pi_G^{f_0}$ defined as

$$\pi_G^{f_0}(z) = \min\left\{\pi_{CG}(z), \frac{f_0(1-\pi_{CG}(z))}{1-f_0}\right\}.$$

The Gomory mixed-integer cut can be represented as follows.

$$\sum_{j\in C_B} \beta_j x_j + \sum_{j\in N\setminus C_B} \left(\beta_j + \max\left\{0, \frac{\pi_{CG}(\theta_0 a_j) - f_0}{1-f_0}\right\}\right) x_j \leq \beta_0, \tag{3.22}$$

Theorem 3.4. *The SCG cut is equivalent to or dominates the Gomory mixed-integer cut* (3.22).

Proof. As mentioned in the proof of Theorem 3.3, the cut-generating function $\pi_G^{f_0}$ can yield a lifted cover inequality for $\mathcal{X}_{ER}$ from the inequality (3.21). The lifted cover inequality can be represented as follows.

$$\sum_{(j,k)\in C_E} w_{jk} + \sum_{k\in[\beta_l]\setminus[\bar{\beta}_l]} w_{jk} + \sum_{j\in N\setminus C_B} \psi_G^{f_0}(a_j) x_j \leq \beta_0,$$

where

$$\psi_G^{f_0}(z) = \lfloor \theta_0 z \rfloor + \max\left\{0, \frac{\pi_{CG}(\theta_0 z) - f_0}{1-f_0}\right\}.$$

We note that the x-inequality of the above is equivalent to the Gomory mixed-integer cut (3.22). To compare the strength of the SCG cut and the Gomory mixed-integer cut, we show that $\psi_G^{f_0}(z) \leq h(z)$ for all $z \in [0, a_0]$. Let $I(k)$ be the range of z such that $k \leq h(z) < k+1$ for some $k \in \{0, ..., \beta_0\}$, as follows.

$$I(k) = [\mu_k - \lambda_E + \rho_k, \mu_{k+1} - \lambda_E + \rho_{k+1})$$

As shown in the proof of Proposition 3.1, $\mu_k - \lambda_E + \rho_k \leq k/\theta_0$. If $z < k/\theta_0$, then $\psi_G^{f_0}(z) < h(z)$ since $\psi_G^{f_0}(z) < k$. Therefore, we only consider the case,

$$\frac{k}{\theta_0} \leq z < \mu_{k+1} - \lambda_E + \rho_{k+1},$$

where $\mu_{k+1} - \lambda_E + \rho_{k+1} \leq (k+1)/\theta_0$.

In this range, by definition, $\psi_G^{f_0}(z) > k$ when $z > (k+f_0)/\theta_0$, and $h(z) > k$ when $z > \mu_{k+1} - \lambda_E$. Recall that $\lambda_E = (1-f_0)/\theta_0$ as shown in the proof of Proposition 3.6. Then,

$$\mu_{k+1} - \lambda_E \leq \frac{k+1}{\theta_0} - \frac{1-f_0}{\theta_0} = \frac{k+f_0}{\theta_0}.$$

This result implies that $h(z) = \psi_G^{f_0}(z) = k$ if $z \leq \mu_{k+1} - \lambda_E$.

Let us consider the rest range

$$\mu_{k+1} - \lambda_E < z < \mu_{k+1} - \lambda_E + \rho_{k+1}.$$

Because $\mu_{k+1} - \lambda_E \leq (k+f_0)/\theta_0$ and $\psi_G^{f_0}(z)$ is continuous,

$$\lim_{z \to (\mu_{k+1} - \lambda_E)+} h(z) \geq k,$$

while

$$\lim_{z \to (\mu_{k+1} - \lambda_E)+} \psi_G^{f_0}(z) = k.$$

In addition, $h(z)$ and $\psi_G^{f_0}(z)$ are linear and increasing functions in this range, where

$$\lim_{z \to (\mu_{k+1} - \lambda_E + \rho_{k+1})-} h(z) \geq \lim_{z \to (\mu_{k+1} - \lambda_E + \rho_{k+1})-} \psi_G^{f_0}(z)$$

because $\mu_{k+1} - \lambda_E + \rho_{k+1} \leq (k+1)/\theta_0$. These results imply that $h(z) \geq \psi_G^{f_0}(z)$ for all z in this range, and $h(z) \geq \psi_G(z)$ for all $z \in I(k)$. Therefore, $h(z) \geq \psi_G(z)$ for all $z \in [0, a_0]$ and the result follows. □

Example 3.1 (Continued)**.** *Recall that the given CG cut is*

$$x_1 + x_2 + x_3 + \lfloor 0.875 \rfloor x_4 + \lfloor 1.5 \rfloor x_5 \leq \lfloor 2.75 \rfloor$$

Using $\psi_G^{f_0}$ where $f_0 = 0.75$, the Gomory mixed-integer cut can be derived from the above CG cut as follows.

$$x_1 + x_2 + x_3 + 0.5x_4 + x_5 \leq 2$$

Figure 3.3 illustrates $\psi_G^{f_0}(z)$ and $h(z)$ in blue and red lines, respectively. As shown in Theorem 3.4, $h(z)$ dominates $\psi_G^{f_0}(z)$.

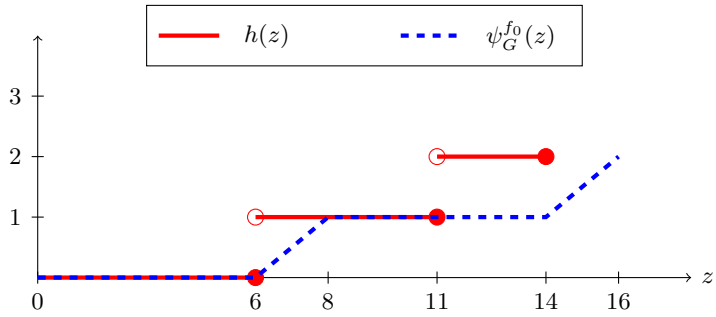


Figure 3.3: Comparison of $h(z)$ and $\psi_G^{f_0}(z)$

3.5 Extension to binary integer linear programs with generalized upper bounds

In this section, we extend the strengthening method to CG cuts for binary integer linear programs with generalized upper bounds. We first show that such a CG cut can be viewed as a CG cut for a GKP polytope. We then define maximal CG cuts for GKP polytopes and provide a way to obtain them from any CG cuts for a GKP polytope. Subsequently, we propose the method for strengthening maximal CG cuts for GKP polytopes, using the method proposed in Section 3.4.

Recall a CG cut (3.1) and the corresponding single-constraint relaxation,

$$\mathcal{X}_S = \left\{ x \in \{0,1\}^n : \sum_{j \in N} u^T A_j x_j \leq u^T \mathbf{b} \right\}.$$

For the binary integer linear programs with generalized upper bounds, we regard that the system $Ax \leq \mathbf{b}$ has only the generalized upper bounds without variable bounds since the latter is redundant. Let K_i for each $i \in I$ be the GUB set such that $N = \cup_{i \in I} K_i$ and $K_i \cap K_j = \emptyset$ for all $i, j \in I$ where $i \neq j$ and $I = \{1, \ldots, m\}$. Without loss of generality, let u_i be the multipliers corresponding to $\sum_{j \in K_i} x_j \leq 1$

for each $i \in I$. Then, we can define a tighter relaxation for the binary integer linear program, $\mathcal{X}_{SG}$, which is described as follows.

$$\mathcal{X}_{SG} = \left\{ x \in \{0,1\}^n : \begin{array}{l} \displaystyle\sum_{i \in I} \sum_{j \in K_i} (u^T A_j - u_i) x_j \leq u^T \mathbf{b} - \sum_{i \in I} u_i, \\ \displaystyle\sum_{j \in K_i} x_j \leq 1, \ i \in I \end{array} \right\}.$$

We show that the CG cut (3.1) is represented as a CG cut for $\mathcal{X}_{SG}$.

$\mathrm{conv}(\mathcal{X}_{SG})$ can be reformulated as a GKP polytope by complementing variables. Let $j(i) = \arg\min\{u^T A_j - u_i : j \in K_i\}$ and $I^- = \{i \in I : u^T A_{j(i)} - u_i < 0\}$. Additionally, let us introduce decision variables $z \in \{0,1\}^n$ such that $z_j = 1 - \sum_{k \in K_i} x_k$ if $j = j(i)$ for all $i \in I^-$ and $z_j = x_j$, otherwise. Then, $\mathcal{X}_{SG}$ can be redefined with z as $\mathcal{X}_G$ described as follows.

$$\mathcal{X}_G = \left\{ z \in \{0,1\}^n : \begin{array}{l} \displaystyle\sum_{i \in I} \sum_{j \in K_i} a_j z_j \leq u^T \left(\mathbf{b} - \sum_{i \in I^-} A_{j(i)} \right) - \sum_{i \in I \setminus I^-} u_i, \\ \displaystyle\sum_{j \in K_i} z_j \leq 1, \ i \in I \end{array} \right\}.$$

where

$$a_j = \begin{cases} u^T A_j - u_i & \text{if } j \in K_i \text{ for some } i \in I \setminus I^- \\ -(u^T A_j - u_i) & \text{if } j = j(i) \text{ for some } i \in I^- \\ u^T A_j - u^T A_{j(i)} & \text{if } j \in K_i \setminus \{j(i)\} \text{ for some } i \in I^- \end{cases}.$$

We note that $a_j \geq 0$ for all $j \in N$. Hence, $\mathrm{conv}(\mathcal{X}_G)$ is a GKP polytope introduced in Chapter 2. The CG cut (3.1) can be redefined with z as the following valid inequality

for $\mathcal{X}_G$.

$$\sum_{i\in I\setminus I^-}\sum_{j\in K_i}\lfloor u^T A_j\rfloor z_j - \sum_{i\in I^-}\lfloor u^T A_{j(i)}\rfloor z_j + \sum_{i\in I^-}\sum_{j\in K_i\setminus\{j(i)\}}(\lfloor u^T A_j\rfloor - \lfloor u^T A_{j(i)}\rfloor)z_j \leq \lfloor u^T \mathbf{b}\rfloor - \sum_{i\in I^-}\lfloor u^T A_{j(i)}\rfloor \quad (3.23)$$

Proposition 3.10. *The inequality* (3.23) *is a CG cut for* $\mathcal{X}_G$.

The above proposition can be shown similarly to Proposition 3.1. We omit the proof for brevity.

Proposition 3.10 states that the CG cut for $\mathcal{X}_{SG}$ can be derived from a CG cut for $\mathcal{X}_G$. In addition, the dominance relationship between two valid inequalities for $\mathcal{X}_{SG}$ is preserved in the corresponding valid inequalities for $\mathcal{X}_G$, and the converse also holds. Therefore, we only consider CG cuts for the GKP polytope $\mathcal{X}_G$ and their strengthening method.

3.5.1 Maximal CG cuts for GKP polytopes

We introduce analogs of the results in Section 3.3 for GKP polytopes. For the sake of brevity, we redefine $\mathcal{X}_G$ as

$$\mathcal{X}_G = \left\{ x \in \{0,1\}^n : \begin{array}{l} \displaystyle\sum_{i\in I}\sum_{j\in K_i} a_j x_j \leq a_0, \\ \displaystyle\sum_{j\in K_i} x_j \leq 1,\ i \in I \end{array} \right\}.$$

where $a_j \in \mathbb{R}_+$ for each $j \in N \cup \{0\}$. Let a CG cut for $\mathcal{X}_G$ be given, which is described as

$$\sum_{i \in I} \sum_{j \in K_i} \lfloor \hat{\theta}_0 a_j + \hat{\theta}_i \rfloor x_j \leq \left\lfloor \hat{\theta}_0 a_0 + \sum_{i \in I} \hat{\theta}_i \right\rfloor, \tag{3.24}$$

for some $(\hat{\theta}_0, \hat{\theta}) \in \mathbb{R}_+^{m+1}$.

Proposition 3.11. *If the CG cut* (3.1) *is non-dominated, then* $\hat{\theta}_i < 1$ *for each* $i \in I$ *and* $\frac{1}{2} \leq f_0 < 1$ *where* $f_0 = \hat{\theta}_0 a_0 + \sum_{i \in I} \theta_i - \lfloor \hat{\theta}_0 a_0 + \sum_{i \in I} \hat{\theta}_i \rfloor$.

Proof. Suppose that there exists $k \in I$ such that $\hat{\theta}_k \geq 1$. We define $\theta_0^\star = \hat{\theta}_0$ and $\theta_i^\star = \hat{\theta}_i - \lfloor \hat{\theta}_i \rfloor$ for each $i \in I$. Then, the CG cut (3.24) can be represented as a non-negative linear combination of the CG cut defined with $(\theta_0^\star, \theta^\star)$ and $\sum_{j \in K_k} x_j \leq 1$. Therefore, the CG cut (3.24) is dominated by the CG cut defined with $(\theta_0^\star, \theta^\star)$, which contradicts the assumption that (3.24) is non-dominated. The proof for the condition of f_0 is equivalent to the proof of Proposition 3.2. Therefore, the result follows. □

Now, let $\alpha^T x \leq \alpha_0$ be the CG cut (3.24), and we define the following linear program,

$$\max\{\alpha^T x : x \in \mathcal{P}_G\}, \tag{3.25}$$

where $\mathcal{P}_G$ is the linear relaxation of $\mathcal{X}_G$. Then, for the above problem, there exists an optimal solution $x^\star$ that has at most two fractional variables, l_1 and l_2 where $\{l_1, l_2\} \in K_l \cup \{0\}$ for some $l \in I$ and $a_{l_1} \geq a_{l_2}$ with $a_0 = 0$ (Johnson & Padberg, 1981). We note that $l_1 = 0$ implies that the optimal solution has one fractional variable. Let $I^+ = \{l\} \cup \{i \in I : \sum_{j \in K_i} x_j^\star > 0\}$. Additionally, for each $i \in I^+ \setminus \{l\}$, we define $c(i) \in K_i$ as the variable index such that $x_{c(i)}^\star = 1$ while $c(l) = l_2$. Such l, I^+

and $c(i)$'s can be identified in $O(n \log n)$ (Glover & Klingman, 1979). Furthermore, $\sum_{i \in I^+} a_{c(i)} > a_0$ while $\sum_{i \in I^+} a_{c(i)} - a_{l_2} \leq a_0$.

Proposition 3.12. *If the CG cut $\alpha^T x \leq \alpha_0$ is non-dominated, then there exists $(\theta_0^\star, \theta^\star) \in \mathbb{R}_+^{m+1}$, which yields $\alpha^T x \leq \alpha_0$, such that*

1. $\theta_0^\star = (\alpha_{l_2} - \alpha_{l_1})/(a_{l_2} - a_{l_1})$.

2. $\theta_i^\star = \lceil \theta_0^\star a_{c(i)} \rceil - \theta_0^\star a_{c(i)}$ *for each* $i \in I^+$ *and* $\theta_i^\star = 0$ *for each* $i \in I \setminus I^+$.

3. $\theta_i^\star \leq f_0^\star$ *for each* $i \in I^+$ *such that* $\lceil \theta_0^\star a_{c(i)} \rceil = 1$.

where $f_0^\star = \theta_0^\star a_0 + \sum_{i \in I} \theta_i^\star - \lfloor \theta_0^\star a_0 + \sum_{i \in I} \theta_i^\star \rfloor$.

Proof. Let us consider the dual problem of the linear program (3.25), which is represented as follows.

$$\begin{aligned} \min \quad & \theta_0 a_0 + \sum_{i \in I} \theta_i \\ \text{s.t} \quad & \alpha_j \leq \theta_0 a_j + \theta_i, \ j \in K_i, \ \forall i \in I \\ & (\theta_0, \theta) \in \mathbb{R}_+^{m+1} \end{aligned}$$

By the definition of α, $(\hat{\theta}_0, \hat{\theta})$ is a feasible solution for the above problem. Glover & Klingman (1979) showed that there exists an optimal solution $(\theta_0^\star, \theta^\star)$ to the above problem defined as $\theta_0^\star = (\alpha_{l_2} - \alpha_{l_1})/(a_{l_2} - a_{l_1})$ and

$$\theta_i^\star = \begin{cases} \alpha_{c(i)} - \theta_0^\star a_{c(i)} & , \ i \in I^+ \\ 0 & , \ i \in I \setminus I^+ \end{cases}.$$

Let $\sum_{j\in N}\alpha_j^\star x_j \le \alpha_0^\star$ be the CG cut defined with $(\theta_0^\star, \theta^\star)$. Since $(\theta_0^\star, \theta^\star)$ is optimal for the above problem, it is clear that $\alpha_j \le \alpha_j^\star$ for each $j \in N$ and $\alpha_0 \ge \alpha_0^\star$. If there exists $k \in N$ such that $\alpha_k < \alpha_k^\star$, or $\alpha_0 > \alpha_0^\star$, then $\sum_{j\in N}\alpha_j^\star x_j \le \alpha_0^\star$ dominates $\sum_{j\in N}\alpha_j x_j \le \alpha_0$, which conflicts to the assumption that $\sum_{j\in N}\alpha_j x_j \le \alpha_0$ is non-dominated. Therefore, $\alpha_j = \alpha_j^\star$ for each $j \in N$ and $\alpha_0 = \alpha_0^\star$.

By the definition of $(\theta_0^\star, \theta^\star)$, $\alpha_{c(i)} = \theta_0^\star a_c(i) + \theta_i^\star$ for each $i \in I^+$. If there exists $k \in I^+$ such that $\theta_k^\star \ge 1$, the CG cut $\sum_{j\in N}\alpha_j x_j \le \alpha_0$ is a dominated one by Proposition 3.11. Therefore, for each $i \in I^+$, $\theta_i^\star$ should be less than 1, which implies that $\theta_0^\star a_{c(i)} \le \alpha_{c(i)} < \theta_0^\star a_{c(i)} + 1$. Since $\alpha_{c(i)}$'s are integers, $\alpha_{c(i)} = \lceil \theta_0^\star a_{c(i)} \rceil$, that is, $\theta_i^\star = \lceil \theta_0^\star a_{c(i)} \rceil - \theta_0^\star a_{c(i)}$ for each $i \in I^+$.

Now, suppose that there exists $k \in I^+$ such that $\alpha_{c(k)} = 1$ and $\theta_k^\star > f_0^\star$. Then,

$$\lfloor \alpha_0 + f_0^\star - \theta_k^\star \rfloor = \alpha_0 - 1.$$

If $\theta_k^\star$ is replaced with 0, the CG cut defined with $(\theta_0^\star, \theta^\star)$ can be described as follows.

$$\sum_{i\in I\setminus\{k\}}\sum_{j\in K_i}\alpha_j x_j + \sum_{j\in K_k}\lfloor \theta_0^\star a_j \rfloor x_j \le \alpha_0 - 1 \tag{3.26}$$

The CG cut (3.26) induces another inequality through summation with $\sum_{j\in K_k} x_j \le 1$, which is described as

$$\sum_{i\in I\setminus\{k\}}\sum_{j\in K_i}\alpha_j + \sum_{j\in K_k}(\lfloor \theta_0^\star a_j \rfloor + 1)x_j \le \alpha_0. \tag{3.27}$$

Then, $\alpha^T x \le \alpha_0$ is dominated by the inequality (3.27) because $\lfloor \theta_0^\star a_j \rfloor \le \alpha_j \le \lfloor \theta_0^\star a_j \rfloor + 1$ for each $j \in K_k$, while the CG cut (3.26) dominates the inequality (3.27).

This result contradicts the assumption that $\alpha^T x \leq \alpha_0$ is non-dominated. Therefore, the result follows. □

From Proposition 3.11 and 3.12, non-dominated CG cuts for the GKP polytope, $\alpha^T x \leq \alpha_0$, can be rewritten as

$$\sum_{i \in I^+} \sum_{j \in K_i} \lfloor \theta_0^\star a_j + \theta_i^\star \rfloor x_j + \sum_{i \in I \setminus I^+} \sum_{j \in K_i} \lfloor \theta_0^\star a_j \rfloor \leq \left\lfloor \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star \right\rfloor, \tag{3.28}$$

with some $\theta_0^\star \in \mathbb{R}_+$, $I^+ \subseteq I$, and $c(i) \in K_i$ for each K_i such that $\sum_{i \in I^+} a_{c(i)} > a_0$, where $\theta_i^\star = \lceil \theta_0^\star a_{c(i)} \rceil - \theta_0^\star a_{c(i)}$ for each $i \in I^+$, and $\theta_i^\star = 0$ for each $i \in I \setminus I^+$.

Now, let us define maximal CG cuts for the GKP polytope.

Definition 3.2. *The inequality of the form* (3.28) *is called a maximal CG cut for* $\mathcal{X}_G$ *if*

1. $I^+ \subseteq I$ *and* $c(i) \in K_i$ *for each* $i \in I^+$ *such that* $\sum_{i \in I^+} a_{c(i)} > a_0$.
2. $\theta_0^\star = p/(a_{l_2} - a_{l_1})$ *for some* $p \in \mathbb{Z}_+$ *and* $\{l_1, l_2\} \subseteq K_l \cup \{0\}$ *where* $l \in I^+$ *and* $c(l) = l_2$. *In addition,* $\theta_i^\star = \lceil \theta_0^\star a_{c(i)} \rceil - \theta_0^\star a_{c(i)}$ *for each* $i \in I^+$ *while* $\theta_j^\star = 0$ *for each* $i \in I \setminus I^+$.
3. $\frac{1}{2} \leq f_0^\star$ *and* $\theta_i^\star \leq f_0^\star$ *if* $\lceil \theta_0^\star a_{c(i)} \rceil = 1$ *for each* $i \in I^+$.

where $f_0^\star = \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star - \lfloor \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star \rfloor$.

We note that the maximal CG cuts are sufficient to describe the Chvátal closure of $\mathcal{P}_G$.

A maximal CG cut can be obtained efficiently for a given CG cut for the GKP polytope, which is at least as strong as the given one. Let $\alpha^T x \leq \alpha_0$ be the given CG

cut. By solving the linear program (3.25), the optimal solution $x^\star$ can be obtained in $O(n \log n)$ with $I^+ \subseteq I$, $l \in I^+$, $c(i) \in K_i$ for each $i \in I^+$, and $\{l_1, l_2\} \subseteq K_l \cup \{0\}$ where $\sum_{i \in I^+} a_{c(i)} > a_0$. Then, we define $(\bar{\theta}_0, \bar{\theta})$ as $\bar{\theta} = (\alpha_{l_2} - \alpha_{l_1})/(a_{l_2} - a_{l_1})$ and

$$\bar{\theta}_i = \begin{cases} \lceil \bar{\theta}_0 a_{c(i)} \rceil - \bar{\theta}_0 a_{c(i)} & , \ i \in I^+ \\ 0 & , \ i \in I \setminus I^+ \end{cases},$$

where $\alpha_0 = a_0 = 0$. Additionally, a CG cut $\sum_{j \in N} \bar{\alpha}_j x_j \leq \bar{\alpha}_0$ can be defined with $(\bar{\theta}_0, \bar{\theta})$, which is at least as strong as the given CG cut by the proof of Proposition 3.12. Furthermore, the CG cut satisfies the first and second conditions in Definition 3.2.

Let $\bar{f}_0 = \bar{\theta}_0 a_0 + \sum_{i \in I^+} \bar{\theta}_i - \lfloor \bar{\theta}_0 a_0 + \sum_{i \in I^+} \bar{\theta}_i \rfloor$. If $0 < \bar{f}_0 \leq \frac{1}{2}$, we define $(\theta_0^\star, \theta^\star)$ as $\theta_0^\star = t\bar{\theta}_0$ and $\theta_j^\star = t\bar{\theta}_j - \lfloor t\bar{\theta}_j \rfloor$ for each $j \in N$ where

$$t = \left\lceil \frac{1}{2\bar{f}_0} \right\rceil.$$

Subsequently, let $\sum_{j \in N} \alpha_j^\star x_j \leq \alpha_0^\star$ be the CG cut defined with $(\theta_0^\star, \theta^\star)$. Then, the CG cut is at least as strong as $\sum_{j \in N} \bar{\alpha}_j x_j \leq \bar{\alpha}_0$ by the proof of Proposition 3.11, while it still satisfies the first and second conditions in Definition 3.2.

Now, suppose that there exists $k \in I^+$ such that $\alpha_{c(k)}^\star = 1$ and $\theta_k^\star > f_0^\star$ where $f_0^\star = \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star - \lfloor \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star \rfloor$.

Proposition 3.13. *$k \neq l$ and $\sum_{i \in I^+ \setminus \{k\}} a_{c(i)} > a_0$.*

Proof. If $k = l$, then $\alpha_{c(l)}^\star = \alpha_{l_2}^\star = 1$. Hence, $\theta_l^\star = 1 - \theta_0^\star a_{l_2}$. Let $\lambda_G = \sum_{i \in I^+} a_{c(i)} - a_0 > 0$. By the construction of I^+ and l, $\lambda_G \leq a_{l_2}$. On the other hand, $f_0^\star$ can be

represented with λ_G. By definition,

$$\begin{aligned} f_0^\star &= \theta_0^\star a_0 + \sum_{i\in I^+} \theta_i^\star - \left\lfloor \theta_0^\star a_0 + \sum_{i\in I^+} \theta_i^\star \right\rfloor \\ &= \sum_{i\in I^+} \theta_0^\star a_{c(i)} - \theta_0^\star \lambda_G + \sum_{i\in I^+} \theta_i^\star - \left\lfloor \sum_{i\in I^+} \theta_0^\star a_{c(i)} - \theta_0^\star \lambda_G + \sum_{i\in I^+} \theta_i^\star \right\rfloor \\ &= -\theta_0^\star \lambda_G - \lfloor -\theta_0^\star \lambda_G \rfloor \\ &= \lceil \theta_0^\star \lambda_G \rceil - \theta_0^\star \lambda_G \end{aligned}$$

Because $0 < \lambda_G \leq a_{l_2}$ and $0 < \lceil \theta_0^\star \lambda_G \rceil \leq \lceil \theta_0^\star a_{l_2} \rceil = 1$, $f_0^\star = 1 - \theta_0^\star \lambda_G$ while $\theta_l^\star = 1 - \theta_0^\star a_{l_2}$. This result implies that $\theta_l^\star \leq f_0^\star$, hence, $k \neq l$.

Now, we show that $\theta_0^\star(\sum_{i\in I^+\setminus\{k\}} a_{c(i)} - a_0) > 0$ if $k \neq l$. Using the definition of λ_G,

$$\theta_0^\star \left(\sum_{i\in I^+\setminus\{k\}} a_{c(i)} - a_0 \right) = \theta_0^\star(\lambda_G - a_{c(k)}) = \lceil \theta_0^\star \lambda_G \rceil - f_0^\star - (\alpha_{c(k)}^\star - \theta_k^\star)$$

Because $\lceil \theta_0^\star \lambda_G \rceil \geq 1$ while $\alpha_{c(k)}^\star = 1$ and $\theta_k^\star - f_0^\star > 0$ by the assumption,

$$\theta_0^\star \left(\sum_{i\in I^+\setminus\{k\}} a_{c(i)} - a_0 \right) = \lceil \theta_0^\star \lambda_G \rceil - 1 + \theta_k^\star - f_0^\star > 0.$$

Therefore, the result follows. □

Let us replace $\theta_k^\star$ with 0. Then the CG cut $\sum_{j\in N} \alpha_j^\star x_j \leq \alpha_0^\star$ is dominated by the CG cut defined with the modified $(\theta_0^\star, \theta^\star)$ as shown in the proof of Proposition 3.12. We replace I^+ and $\sum_{j\in N} \alpha_j^\star x_j \leq \alpha_0^\star$ with $I^+ \setminus \{k\}$ and the CG cut defined with the modified $(\theta_0^\star, \theta^\star)$, respectively. By repeatedly eliminating $k \in I^+$ that violates

the third condition of Definition 3.2 until no such k exists, we can obtain a maximal CG cut that is at least as strong as the given CG cut. Since one element is removed from I^+ in each iteration, the process terminates within at most $O(m)$ iterations.

We describe the overall algorithm to derive the maximal CG cut for the GKP polytope from the given one in Algorithm 9.

Algorithm 9 Construct a maximal CG cut for GKP polytope from the given one

procedure MAXIMAL$_G(\hat{\theta}_0, \hat{\theta})$
  $\alpha_0 \leftarrow \hat{\theta}_0 a_0 + \sum_{i \in I} \hat{\theta}_i$, $a_j \leftarrow \lfloor \hat{\theta}_0 a_j + \hat{\theta}_i \rfloor$ for each $j \in K_i$ and $i \in I$;
  $\{c(i)\}_{i \in I^+}, \{l_1, l_2\} \in K_l \leftarrow$solve $\max\{\alpha^T x : x \in \mathcal{P}_G\}$;
  $\theta_0^\star \leftarrow (\alpha_{l_2} - \alpha_{l_1})/(a_{l_2} - a_{l_1})$, $\theta_i^\star \leftarrow \lceil \theta_0^\star a_{c(i)} \rceil - \theta_0^\star a_{c(i)}$ for each $i \in I^+$;
  $f_0^\star \leftarrow \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star - \lfloor \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star \rfloor$;
  if $f_0^\star < \frac{1}{2}$ **then**
    $t \leftarrow \lceil 1/(2 f_0^\star) \rceil$;
    $\theta_0^\star \leftarrow t\theta_0^\star$, $\theta_i^\star \leftarrow t\theta_i^\star - \lfloor t\theta_i^\star \rfloor$ for each $i \in I^+$;
  end if
  repeat
    $f_0^\star \leftarrow \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star - \lfloor \theta_0^\star a_0 + \sum_{i \in I^+} \theta_i^\star \rfloor$;
    Find $k \in I^+$ such that $\lceil \theta_0^\star a_{c(k)} \rceil = 1$ and $\theta_k^\star > f_0^\star$;
    $I^+ \leftarrow I^+ \setminus \{k\}$, $\theta_k^\star \leftarrow 0$;
  until $\nexists k$;
  return $(\theta_0^\star, \theta^\star)$;
end procedure

The maximal CG cut for the GKP polytope can also be obtained in $O(n \log n)$. In the following section, we present a method further to strengthen maximal CG cuts for the GKP polytope.

3.5.2 Strengthening maximal CG cuts for GKP polytopes

We first introduce the surrogate knapsack polytope, which bridges GKP polytopes and binary knapsack polytopes. A given maximal CG cut for a GKP polytope can be interpreted as a CG cut for the surrogate knapsack polytope. Then, we apply the CG cut strengthening method for binary knapsack polytopes to the interpreted CG

cut.

Let $\beta^T x \leq \beta_0$ be a given maximal CG cut for $\mathcal{X}_G$, which is defined with (θ_0, θ) for some $\theta_0 \in \mathbb{R}_+$, $I^+ \subseteq I$, $c(i) \in K_i$ for each $i \in I^+$, $l \in I$, and $\{l_1, l_2\} \subseteq K_l \cup \{0\}$, where $\theta_0(a_{l_2} - a_{l_1})$ is an integer, $\theta_i = \lceil \theta_0 a_{c(i)} \rceil - \theta_0 a_{c(i)}$ for each $i \in I^+$, $\theta_i = 0$ for each $i \in I \setminus I^+$, and $\sum_{i \in I^+} a_{c(i)} > a_0$ while $\sum_{i \in I^+} a_{c(i)} - a_{l_1} \leq a_0$. Additionally, let $\lambda_G = \sum_{i \in I^+} a_{c(i)} - a_0$ and $f_0 = \theta_0 a_0 + \sum_{i \in I^+} \theta_i - \lfloor \theta_0 a_0 + \sum_{i \in I^+} \theta_i \rfloor$. Then, $\theta_i \leq f_0$ for each $i \in I^+$ such that $\beta_{c(i)} = 1$ due to Proposition 3.12. The maximal CG cut can be rewritten as

$$\sum_{i \in I^+} \sum_{j \in K_i} \lfloor \theta_0 a_j + \theta_i \rfloor x_j + \sum_{i \in I \setminus I^+} \sum_{j \in K_i} \lfloor \theta_0 a_j \rfloor x_j \leq \left\lfloor \theta_0 a_0 + \sum_{i \in I^+} \theta_i \right\rfloor \tag{3.29}$$

Let $L_i = \{j \in K_i \setminus \{c(i)\} : a_j \leq a_{c(i)}\}$ and $U_i = K_i \setminus (L_i \cup \{c(i)\})$ for each $i \in I^+$. The surrogate knapsack polytope, $\mathcal{X}_K$, is defined as

$$\mathcal{X}_K = \left\{ y \in \{0,1\}^n : \sum_{j \in N} \bar{a}_j y_j \leq \bar{a}_0 \right\},$$

where

$$\bar{a}_j = \begin{cases} a_{c(i)} - a_j & \text{, if } j \in L_i \text{ for some } i \in I^+ \\ a_j - a_{c(i)} & \text{, if } j \in U_i \text{ for some } i \in I^+ \\ a_j & \text{, otherwise.} \end{cases},$$

and $\bar{a}_0 = a_0 + \sum_{i \in I^+} \sum_{j \in L_i} \bar{a}_j$. We note that the surrogate knapsack polytope is a binary knapsack polytope.

Proposition 3.14. *Suppose that the following inequality is valid for $\mathcal{X}_K$.*

$$\sum_{j\in N}\gamma_j y_j \le \gamma_0 \tag{3.30}$$

Then, the inequality

$$\sum_{i\in I^+}\sum_{j\in L_i}(\gamma_{c(i)}-\gamma_j)x_j + \sum_{i\in I^+}\gamma_{c(i)}x_{c(i)} + \sum_{i\in I^+}\sum_{j\in U_i}(\gamma_{c(i)}+\gamma_j)x_j + \sum_{i\in I\setminus I^+}\sum_{j\in K_i}\gamma_j x_j \le \gamma_0 - \sum_{i\in I^+}\sum_{j\in L_i}\gamma_j \tag{3.31}$$

is valid for $\mathcal{X}_G$.

Proof. Using the definition of $\mathcal{X}_K$, a feasible solution $\hat{x}$ for $\mathcal{X}_G$ can be transformed into $\bar{y}(\hat{x}) \in \mathcal{X}_K$, where

$$\bar{y}_j(\hat{x}) = \begin{cases} 1-\hat{x}_j & \text{, if } j \in L_i \text{ for some } i \in I^+ \\ \sum_{l\in K_i}\hat{x}_l & \text{, if } j = c(i) \text{ for some } i \in I^+ \\ \hat{x}_j & \text{, otherwise.} \end{cases},$$

Let $\mathcal{Y}(\mathcal{X}_G) = \{\bar{y}(x) : x \in \mathcal{X}_G\}$. Because $\mathcal{Y}(\mathcal{X}_G) \subseteq \mathcal{X}_K$,

$$\sum_{j\in N}\gamma_j\bar{y}_j(\hat{x}) \le \gamma_0,\ \forall \hat{x} \in \mathcal{X}_G.$$

By replacing $\bar{y}(\hat{x})$ with $\hat{x}$, we can see that the inequality (3.31) is valid for all $\hat{x} \in \mathcal{X}_G$. Therefore, the result follows. □

We call the inequality (3.31) as the *G-inequality* for the valid inequality for $\mathcal{X}_K$. It can be easily shown that the dominance between valid inequalities for $\mathcal{X}_K$ is

preserved in their G-inequalities.

The given maximal CG cut for $\mathcal{X}_G$ can be interpreted as the G-inequality of a CG cut for $\mathcal{X}_K$. Let $C_G = \{j \in L_i \cup \{c(i)\} : i \in I^+\}$.

Proposition 3.15. *The G-inequality of the following CG cut for $\mathcal{X}_K$ is equivalent to the given maximal CG cut* (3.29) *for $\mathcal{X}_G$.*

$$\sum_{j \in C_G} \lfloor \theta_0 \bar{a}_j + v_j \rfloor y_j + \sum_{N \setminus C_G} \lfloor \theta_0 \bar{a}_j \rfloor y_j \leq \left\lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j \right\rfloor, \tag{3.32}$$

where

$$v_j = \begin{cases} \lceil \theta_0 \bar{a}_j \rceil - \theta_0 \bar{a}_j & , \ j \in L_i \text{ for some } i \in I^+ \\ \theta_i & , \ j = c(i) \text{ for some } i \in I^+ \\ 0 & , \text{ otherwise.} \end{cases} .$$

Proof. It is clear that the coefficients of $x_{c(i)}$'s in the G-inequality are equivalent to (3.29). In addition, the coefficients of x_j's such that $j \in K_i$ for each $i \in I \setminus I^+$ are equivalent to (3.29) because $\bar{a}_j = a_j$. For each $j \in L_i$ and $i \in I^+$, the coefficient of x_j in the G-inequality is

$$\lfloor \theta_0 \bar{a}_{c(i)} + \theta_i \rfloor - \lceil \theta_0 \bar{a}_j \rceil.$$

Because $\theta_0 \bar{a}_{c(i)} + \theta_i \in \mathbb{Z}_+$ by the definition of θ_i, $\bar{a}_{c(i)} = a_{c(i)}$, and $\bar{a}_j = a_{c(i)} - a_j$, the coefficient can be rewritten as

$$\theta_0 a_{c(i)} + \theta_i + \lfloor \theta_0 (a_j - a_{c(i)}) \rfloor = \lfloor \theta_0 a_j + \theta_i \rfloor,$$

which is equivalent to the coefficient in (3.29).

In a similar way, for each $j \in U_i$ and $i \in I^+$, the coefficient of x_j in the G-

inequality is

$$\lfloor \theta_0 \bar{a}_{c(i)} + \theta_i \rfloor + \lfloor \theta_0 \bar{a}_j \rfloor,$$

and it can be rewritten as

$$\theta_0 a_{c(i)} + \theta_i + \lfloor \theta_0 (a_j - a_{c(i)}) \rfloor = \lfloor \theta_0 a_j + \theta_i \rfloor,$$

which is equivalent to the coefficient in (3.29).

Finally, we show that

$$\left\lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j \right\rfloor - \sum_{i \in I^+} \sum_{j \in L_i} \lceil \theta_0 \bar{a}_j \rceil = \left\lfloor \theta_0 \bar{a}_0 + \sum_{i \in I^+} \theta_i \right\rfloor.$$

The right-hand side of the CG cut (3.32) can be rewritten as follows.

$$\begin{aligned} \left\lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j \right\rfloor &= \left\lfloor \theta_0 \left(a_0 + \sum_{i \in I^+} \sum_{j \in L_i} \bar{a}_j \right) + \sum_{i \in I^+} \sum_{j \in L_i} (\lceil \theta_0 \bar{a}_j \rceil - \theta_0 \bar{a}_j) + \sum_{i \in I^+} \theta_i \right\rfloor \\ &= \left\lfloor \theta_0 a_0 + \sum_{i \in I^+} \theta_i \right\rfloor + \sum_{i \in I^+} \sum_{j \in L_i} \lceil \theta_0 \bar{a}_j \rceil \end{aligned}$$

Then,

$$\left\lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j \right\rfloor - \sum_{i \in I^+} \sum_{j \in L_i} \lceil \theta_0 \bar{a}_j \rceil = \left\lfloor \theta_0 \bar{a}_0 + \sum_{i \in I^+} \theta_i \right\rfloor.$$

Therefore, the result follows. □

Proposition 3.15 implies that the given maximal CG cut for $\mathcal{X}_G$ can be obtained from the CG cut (3.32). Furthermore, a cut that dominates the CG cut (3.32) can yield a G-inequality stronger than the given maximal CG cut for the GKP polytope. Such a stronger cut can be obtained by applying the CG cut strengthening method,

presented in Section 3.4, to (3.32) if it is a maximal CG cut for $\mathcal{X}_K$. However, the CG cut (3.32) may not be maximal. Recall that $f_0 = \theta_0 a_0 + \sum_{i \in I^+} \theta_i - \lfloor \theta_0 a_0 + \sum_{i \in I^+} \theta_i \rfloor$. We additionally define $f_j = \theta_0 a_j + \theta_i - \lfloor \theta_0 a_j + \theta_j \rfloor$ for each $j \in K_i$ and $i \in I$.

Proposition 3.16. *The CG cut* (3.32) *is a maximal CG cut for* $\mathcal{X}_K$ *if* $f_j \leq f_0$ *for all* $j \in L_i$ *and* $i \in I^+$.

Proof. By the definition of v_j's, the CG cut (3.32) can be rewritten as follows.

$$\sum_{j \in C_G} \lceil \theta_0 \bar{a}_j \rceil y_j + \sum_{j \in C_G} \lfloor \theta_0 \bar{a}_j \rfloor y_j \leq \left\lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j \right\rfloor$$

We first show that C_G is a cover for $\mathcal{X}_K$. By the definition of C_G,

$$\sum_{j \in C_G} \bar{a}_j = \sum_{i \in I^+} \sum_{j \in L_i} \bar{a}_j + \sum_{i \in I^+} a_{c(i)} > \sum_{i \in I^+} \sum_{j \in L_i} \bar{a}_j + a_0 = \bar{a}_0.$$

Therefore, C_G is a cover for $\mathcal{X}_K$. In addition, since $a_{l_2} - a_{l_1} = \bar{a}_{l_1}$ and $l_1 \in C_G$, we can rewrite θ_0 as $(\beta_{l_2} - \beta_{l_1})/\bar{a}_{l_1}$. Hence, the second condition in Definition 3.1 is satisfied.

Let $\bar{f}_0 = \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j - \lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j \rfloor$. Now, we show that v_j's and $\bar{f}_0$ satisfy the third condition in Definition 3.1. As shown in the proof of Proposition 3.15, $\bar{f}_0 = f_0$, that is, $\frac{1}{2} \leq \bar{f}_0$. Furthermore, for each $j \in L_i$ and $i \in I^+$,

$$\begin{aligned} v_j &= \lceil \theta_0 a_{c(i)} - \theta_0 a_j \rceil - \theta_0 a_{c(i)} + \theta_0 a_j \\ &= \theta_0 a_j + \lceil \theta_0 a_{c(i)} \rceil - \theta_0 a_{c(i)} - \lfloor \theta_0 a_j + \lceil \theta_0 a_{c(i)} \rceil - \theta_0 a_{c(i)} \rfloor \\ &= \theta_0 a_j + \theta_i - \lfloor \theta_0 a_j + \theta_i \rfloor = f_j^\star. \end{aligned}$$

Therefore, the assumption that $f_j \leq f_0$ for each $j \in L_i$ and $i \in I^+$ implies $v_j \leq \bar{f}_0$ for each $j \in L_i$ and $i \in I^+$. Recall that (θ_0, θ) satisfies $\theta_i \leq f_0$ for each $i \in I^+$ such that $\lceil \theta_0 a_{c(i)} \rceil = 1$ because it yields the maximal CG cut $\beta^T x \leq \beta_0$ for $\mathcal{X}_G$. Then, $v_{c(i)} = \theta_i \leq f_0 = \bar{f}_0$ for each $i \in I^+$ such that $\lceil \theta_0 \bar{a}_{c(i)} \rceil = 1$ where $\bar{a}_{c(i)} = a_{c(i)}$. Therefore, the result follows. □

Even if the CG cut (3.32) is not a maximal CG cut for $\mathcal{X}_K$, we can obtain the one through the CG cut (3.32) using Algorithm 7. As shown in the proof of Proposition 3.16, the first and second conditions in Definition 3.2 are satisfied without the assumption that $f_j \leq f_0$ for all $j \in L_i$ and $i \in I^+$. This result implies that a maximal CG cut can be constructed from the CG cut (3.32) by eliminating some variables in C_G, which violate the third condition in Definition 3.2. Let $C_G^\star$ be the modified cover. The obtained maximal CG cut for $\mathcal{X}_K$ can be written as

$$\sum_{j \in C_G^\star} \lceil \theta_0 \bar{a}_j \rceil y_j + \sum_{j \in C_G^\star} \lfloor \theta_0 \bar{a}_j \rfloor y_j \leq \left\lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G^\star} v_j \right\rfloor, \tag{3.33}$$

Then, we can obtain a SCG cut for $\mathcal{X}_K$ by applying the CG cut strengthening method proposed in Section 3.4 to the above maximal CG cut.

Theorem 3.5. *The G-inequality of the obtained SCG cut for $\mathcal{X}_K$ derived from* (3.33) *is equivalent to or dominates the given maximal CG cut $\beta^T x \leq \beta_0$ for $\mathcal{X}_G$.*

Proof. The SCG cut is at least as strong as the CG cut (3.33) by Proposition 3.2 while the CG cut (3.33) is equivalent to or dominates the CG cut (3.32) whose G-inequality is the given maximal CG cut for $\mathcal{X}_G$. Because the dominance between valid inequalities for $\mathcal{X}_K$ is preserved in their G-inequalities, the G-inequality of

the obtained SCG cut is at least as strong as the given maximal CG cut for $\mathcal{X}_G$. Therefore, the result follows. □

Additionally, we compare the G-inequality of the obtained cut with the Gomory mixed-integer cut for $\mathcal{X}_G$ derived from the following inequality.

$$\sum_{i\in I^+}\sum_{j\in K_i}(\theta_0 a_j+\theta_i)x_j+\sum_{i\in I\setminus I^+}\sum_{j\in K_i}\theta_0 a_j x_j\le \theta_0 a_0+\sum_{i\in I^\star}\theta_i$$

Then, the Gomory mixed-integer cut is represented as follows.

$$\sum_{i\in I^+}\sum_{j\in K_i}\psi_G^{f_0}\left(a_j+\frac{\theta_i}{\theta_0}\right)x_j+\sum_{i\in I\setminus I^+}\sum_{j\in K_i}\psi_G^{f_0}(a_j)x_j\le\left\lfloor\theta_0 a_0+\sum_{i\in I^+}\theta_i\right\rfloor. \tag{3.34}$$

Theorem 3.6. *The G-inequality of the obtained SCG cut for $\mathcal{X}_K$ derived from* (3.33) *is equivalent to or dominates the Gomory mixed-integer cut* (3.34) *if $f_j\le f_0$ for all $j\in L_i$ and $i\in I^+$.*

Proof. The assumption implies that the CG cut (3.32) is a maximal CG cut for $\mathcal{X}_K$ by Proposition 3.16. In addition, the fractional part of the right-hand side of the CG cut (3.32) is equivalent to f_0.

This observation allows us to define a Gomory mixed-integer cut for $\mathcal{X}_K$ from the following inequality.

$$\sum_{j\in C_G}(\theta_0\bar{a}_j+v_j)y_j+\sum_{j\in N\setminus C_G}\theta_j\bar{a}_j y_j\le\theta_0\bar{a}_0+\sum_{j\in C_G}v_j$$

The corresponding Gomory mixed-integer cut for $\mathcal{X}_K$ is expressed as

$$\sum_{j\in C_G} \lceil \theta_0 \bar{a}_j \rceil y_j + \sum_{j\in N\setminus C_G} \psi_G^{f_0}(\bar{a}_j) y_j \leq \left\lfloor \theta_0 \bar{a}_0 + \sum_{j\in C_G} v_j \right\rfloor. \tag{3.35}$$

Then, the G-inequality of the cut (3.35) is represented as follows.

$$\begin{aligned}\sum_{i\in I^+}\sum_{j\in L_i} (\lceil \theta_0 \bar{a}_{c(i)} \rceil - \lceil \theta_0 \bar{a}_j \rceil) x_j + \sum_{i\in I^+} \lceil \theta_0 \bar{a}_{c(i)} \rceil x_{c(i)} + \sum_{i\in I^+}\sum_{j\in U_i} (\lceil \theta_0 \bar{a}_{c(i)} \rceil + \psi_G^{f_0}(\bar{a}_j)) x_j \\ + \sum_{i\in I\setminus I^+}\sum_{j\in K_i} \psi_G^{f_0}(\bar{a}_j) x_j \leq \left\lfloor \theta_0 \bar{a}_0 + \sum_{j\in C_G} v_j \right\rfloor - \sum_{i\in I^+}\sum_{j\in L_i} \lceil \theta_0 \bar{a}_j \rceil \end{aligned} \tag{3.36}$$

We first show that the inequality (3.36) is equivalent to (3.34) under the assumption that $f_j \leq f_0$ for all $j \in L_i$ and $i \in I^+$. It is clear that the coefficients of variables in K_i's such that $i \in I \setminus I^+$ in both the inequalities are equivalent to $\psi_G^{f_0}(a_j)$ because $\bar{a}_j = a_j$. For each $i \in I^+$, the coefficient of $x_{c(i)}$ in the inequality (3.34) is equivalent to that in (3.36) because

$$\psi_G^{f_0}\left(a_{c(i)} + \frac{\theta_i}{\theta_0}\right) = \lfloor \theta_0 a_{c(i)} + \theta_i \rfloor = \lceil \theta_0 a_{c(i)} \rceil = \lceil \theta_0 \bar{a}_{c(i)} \rceil.$$

For each $j \in L_i$ and $i \in I^+$, the coefficient of x_j in the inequality (3.34) can be rewritten as

$$\psi_G^{f_0}\left(a_j + \frac{\theta_i}{\theta_0}\right) = \lfloor \theta_0 a_j + \theta_i \rfloor,$$

due to the assumption that $f_j \leq f_0$. Then, $\lfloor \theta_0 a_j + \theta_i \rfloor$ is equivalent to the coefficient of x_j in the inequality (3.36) because

$$\lfloor \theta_0 a_j + \theta_i \rfloor = \lfloor \theta_0 a_j + \lceil \theta_0 a_{c(i)} \rceil - \theta_0 a_{c(i)} \rfloor = \lceil \theta_0 a_{c(i)} \rceil - \lceil \theta_0 (a_{c(i)} - a_j) \rceil$$

$$= \lceil \theta_0 \bar{a}_{c(i)} \rceil - \lceil \theta_0 \bar{a}_j \rceil$$

For each variable in U_i for each $i \in I^+$, the coefficient in the inequality (3.36) can be rewritten as

$$\theta_0 a_{c(i)} + \theta_i + \psi_G^{f_0}(\bar{a}_j) = \psi_G^{f_0}\left(a_{c(i)} + \bar{a}_j + \frac{\theta_i}{\theta_0}\right) = \psi_G^{f_0}\left(a_j + \frac{\theta_i}{\theta_0}\right),$$

because $\theta_0 a_{c(i)} + \theta_i$ is an integer. Therefore, the coefficients of x_j in both inequalities are equivalent. Finally, the right-hand side of the inequality (3.36) is

$$\begin{aligned}
&\left\lfloor \theta_0 \bar{a}_0 + \sum_{j \in C_G} v_j \right\rfloor - \sum_{i \in I^+} \sum_{j \in L_i} \lceil \theta_0 \bar{a}_j \rceil \\
&= \left\lfloor \theta_0 \left(a_0 - \sum_{i \in I^+} \sum_{j \in L_i} \bar{a}_j \right) + \sum_{i \in I^+} \theta_i + \sum_{i \in I^+} \sum_{j \in L_i} (\lceil \theta_0 \bar{a}_j \rceil - \theta_0 \bar{a}_j) \right\rfloor - \sum_{i \in I^+} \sum_{j \in L_i} \lceil \theta_0 \bar{a}_j \rceil \\
&= \left\lfloor \theta_0 a_0 + \sum_{i \in I^+} \theta_i \right\rfloor
\end{aligned}$$

Therefore, the inequality (3.36) is equivalent to (3.34).

By Theorem 3.4, the obtained SCG cut is at least as strong as the inequality (3.35). Because the G-inequality of (3.35) is equivalent to the Gomory mixed-integer cut (3.34), the G-inequality of the obtained SCG cut is at least as strong as the Gomory mixed-integer cut (3.34). □

3.6 Computational test results

This section presents the computational test results for evaluating the performance of the proposed CG cut strengthening methods. Our methods generate a SCG cut

from a maximal CG cut for binary knapsack or GKP polytopes, where the maximal CG cut is derived from a given CG cut. As discussed in Section 3.3 and 3.5, the maximal CG cut can already be stronger than the given CG cut. Hence, the effect of SCG cuts may seem to originate from the effect of maximal CG cuts. Therefore, we also evaluated the performance of the obtained maximal CG cuts to measure the effectiveness of our strengthening methods precisely. Furthermore, the maximal CG cuts can be strengthened to Gomory mixed-integer cuts using the corresponding cut-generating function. We also report the results of Gomory mixed-integer cuts to compare with our strengthening methods.

We used two types of benchmark instances: generalized assignment problem instances from OR-Library (Beasley, 1990) and MIPLIB instances used in Chapter 2. The generalized assignment problem (GAP) is represented as

$$\begin{aligned} \text{(GAP)} \quad \max \quad & \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} \\ \text{s.t} \quad & \sum_{j \in N} a_{ij} x_{ij} \leq b_i, \ \forall i \in M \\ & \sum_{i \in M} x_{ij} \leq 1, \ \forall j \in N \\ & x_{ij} \in \{0, 1\}, \ \forall i \in M, \ \forall j \in N, \end{aligned}$$

where $a_{ij} > 0$ and $c_{ij} > 0$. The feasible solution set of the GAP is defined with $|M|$ different binary knapsack polytopes. For the description of MIPLIB instances, see Section 2.6 in Chapter 2. The GAP and MIPLIB instances are summarized in Appendix A.

For each instance, we utilized the cutting plane algorithm with CG cuts. In

each iteration, CG cuts were generated from binary knapsack or GKP polytopes defined with each constraint. To define GKP polytopes, we employed the GUB set identification strategy proposed by Gu et al. (1998).

We set 300 seconds time limit for the cutting plane algorithm. All algorithms were implemented using C++ with the linear programming solver provided by Xpress (Guéret et al., 2002). All tests were performed using a machine with an Intel Core i7, 3.10GHz CPU, and 16GB RAM.

3.6.1 Effectiveness of SCG cuts derived from binary knapsack polytopes

We evaluated the effectiveness of our strengthening method for maximal CG cuts for binary knapsack polytopes. As mentioned earlier, we defined the binary knapsack polytopes using the constraints in the test instances. A binary knapsack polytope is a special case of a GKP polytope where GUB sets are singletons. Hence, in each iteration of the cutting plane algorithm, we generated CG cuts from the binary knapsack polytopes using the heuristic separation algorithm proposed in Chapter 2. We derived the maximal CG cuts from the generated CG cuts and then strengthened the maximal CG cuts to the SCG cuts or Gomory mixed-integer cuts. Because SCG or Gomory mixed-integer cuts can have fractional coefficients, we multiplied 1,000 by each coefficient and rounded down the coefficients to make the cuts numerically stable.

We denote the results by the cutting plane algorithm using the maximal CG cuts as "MCG" while those by using the SCG cuts and Gomory mixed-integer cuts obtained from the maximal CG cuts are denoted as "SCG" and "GMI", respectively.

For each instance, we measured the changes in the integrality gap closed to evaluate the formulation-enhancing effect of strengthened cuts as follows.

$$\Delta\text{IGC}\ (\%) = \text{IGC} - \text{IGC}_{CG}$$

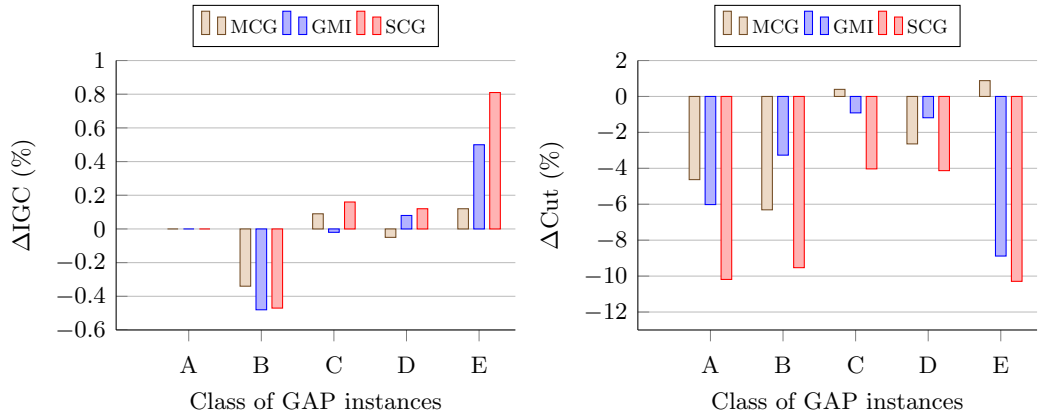
where IGC is the integrality gap improved by the generated cuts using strengthening methods, while $\#\text{IGC}_{CG}$ is the integrality gap improved by CG cuts. The changes in the number of generated cuts and the cutting plane times are measured as the ratio compared to CG cuts, which are denoted as "Δ#Cut (%)" and "ΔTime (%)", respectively. These measures are computed as

$$\Delta\text{Cut}\ (\%) = \frac{\#\text{Cut} - \#\text{Cut}_{CG}}{\#\text{Cut}_{CG}}, \quad \Delta\text{Time}\ (\%) = \frac{\text{Time} - \text{Time}_{CG}}{\text{Time}_{CG}},$$

where #Cut and Time means the number of generated cuts and the cutting plane time in seconds using strengthening methods, respectively, while $\#\text{Cut}_{CG}$ and Time_{CG} are those by CG cuts. We also reported the total time spent on strengthening generated CG cuts (Strengthening time), which we refer to as "STime (s)".

We first present the computational test results for the GAP instances. We set the CG cut separation heuristic parameter as $T = \min\{n, |M|\}$ and adopted variable ordering strategy `S3`. The GAP instances consist of five classes (A, B, C, D, E), and we report the average results for each class. The detailed results are given in Appendix C.1.

Figure 3.4a and 3.4b illustrate the changes in the integrality gap closed and the number of generated cuts, respectively. Although the strengthened cuts are theoretically stronger than the CG cuts, the formulation-enhancing effect utilizing them



(a) Changes in the integrality gap closed (b) Changes in the number of generated cuts

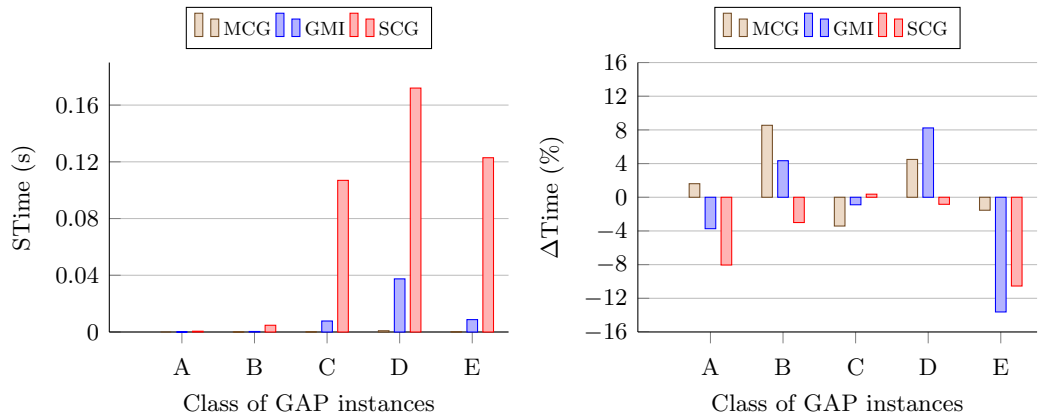
Figure 3.4: ΔIGC (%) and ΔCut (%) by strengthening methods for GAP instances

decreased in several instances, specifically in class B. This result can be explained by the fact that we used the heuristic separation algorithm to generate CG cuts. Even though a CG cut exists that cuts off an incumbent solution obtained after adding a strengthened cut, the heuristic algorithm may fail to identify such a CG cut. Therefore, enhanced formulations obtained using the strengthened cuts may provide weaker relaxations than those by CG cuts.

Although the weakened relaxations were obtained in some instances, the difference was negligible, as shown in Figure 3.4a. Nonetheless, in most instances, the strengthened cuts could yield tighter relaxations compared to the CG cuts. In particular, the SCG cuts could significantly improve the relaxations than the maximal CG cuts. Furthermore, the SCG cuts enhanced the formulations more effectively than the Gomory mixed-integer cuts. These results correspond to our strength comparison result in Section 3.4 and imply that our strengthening method can generate cuts that dominate maximal CG and Gomory mixed-integer cuts.

As shown in Figure 3.4b, using strengthened cuts could significantly reduce the

size of the enhanced formulations in most instances, compared to the CG cuts. Specifically, the SCG cuts could yield the enhanced formulations defined with far fewer cuts than the maximal CG cuts or the Gomory mixed-integer cuts on average.



(a) Strengthening time

(b) Changes in cutting plane time

Figure 3.5: STime (s) and ΔTime (%) by strengthening methods for GAP instances

Figure 3.5a and 3.5b show the strengthening time and the changes in the cutting plane times, respectively. The maximal CG cuts could be obtained from the generated CG cuts in an insignificant amount of computation time, as shown in Figure 3.5a. On the other hand, our strengthening method spends more computation time compared to generating the Gomory mixed-integer cuts because our strengthening method requires $O(n \log n)$ computations. In contrast, $O(n)$ computations are necessary to obtain the Gomory mixed-integer cuts. However, the strengthening time for SCG was less than 0.2 seconds, while the cutting plane algorithms' time limit was 300 seconds. Therefore, the strengthening time was negligible.

Although our strengthening method spent much additional computation time, the cutting plane algorithm using the SCG cuts terminated earlier than when using the CG cuts in most instances. The decrease in the number of generated cuts implies

a reduction in the number of iterations of the cutting plane algorithm, which reduces the cutting plane time.

Figures 3.6, 3.7, and 3.8 illustrate the results for the MIPLIB instances. We reported the instances where at least one cut was obtained through the strengthening methods, which is different from the generated CG cuts. The detailed results are given in Appendix C.2. For the heuristic CG cut separation algorithm, we set the parameter $T = \min\{n, r^\star\}$ and adopted the variable ordering strategy `S3`.

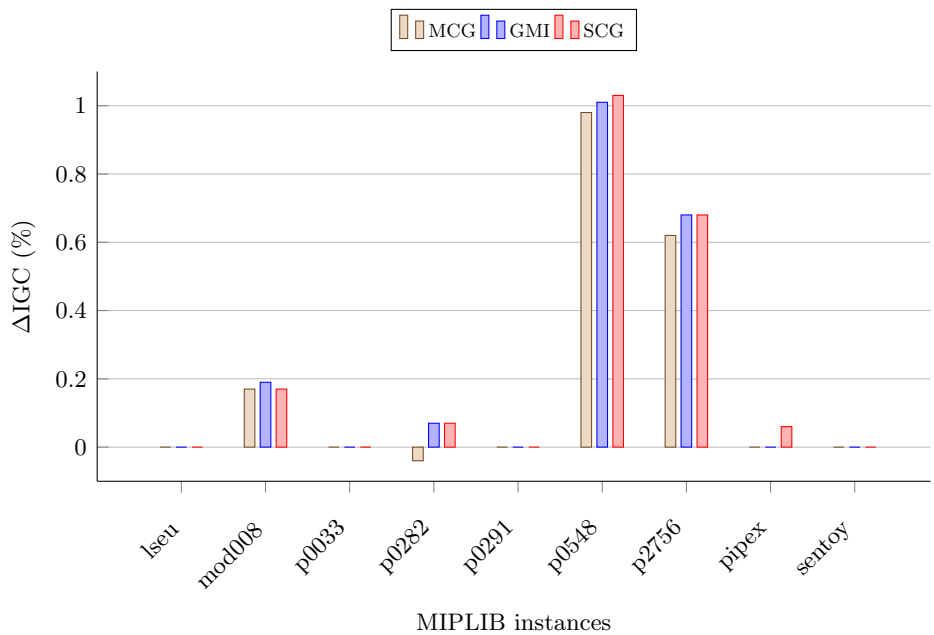


Figure 3.6: ΔIGC (%) by strengthening methods for MIPLIB instances

The effect of our strengthening method for MIPLIB instances was similar to that in GAP instances. The strengthened cuts provided slightly more enhanced formulations than CG cuts, while the SCG cuts yielded tighter relaxations compared to other strengthened cuts. Furthermore, the strengthened cuts provided formulations with fewer cuts compared to CG cuts in most instances. In particular, using the SCG

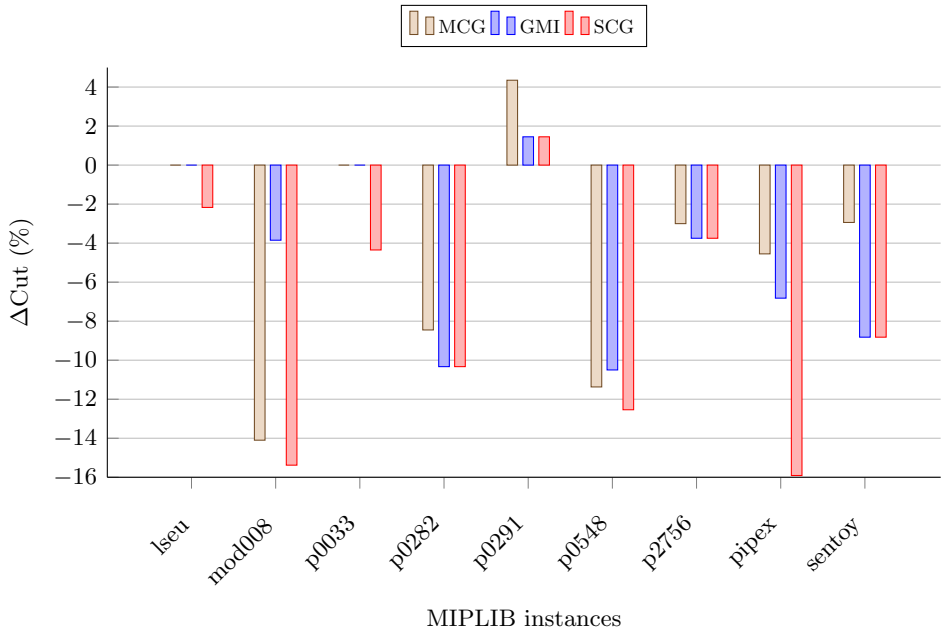


Figure 3.7: ΔCut (%) by strengthening methods for MIPLIB instances

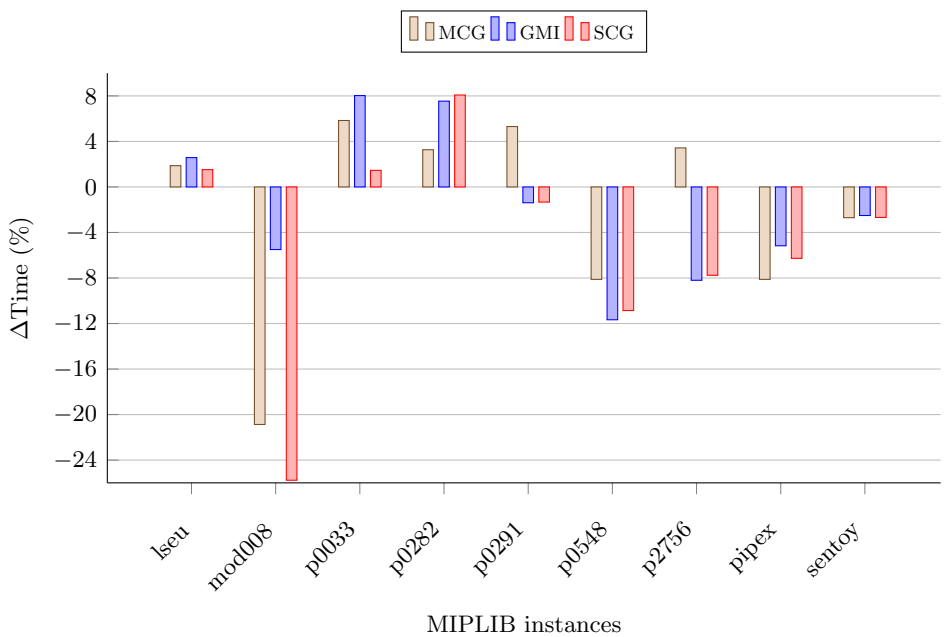


Figure 3.8: ΔTime (%) by strengthening methods for MIPLIB instances

cuts was more effective in reducing the formulation size compared to using the other strengthened cuts. The strengthening times were negligible. As a result, the cutting plane time was decreased in most instances when using the SCG cuts compared to the CG cuts.

In conclusion, our strengthening method provided more enhanced formulations with fewer cuts compared to results using CG cuts. The generated SCG cuts outperformed the maximal CG cuts and Gomory mixed-integer cuts in formulation enhancement and formulation-size reduction. Therefore, the proposed method shows promise for deriving effective cuts. Although the method required additional computation time to generate SCG cuts, it ultimately reduced the cutting plane time due to the decreased number of iterations and generated cuts.

3.6.2 Effectiveness of SCG cuts derived from GKP polytopes

In this section, we evaluated the effectiveness of our strengthening method for maximal CG cuts for GKP polytopes. The GKP polytopes were defined using the constraints of test instances and identified GUB sets by Gu et al. (1998). In each iteration of the cutting plane algorithm, CG cuts were generated from the GKP polytopes using the heuristic separation algorithm proposed in Chapter 2. We derived the maximal CG cuts from the generated CG cuts and then applied our strengthening method to the cuts.

The GKP polytopes defined from GAP instances are equivalent to binary knapsack polytopes. Therefore, we only present the results for MIPLIB instances where at least one strengthened cut is obtained, which differs from the generated CG cuts. We set the parameter for the heuristic CG cut separation algorithm as $T = \min\{m, r^\star\}$,

where m is the number of identified GUB sets, and adopted the variable ordering strategy `S3`.

Figure 3.9, 3.9, and 3.11 illustrate the results for MIPLIB instances. The detailed results are given in Appendix C.3.

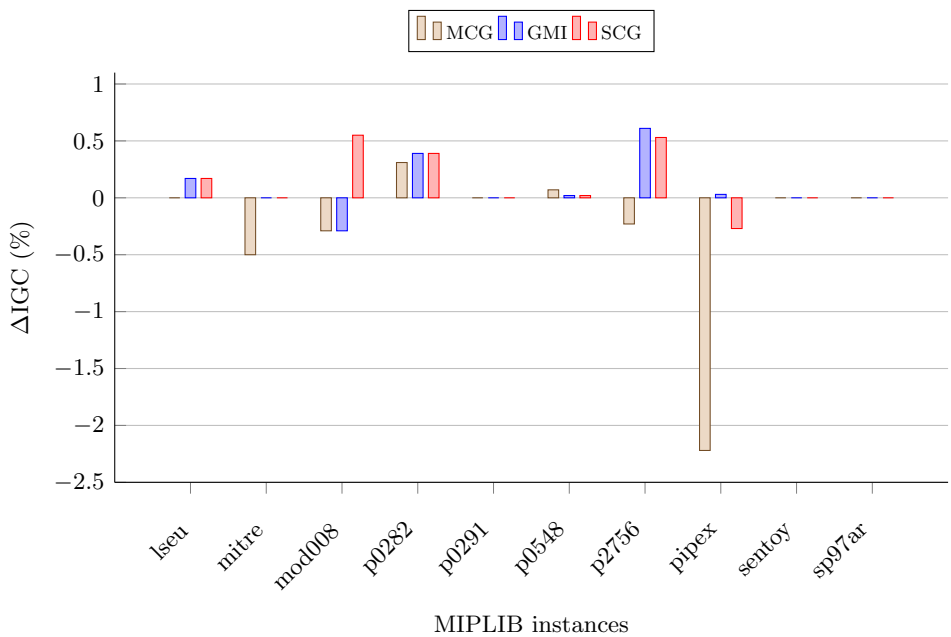


Figure 3.9: ΔIGC (%) by strengthening methods for MIPLIB instances

Figure 3.9 shows that the strengthened cuts yielded more enhanced formulations than CG cuts in most instances. However, the formulation-enhancing effect deteriorated compared to those obtained by strengthened cuts derived from binary knapsack polytopes. Additionally, we could not observe a clear dominance relation between the performance of Gomory mixed-integer and SCG cuts. This result can be explained by the fact that the SCG cuts derived from GKP polytopes cannot always dominate the Gomory mixed-integer cuts in theory, as discussed in Section 3.5.

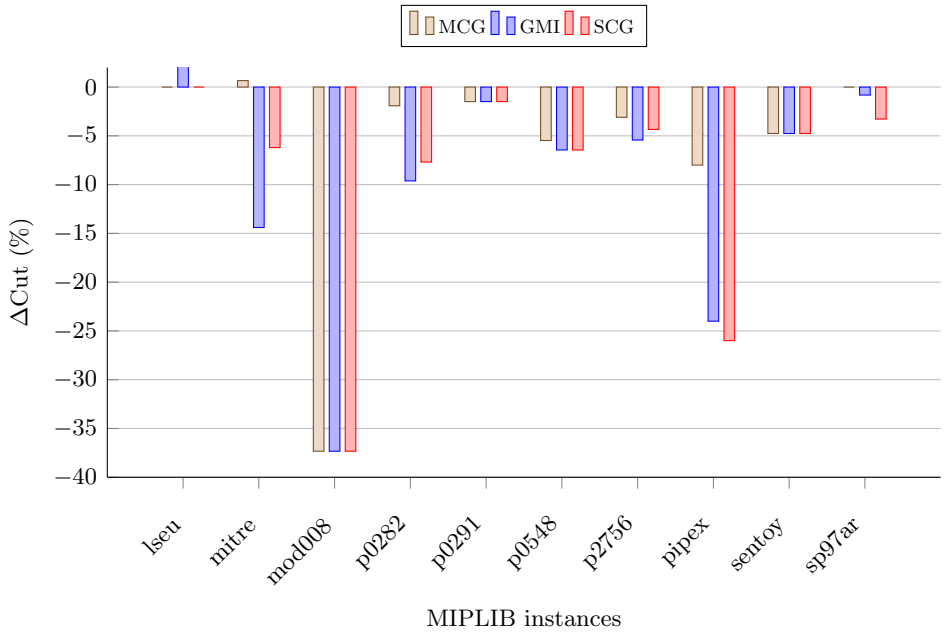


Figure 3.10: ΔCut (%) by strengthening methods for MIPLIB instances

The size of the enhanced formulations was still significantly reduced when using the strengthened cuts, as shown in Figure 3.10. The reduction effects of the Gomory mixed-integer cuts and the SCG cuts were comparable in most instances.

Figure 3.11 shows that the cutting plane time using the strengthened cuts also decreased in most instances. In particular, in most instances, the cutting plane algorithm using the Gomory mixed-integer cuts terminated earlier than the other strengthened cuts.

In conclusion, our strengthening method for maximal CG cuts for GKP polytopes could provide slightly more enhanced formulations defined with fewer cuts compared to results by the CG cuts. Furthermore, the cutting plane time could be saved due to the decreased number of iterations and generated cuts. However, the performance was similar to using the Gomory mixed-integer cuts.

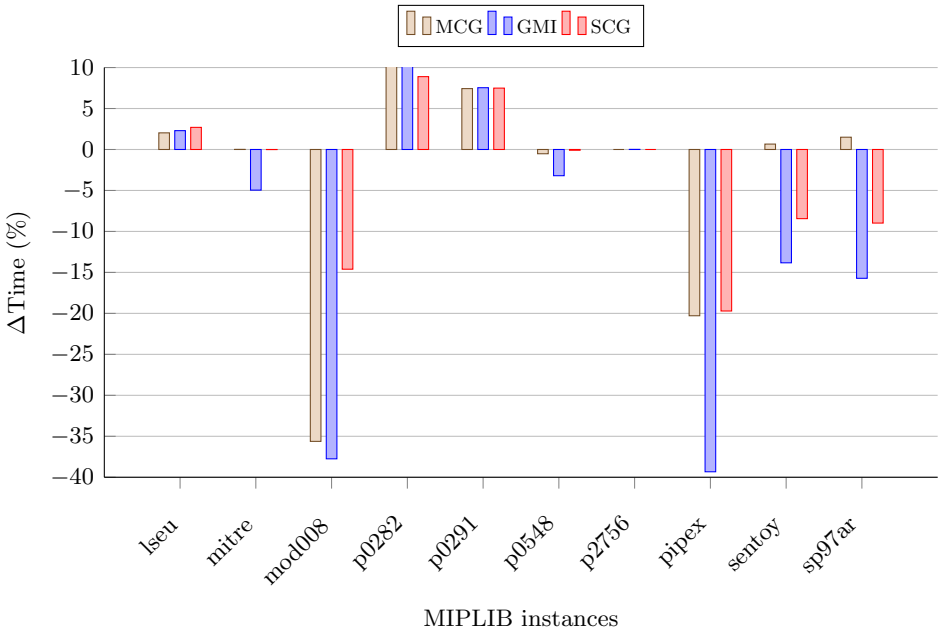


Figure 3.11: ΔTime (%) by strengthening methods for MIPLIB instances

3.7 Conclusion

In this study, we presented a novel method to strengthen CG cuts for binary integer linear programs. We first defined maximal CG cuts for binary knapsack polytopes and proposed a method to derive the maximal CG cuts from the given CG cuts for binary integer linear programs. We then introduced extended knapsack polytopes where the maximal CG cuts can be interpreted as their lifted cover inequalities. Our CG cut strengthening method utilized the lifting function that yields stronger lifted cover inequalities for the extended knapsack polytopes. Through theoretical analysis, we showed that obtained cuts are at least as strong as the Gomory mixed-integer cuts derived from the maximal CG cuts. In addition, we extended the method to CG cuts for binary integer linear programs with generalized upper bounds.

The computational test results demonstrated that the proposed strengthening methods yield cuts that can effectively enhance the formulations of binary integer linear programs. However, the performance of the cuts strengthened from maximal CG cuts for GKP polytopes was indifferent to the Gomory mixed-integer cuts.

Our extension to maximal CG cuts for GKP polytopes is based on the observation that maximal CG cuts for GKP polytopes can be interpreted as CG cuts for another binary knapsack polytope. Therefore, our strengthening method does not explicitly incorporate the generalized upper bounds. Another strengthening method can be devised by explicitly incorporating the generalized upper bounds. These methods could lead to even more enhanced formulations for binary integer linear programs with generalized upper bounds and potentially outperform the Gomory mixed-integer cuts.

Additionally, we evaluated the performance of the proposed strengthening methods for CG cuts derived from each constraint of the test instances. However, it is worth noting that our strengthening methods can be applied to CG cuts derived from any single-constraint relaxations, such as each row of the optimal simplex tableau. Therefore, evaluating the performance using such single-constraint relaxations is also necessary.

Chapter 4

Lifting heuristic of probabilistic cover inequalities for the chance-constrained binary knapsack problem

This chapter proposes an efficient lifting heuristic for probabilistic cover inequalities for the chance-constrained binary knapsack problem (CKP). We first introduce a non-convex relaxation for the CKP, formulated as a non-convex optimization problem, and show that the relaxation provides a tighter upper bound for the CKP compared to other continuous relaxations in the literature. Non-convex optimization problems are difficult to solve in general. However, we show that the non-convex relaxation for the CKP can be solved in polynomial time. We then propose a lifting heuristic for probabilistic cover inequalities using the polynomial-time algorithm for the non-convex relaxation. Computational test results demonstrate that the proposed lifting heuristic outperforms existing methods in terms of computational efficiency, while the effectiveness of the resulting lifted probabilistic cover inequalities remains competitive.

4.1 Introduction

We consider the chance-constrained binary knapsack problem (CKP) described as follows.

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \mathbb{P}\left\{ \sum_{j \in N} \tilde{a}_j x_j \leq b \right\} \geq \epsilon \\ & x \in \{0,1\}^n \end{aligned} \tag{4.1}$$

Here, $\tilde{a}_j$'s are random variables representing the weights of the items in N. Due to the chance constraint (4.1), the feasible solution set of the CKP depends on the distribution of $\tilde{a}_j$'s. In this study, we assume that the item weights are mutually independent and normally distributed, i.e., $\tilde{a}_j \sim N(a_j, \sigma_j^2)$ where $a_j \geq 0$ and $\sigma_j \geq 0$ for each $j \in N$. Then, the CKP can be formulated as a deterministic integer convex program (ICP) as follows.

$$\begin{aligned} \text{ICP:} \quad \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} a_j x_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 x_j^2} \leq b \\ & x \in \{0,1\}^n \end{aligned}$$

$\Phi^{-1}(\cdot)$ denotes the inverse of the cumulative density function for the standard normal distribution. Throughout this study, we refer to the CKP as the above problem. Without loss of generality, we assume that $a_j + \sigma_j > 0$, $a_j + \Phi^{-1}(\epsilon)\sigma \leq b$, and

$\sum_{j \in N} a_j + \Phi^{-1}(\epsilon)\sqrt{\sum_{j \in N} \sigma_j^2} > b.$

The CKP is a generalization of the binary knapsack problem, where the latter is a special case with $\sigma_j = 0$ for each $j \in N$. Therefore, the CKP is also NP-hard. The optimal solution for a CKP with a few items can be obtained using the branch-and-bound algorithm provided by commercial optimization software because the problem is an integer convex program. However, solving large-sized CKPs remains difficult. Scalable exact solution approaches, such as the dynamic programming algorithm for the binary knapsack problem, have not been studied yet. Instead, several studies have proposed scalable approximation algorithms (Goyal & Ravi, 2010; Han et al., 2016) and heuristics (Joung & Lee, 2020) for the CKP.

The CKP may seem artificial and disconnected from practical applications due to the assumption of the probability distribution of random variables. However, normal distributions have been considered in some applications of chance-constrained programs, for example, multi-robot teaming with uncertain reachable distances of robots (Yang & Chakraborty, 2018), vaccine allocation with uncertain vaccine efficiency (Tanner & Ntaimo, 2010), and network design with uncertain capacities (Atamtürk & Bhardwaj, 2018). For those problems, the CKP can represent single-constraint relaxations for the approximations that introduce the constraint-wise independence assumption between random input data, as discussed in Section 1.2.5. Additionally, the CKP is related to the robust knapsack problem under the ellipsoidal uncertainty set. Such an uncertainty set can be generally expressed as

$$\Xi = \left\{ \xi \in \mathbb{R}^n_+ : \sum_{j \in N} \left(\frac{\xi_j - a_j}{\sigma_j} \right)^2 \leq (\Phi^{-1}(\epsilon))^2 \right\},$$

for some $\sigma_j \in \mathbb{R}_+$ and $a_j \in \mathbb{R}_+$ for each $j \in N$ with some $\Phi^{-1}(\epsilon) > 0$. Then, the robust knapsack problem can be formulated as

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} \xi_j x_j \leq b, \ \forall \xi \in \Xi \\ & x \in \{0,1\}^n \end{aligned}$$

where the robust counterpart is

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} a_j x_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 x_j^2} \leq b \\ & x \in \{0,1\}^n. \end{aligned}$$

Therefore, the CKP is equivalent to the robust knapsack problem under the ellipsoidal uncertainty set.

Valid inequalities for the CKP can be used to solve these problems. Generalizing the concept of the cover (Balas, 1975; Wolsey, 1975; Hammer et al., 1975) for the binary knapsack problem, Atamtürk & Narayanan (2009) introduced probabilistic cover inequalities for the CKP described as

$$\sum_{j \in C_P} x_j \leq |C_P| - 1,$$

where $C_P \subseteq N$ is a probabilistic cover such that $\sum_{j \in C_P} a_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in C_P} \sigma_j^2} > b$.

The probabilistic cover inequalities can be strengthened using the sequential lifting technique proposed by Padberg (1975). The resulting lifted probabilistic cover inequalities can be written as follows.

$$\sum_{j \in C_P} x_j + \sum_{j \in N \setminus C_P} \gamma_j x_j \leq |C_P| - 1$$

In the sequential lifting technique, each of the lifting coefficients γ_j's is determined by solving the lifting problem, which aims to identify the maximum value of γ_j for each $j \in N$ while ensuring the validity of the resulting lifted probabilistic cover inequality for the CKP. The lifting problems can be formulated as another CKP where c_j's are bounded by n (Joung & Park, 2017).

In the cutting plane algorithm using lifted probabilistic cover inequalities, a number of the lifting problems should be solved repeatedly. However, to the best of our knowledge, efficient exact solution approaches for the lifting problem have not been studied yet. Although some studies have proposed lifting heuristics that utilize the upper bounds of the lifting problems (Atamtürk & Narayanan, 2009; Joung & Park, 2017), these methods may be time-consuming for the large-sized CKPs due to their high computational complexity.

In this study, we propose a more efficient lifting heuristic for probabilistic cover inequalities by utilizing a continuous relaxation for the CKP. We first introduce a non-convex continuous relaxation of the CKP and show that the relaxation provides tighter upper bounds compared to the other continuous relaxations found in the literature. The relaxation is formulated as a non-convex optimization problem for which efficient solution approaches do not exist. However, we show that the

non-convex continuous relaxation for the CKP can be solved in polynomial time. Subsequently, we devise a heuristic procedure to lift probabilistic cover inequalities using the exact polynomial-time algorithm for the non-convex continuous relaxation. Finally, we present computational test results to evaluate the performance of our lifting heuristic.

The remainder of this chapter is organized as follows. We review the relevant literature in Section 4.2. In Section 4.3, we introduce continuous relaxations for the CKP and compare their strength. In Section 4.4, we propose an exact polynomial time algorithm for a non-convex continuous relaxation of the CKP. The lifting heuristic using the algorithm is devised in Section 4.5. The computational efficiency of the proposed lifting heuristic, as well as the effectiveness of the resulting lifted probabilistic cover inequalities, is demonstrated in Section 4.6. Finally, the concluding remarks are given in Section 4.7.

4.2 Literature reviews

Continuous relaxations for the CKP can be obtained from various formulations for the CKP proposed in the literature. Firstly, a continuous relaxation can be defined from ICP by relaxing the integrality restriction. Goyal & Ravi (2010) showed that this relaxation can have a large integrality gap, particularly for large n.

Atamtürk & Narayanan (2008) introduced an integer linear program for the CKP using a submodular polytope (Conforti, Gérard, et al., 2014). The LP relaxation of this formulation leads to another continuous relaxation for the CKP, described with exponentially many constraints. Nevertheless, Atamtürk & Narayanan (2008) showed that the relaxation can be solved efficiently using a cutting plane algorithm.

Another possible formulation for the CKP can be derived by exploiting the binarity of the decision variables. The ICP can be reformulated by replacing x_j^2 with x_j for each $j \in N$ because $x_j^2 = x_j$ when x_j is binary. Goyal & Ravi (2010) used this formulation to develop a polynomial time approximation scheme for the CKP. This formulation can provide a continuous relaxation by relaxing the integrality restriction. Because the feasible solution set of the relaxation is non-convex, we call the relaxation the *non-convex relaxation* for the CKP.

These continuous relaxations for the CKP can be utilized to lift probabilistic cover inequalities. As mentioned in Section 4.1, the lifting problem for probabilistic cover inequalities can be formulated as another CKP. Instead of solving the lifting problem exactly, one can determine each lifting coefficient using the upper bound of the lifting problem, which can be obtained from one of the three continuous relaxations of the lifting problem. The resulting lifted probabilistic cover inequalities vary depending on which continuous relaxation is utilized because the relaxations may provide different upper bounds. However, to the best of our knowledge, comparative studies between the continuous relaxations for the CKP have not yet been conducted. Additionally, the relaxations have not been employed for the lifting heuristic, except for Atamtürk & Narayanan (2009).

Atamtürk & Narayanan (2009) proposed a lifting heuristic that utilizes the non-convex relaxation for the lifting problem. By exchanging the roles of the objective and constraint, the authors introduced an algorithm with a time complexity of $O(|C_P|n^3 \log n)$ to solve the relaxation for the lifting problem using the parametric optimization approach. While their lifting heuristic can be executed in polynomial time, obtaining lifted probabilistic cover inequalities for large-sized CKPs may re-

quire considerable computation time due to the high computational complexity involved. In this study, we propose a more efficient algorithm to solve the non-convex relaxation for the lifting problem. Our algorithm can also efficiently solve the non-convex relaxation for the CKP, not just for the lifting problem.

Another approach for determining lifting coefficients is based on the equivalence between the CKP and the robust knapsack problem under an ellipsoidal uncertainty set. Han et al. (2016) proposed a pseudo-polynomial time approximation algorithm for the robust knapsack problem by introducing the inner approximation of the ellipsoidal uncertainty set, represented as a polyhedron. Based on the observation that the robust knapsack problem under the approximated uncertainty set can provide the upper bound of the CKP, Joung & Park (2017) proposed a lifting heuristic for probabilistic cover inequalities, which utilizes the algorithm proposed by Han et al. (2016). Through extensive computational tests, the authors demonstrated that the proposed lifting heuristic could generate competitive lifted probabilistic cover inequalities in less computation time compared to Atamtürk & Narayanan (2009). However, their lifting heuristic has a pseudo-polynomial time complexity. In contrast, we propose a polynomial-time lifting heuristic.

4.3 Comparison of continuous relaxations for the chance-constrained binary knapsack problem

4.3.1 Continuous relaxations for the chanced-constrained binary knapsack problem

This section introduces continuous relaxations for the CKP, which are briefly reviewed in Section 4.2. We then compare the upper bounds provided by the relaxations.

The first relaxation can be derived from the ICP defined in Section 4.1 by relaxing the integrality restriction, which is described as $\max\{\sum_{j\in N} c_j x_j : x \in \mathcal{P}_C\}$, where

$$\mathcal{P}_C = \left\{ x \in [0,1]^n : \sum_{j\in N} a_j x_j + \Phi^{-1}(\epsilon)\sqrt{\sum_{j\in N} \sigma_j^2 x_j^2} \leq b \right\}.$$

We denote the above relaxation as the *convex relaxation* for the CKP because the feasible solution set $\mathcal{P}_C$ is a convex set. The upper bound provided from the convex relaxation, $z_C = \max\{\sum_{j\in N} c_j x_j : x \in \mathcal{P}_C\}$ can be obtained efficiently because the relaxation is a convex optimization problem.

The CKP can also be formulated as a problem that finds $S \subseteq N$ such that

$$\begin{aligned} \max \quad & c(S) \\ \text{s.t} \quad & F(S) \leq b \\ & S \subseteq N, \end{aligned}$$

where $c(s) = \sum_{j\in S} c_j$ and $F(S) = \sum_{j\in S} a_j + \Phi^{-1}(\epsilon)\sqrt{\sum_{j\in S} \sigma_j^2}$. $F(S)$ is a submodular set function, and the above problem can be reformulated as an integer linear

program as follows (Atamtürk & Narayanan, 2008).

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} \hat{\omega}_j x_j \leq b, \ \forall \hat{\omega} \in \text{ext}(\Omega) \\ & x \in \{0,1\}^n \end{aligned}$$

where $\text{ext}(\Omega)$ is the set of extreme points of Ω described as

$$\Omega = \left\{ \omega \in \mathbb{R}^n : \sum_{j \in S} \omega_j \leq F(S), \ \forall S \subseteq N \right\}.$$

Another continuous relaxation for the CKP can be obtained from the above formulation by relaxing the integrality restriction, which is described as $\max\{\sum_{j \in N} c_j x_j : x \in \mathcal{P}_P\}$, where

$$\mathcal{P}_P = \{x \in [0,1]^n : \hat{\omega}_j x_j \leq b, \ \forall \hat{\omega} \in \text{ext}(\Omega)\}$$

Because $|\text{ext}(\Omega)|$ is finite, $\mathcal{P}_P$ is a polyhedron. Hence, we call this relaxation the *polyhedral relaxation* for the CKP. Even though the polyhedral relaxation has exponentially many constraints, it can be solved in polynomial time because the separation problem for a given $\hat{x} \in [0,1]^n$, $\max\{\omega^T \hat{x} : \omega \in \Omega\}$, can be solved in $O(n \log n)$ (Edmond, 1970). Hence, the upper bound provided from the polyhedral relaxation, $z_P = \max\{\sum_{j \in N} c_j x_j : x \in \mathcal{P}_P\}$, can be obtained efficiently using the cutting plane algorithm.

On the other hand, the ICP can be rewritten by replacing x_j^2 with x_j for each

$j \in N$ as follows.

$$\begin{aligned} \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} a_j x_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 x_j} \leq b \\ & x \in \{0,1\}^n \end{aligned}$$

The binarity of the decision variables ensures the validity of the above formulation. This formulation yields a continuous relaxation described as $\max\{\sum_{j \in N} c_j x_j : x \in \mathcal{P}_{NC}\}$ where

$$\mathcal{P}_{NC} = \left\{ x \in [0,1] : \sum_{j \in N} a_j x_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 x_j} \leq b \right\}.$$

For the sake of brevity, let $g(x) = \sum_{j \in N} a_j x_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 x_j}$.

Proposition 4.1. *$g(x)$ is concave on $\mathbb{R}^n_+$.*

Proof. The function $\sqrt{z}$ is concave where $z \in \mathbb{R}_+$. Hence, for all $\theta \in [0,1]$ and $z_1, z_2 \in \mathbb{R}_+$, the following inequality holds.

$$\theta \sqrt{z_1} + (1-\theta)\sqrt{z_2} \leq \sqrt{\theta z_1 + (1-\theta) z_2}$$

From this result, for all $u, v \in \mathbb{R}^n_+$ and $\theta \in [0,1]$, it can be shown that

$$\theta g(u) + (1-\theta) g(v) = \theta \sum_{j \in N} a_j u_j + (1-\theta)$$

$$
\begin{aligned}
&= \sum_{j \in N} a_j v_j + \Phi^{-1}(\epsilon) \left(\theta \sqrt{\sum_{j \in N} \sigma_j^2 u_j} + (1-\theta) \sqrt{\sum_{j \in N} \sigma_j^2 v_j} \right) \\
&\leq \sum_{j \in N} a_j (\theta u_j + (1-\theta) v_j) + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 (\theta u_j + (1-\theta) v_j)} \\
&= g(\theta u + (1-\theta) v).
\end{aligned}
$$

Therefore, $g(x)$ is a concave function. □

By Proposition 4.1, $\mathcal{P}_{NC}$ is a non-convex set. Hence, we refer to the relaxation as the *non-convex relaxation* while z_{NC} denotes the upper bound provided by the relaxation. We first characterize the form of the optimal solution of the non-convex relaxation.

Proposition 4.2. *There exists an optimal solution for the non-convex relaxation in which, at most, one variable has a fractional value.*

Proof. Let $\hat{x} \in \mathcal{P}_{NC}$ be the optimal solution of the non-convex relaxation, and suppose that $\hat{x}$ has more than two fractional variables. Then, we show that another optimal solution can be constructed, where one of the fractional variables becomes 0 or 1.

Let k and l be the indices of the variables with fractional values in $\hat{x}$. Since $\hat{x}$ is optimal, $g(\hat{x}) = b$. If $g(\hat{x}) < b$, $\hat{x}_k$ or $\hat{x}_l$ can be increased, which implies that $\hat{x}$ is not an optimal solution. Let us consider the following optimization problem.

$$
\max \quad \sum_{j \in N \setminus \{k,l\}} c_j \hat{x}_j + c_k x_k + c_l x_l
$$

$$\text{s.t} \quad \sum_{j \in N \setminus \{k,l\}} a_j \hat{x}_j + a_k x_k + a_l x_l + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N \setminus \{k,l\}} \sigma_j^2 \hat{x}_j + \sigma_k^2 x_k + \sigma_l^2 x_l} = b \tag{4.2}$$

$$0 \leq x_k, x_l \leq 1$$

Let $(x_k^\star, x_l^\star)$ be the optimal solution to the above problem. Using $(x_k^\star, x_l^\star)$, we can construct another solution $x^\star$ for the non-convex relaxation, where $x_j^\star = \hat{x}_j$ for $j \in N \setminus \{k, l\}$. Then, $x^\star$ is also an optimal solution for the relaxation, that is, $\sum_{j \in N} c_j \hat{x}_j = \sum_{j \in N} c_j x_j^\star$.

On the other hand, the above problem can be reformulated as an optimization problem with a single variable. Due to the constraint (4.2), x_l can be expressed as a function for x_k, $l(x_k)$ such that

$$\sum_{j \in N \setminus \{k,l\}} a_j \hat{x}_j + a_k x_k + a_l l(x_k) + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N \setminus \{k,l\}} \sigma_j^2 \hat{x}_j + \sigma_k^2 x_k + \sigma_l^2 l(x_k)} = b. \tag{4.3}$$

Then, $(x_k^\star, x_l^\star)$ can be obtained by solving

$$\max_{0 \leq x_k \leq 1} \{c_k x_k + c_l l(x_k) : 0 \leq l(x_k) \leq 1\}. \tag{4.4}$$

We show that the objective function of the problem (4.4) is convex. For simplicity, let

$$s(x_k) = \sqrt{\sum_{j \in N \setminus \{k,l\}} \sigma_j^2 \hat{x}_j + \sigma_k^2 x_k + \sigma_l^2 l(x_k)}$$

where the first and second derivatives are

$$s'(x_k) = \frac{\sigma_k^2 + \sigma_l^2 l'(x_k)}{2s(x_k)},$$

and

$$s''(x_k) = \frac{\sigma_l^2 l''(x_k)s(x_k) - (\sigma_k^2 + \sigma_l^2 l'(x_k))s'(x_k)}{2s(x_k)^2},$$

respectively. From the second derivative of the equality (4.3), we can see that

$$a_l l''(x_k) + \Phi^{-1}(\epsilon)s''(x_k) = 0,$$

and the following equality holds.

$$\left(a_l + \Phi^{-1}(\epsilon)\frac{\sigma_l^2}{2s(x_k)}\right) l''(x_k) = \Phi^{-1}(\epsilon)\frac{(\sigma_k^2 + \sigma_l^2 l'(x_k))^2}{2s(x_k)^3}$$

which implies that $l''(x_k) \geq 0$, that is, $l(x_k)$ is convex.

This result implies that there exists an optimal solution for the problem (4.4), lying on the boundary of x_k, that is, $x_k^\star \in \{0, 1\}$ or $l(x_k^\star) \in \{0, 1\}$. In other words, $x_k^\star \in \{0, 1\}$ or $x_l^\star \in \{0, 1\}$. Then, the corresponding $x^\star$ is another optimal solution for the relaxation, which has fewer fractional variables. By applying this construction repeatedly until, at most one variable has a fractional value, we can obtain an optimal solution for the relaxation, which satisfies the condition in the statement. □

The following section shows that the non-convex relaxation provides a tighter bound for the CKP, compared to the other continuous relaxations, based on Proposition 4.2.

4.3.2 Bound comparison for continuous relaxations

Proposition 4.3. $z_{NC} \leq z_C$

Proof. We show that $\mathcal{P}_{NC} \subseteq \mathcal{P}_C$. For given $\hat{x} \in \mathcal{P}_{NC}$, it is clear that

$$\sum_{j \in N} a_j \hat{x}_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 \hat{x}_j^2} \leq \sum_{j \in N} a_j \hat{x}_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_j^2 \hat{x}_j} \leq b,$$

because $\hat{x} \in [0,1]^n$. Therefore, the result follows. □

Proposition 4.4. $z_{NC} \leq z_P$

Proof. We show that the optimal solution of the non-convex relaxation is feasible for $\mathcal{P}_P$. Let $x^\star$ be the optimal solution of the non-convex relaxation. By Proposition 4.2, $x^\star$ can be described as, for some $Q \subseteq N$ and $f \in N \setminus Q$, $x_j^\star = 1$ for $j \in Q$, $0 \leq x_f < 1$, and $x_j = 0$, $j \in N \setminus Q \cup \{f\}$. For brevity, let $Q = \{1, ..., f-1\}$.

Let us consider the separation problem associated with $\mathcal{P}_P$ described as

$$\eta^\star = \max \left\{ \sum_{j \in N} \omega_j x_j^\star : \sum_{j \in S} \omega_j \leq F(S),\ S \subseteq N \right\}.$$

If $x^\star \notin \mathcal{P}_P$, then $\eta^\star > b$ while $\eta^\star \leq b$ if $x^\star \in \mathcal{P}_P$. The optimal solution of the separation problem is $\omega_j^\star = F(S_j) - F(S_{j-1})$ for each $j \in N$ where $S_j = \{1, ..., j\}$ and $S_0 = \emptyset$. Then,

$$\eta^\star = \sum_{j \in N} \omega_j^\star x_j^\star = F(S_{f-1}) + (F(S_f) - F(S_{f-1})) x_f^\star.$$

Let $x(S)$ be the vector that represents a set $S \subseteq N$ such that $x_j(S) = 1$ if $j \in S$, otherwise, 0. Then, it is clear that $F(S) = g(x(S))$ for all $S \subseteq N$, and $\eta^\star$ can be

rewritten as follows.

$$\eta^\star = (1 - x_f^\star)g(x(S_{f-1})) + x_f^\star g(x(S_f)).$$

We note that $x^\star = (1 - x_f^\star)x(S_{f-1}) + x_f^\star x(S_f)$. Therefore,

$$b \geq g(x^\star) = g((1 - x_f^\star)x(S_{f-1}) + x_f^\star x(S_f)) \geq \eta^\star$$

where the last inequality holds due to the concavity of g. This result implies that $x^\star \in \mathcal{P}_P$. Therefore, the result follows. □

Proposition 4.3 and 4.4 imply that the non-convex relaxation provides the tighter upper bound for the CKP compared to other continuous relaxations. Additionally, we show that the integrality gap of the non-convex relaxation, $(z_{NC} - z_{OPT})/z_{OPT}$, is bounded where z_{OPT} is the optimal objective value of the CKP.

Proposition 4.5. $\frac{z_{NC} - z_{OPT}}{z_{OPT}} \leq 1$

Proof. Let $x^\star$ be the optimal solution for the CKP's non-convex relaxation, which satisfies Proposition 4.2. Let $f \in N$ be the fractional variable in $x^\star$ such that $0 \leq x_f^\star < 1$.

From $x^\star$, we can define x^L and x^U as $x_j^L = \lfloor x_j^\star \rfloor$ for each $j \in N$ and $x_f^U = 1$ while $x_j^U = 0$ for each $j \in N \setminus \{f\}$. Then, it is clear that x^L and x^U are feasible solutions for the CKP. Then, the following inequalities hold.

$$z_{NC} = \sum_{j \in N \setminus \{f\}} c_j x_j^\star + c_f x_f^\star \leq \sum_{j \in N \setminus \{f\}} c_j x_j^L + c_f x_f^U \leq 2 z_{OPT}$$

The above inequalities imply that

$$\frac{z_{NC} - z_{OPT}}{z_{OPT}} \leq \frac{z_{OPT}}{z_{OPT}} = 1.$$

Therefore, the result follows. □

Proposition 4.5 implies that the quality of the upper bound provided by the non-convex relaxation is guaranteed, whereas the convex relaxation for the CKP can have a large integrality gap which increases as n increases (Goyal & Ravi, 2010). The following example shows the difference definitely.

Example 4.1 (Goyal & Ravi, 2010)**.** *Let us consider the CKP where* $c_j = \sigma_j = 1$*, and* $a_j = 1/\sqrt{n}$ *for each* $j \in N$*. Additionally, let* $b = 3$ *and* $\Phi^{-1}(\epsilon) = 1.6$*. Then, the optimal solution should select three items for a sufficiently large* n*. If four items are selected, then it is not feasible for the CKP because*

$$\frac{4}{\sqrt{n}} + \Phi^{-1}(\epsilon)\sqrt{4} > 3.$$

Therefore, $z_{OPT} = 3$*. On the other hand, let* $\hat{x}_j = 1/\sqrt{n}$ *for each* $j \in N$*. Then,* $\hat{x}$ *is feasible for the convex relaxation because*

$$\sum_{j \in N} \frac{1}{\sqrt{n}} \cdot \frac{1}{\sqrt{n}} + \Phi^{-1}(\epsilon)\sqrt{\sum_{j \in N} \left(\frac{1}{\sqrt{n}}\right)^2} = 1 + \Phi^{-1}(\epsilon) \leq 3.$$

The corresponding objective value for the convex relaxation is $\sqrt{n}$*. Because* $\hat{x}$ *may not be an optimal solution,* $\sqrt{n} \leq z_C$*. Therefore, the integrality gap of the convex relaxation is greater than* $(\sqrt{n} - 3)/3$*, which increases as* n *increases. On the other*

hand, let us define $x^\star$ *as* $x_1^\star = x_2^\star = x_3^\star = 1$ *and*

$$x_4^\star = 3(\sqrt{n}-1) + \frac{(\Phi^{-1}(\epsilon))^2 n}{2} - \sqrt{\frac{(\Phi^{-1}(\epsilon))^4 n^2}{4} + 3(\Phi^{-1}(\epsilon))^2 n\sqrt{n}},$$

while $x_j = 0$*, otherwise. By Proposition 4.2, it can be easily shown that* $x^\star$ *is the optimal solution for the non-convex relaxation. Then,*

$$\begin{aligned} z_{NC} &= 3\sqrt{n} + \frac{(\Phi^{-1}(\epsilon))^2 n}{2} - \sqrt{\frac{(\Phi^{-1}(\epsilon))^4 n^2}{4} + 3(\Phi^{-1}(\epsilon))^2 n\sqrt{n}} \\ &= \frac{9n}{\sqrt{n} + \frac{(\Phi^{-1}(\epsilon))^2 n}{2} + \sqrt{\frac{(\Phi^{-1}(\epsilon))^4 n^2}{4} + 3(\Phi^{-1}(\epsilon))^2 n\sqrt{n}}} \end{aligned}$$

Because

$$\lim_{n\to\infty} z_{NC} = \frac{9}{(\Phi^{-1}(\epsilon))^2},$$

the integrality gap of the non-convex relaxation converges if $n \to \infty$ *whereas the integrality gap of the convex relaxation diverges because* $\lim_{n\to\infty} z_C = \infty$*. The integrality gap of the non-convex relaxation and the lower bound of the integrality gap of the convex relaxation depending on* n *are illustrated in Figure 4.1.*

Although the non-convex relaxation can provide a tight upper bound for the CKP, it is not straightforward to obtain the bound because the non-convex relaxation is a non-convex optimization problem. However, in the following section, we propose an efficient algorithm to solve the non-convex relaxation using the property of the optimal solution.

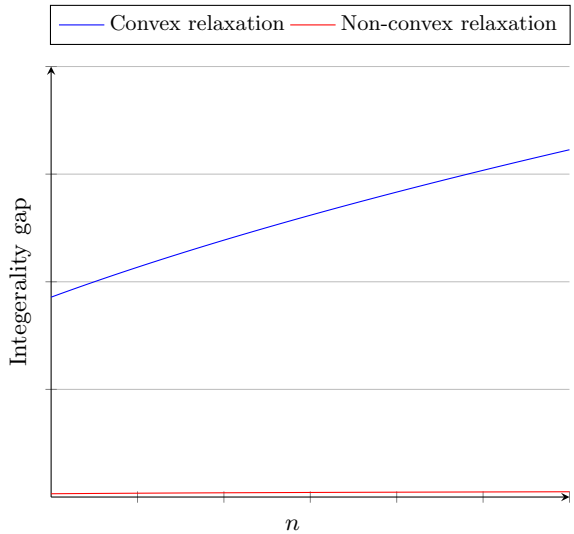


Figure 4.1: Integrality gaps of convex and non-convex relaxations

4.4 Polynomial-time algorithm for the non-convex relaxation

4.4.1 Reformulation of the non-convex relaxation

Let $x^\star$ be an optimal solution to the non-convex relaxation and $\delta^* = \sum_{j \in N} \sigma_j^2 x_j^\star$. Let us consider the following problem, which is the LP relaxation of a binary knapsack problem, when $\delta^\star > 0$,

$$\begin{aligned} \text{SK:} \quad \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} \bar{a}_j x_j \leq \bar{b} \\ & x \in [0,1]^n, \end{aligned}$$

where $\bar{a}_j = a_j + \frac{\Phi^{-1}(\epsilon)}{2\sqrt{\delta^\star}}\sigma_j^2$ for each $j \in N$ and $\bar{b} = b - \frac{\Phi^{-1}(\epsilon)}{2}\sqrt{\delta^\star}$. When $\delta^\star = 0$, the SK is defined as follows.

$$\begin{aligned} \text{SK:} \quad \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} a_j x_j \leq b \\ & x_j = 0, \ j \in N \setminus N' \\ & x \in [0,1]^n, \end{aligned}$$

where $N' = \{j \in N : \sigma_j = 0\}$ and $n' = |N'|$.

Proposition 4.6. *An optimal solution for the SK is optimal for the non-convex relaxation.*

Proof. It is clear that $x^\star$ is a feasible solution for the SK. Hence, $z_{SK} \geq z_{NC}$ where z_{SK} is the optimal objective value of the SK. Let $\mathcal{P}_{SK}$ denote the feasible solution set of the SK. We show that $\mathcal{P}_{SK} \subseteq \mathcal{P}_{NC}$, which implies that $z_{SK} = z_{NC}$.

Let $\hat{x} \in \mathcal{P}_{SK}$ and $\hat{\delta} = \sum_{j \in N} \sigma_j^2 \hat{x}_j$. If $\delta^\star = 0$, then $\sum_{j \in N} a_j \hat{x}_j \leq b$ by the definition of the SK. Because $g(\hat{x}) = \sum_{j \in N} a_j \hat{x}_j$, $g(\hat{x}) \leq b$, which implies that $\hat{x} \in \mathcal{P}_{NC}$. Now, suppose that $\delta^\star > 0$. By the definition of the SK, $\hat{x}$ satisfies the following inequality.

$$\sum_{j \in N} a_j \hat{x}_j + \frac{\Phi^{-1}(\epsilon)}{2} \left(\frac{\hat{\delta}}{\sqrt{\delta^\star}} + \sqrt{\delta^\star} \right) \leq b.$$

In addition, the following inequality holds by the arithmetic–geometric mean inequality.

$$2\sqrt{\hat{\delta}} \leq \frac{\hat{\delta}}{\sqrt{\delta^\star}} + \sqrt{\delta^\star}$$

These results imply that

$$g(\hat{x}) = \sum_{j \in N} a_j \hat{x}_j + \Phi^{-1}(\epsilon)\sqrt{\hat{\delta}} \leq \sum_{j \in N} a_j \hat{x}_j + \frac{\Phi^{-1}(\epsilon)}{2}\left(\frac{\hat{\delta}}{\sqrt{\delta^\star}} + \sqrt{\delta^\star}\right) \leq b.$$

Therefore, $\hat{x} \in \mathcal{P}_{NC}$ and $\mathcal{P}_{SK} \subseteq \mathcal{P}_{NC}$. Because $z_{SK} \geq z_{NC}$, these results imply that $z_{NC} = z_{SK}$. Hence, an optimal solution for the SK is not only feasible but also optimal for the non-convex relaxation. □

Recall that the SK is the LP relaxation of a binary knapsack problem. When $\delta^\star > 0$, an optimal solution for the SK can be obtained in a greedy manner with respect to the values of $p_j(\delta^\star)$'s where

$$p_j(\delta^\star) = \frac{c_j}{a_j + \frac{\Phi^{-1}(\epsilon)}{2\sqrt{\delta^\star}}\sigma_j^2},\ j \in N.$$

Let $\tau(\delta^*) = (\tau_1, ..., \tau_n)$ be the sequence of variables sorted in descending order of $p_j(\delta^\star)$'s. Then, the optimal solution, $x(\delta^\star)$, for the SK is defined as, for $t = \min\{k \in N : \sum_{j=1}^{k} \bar{a}_{\tau_j} > \bar{b}\}$,

$$x_{\tau_j}(\delta^\star) = \begin{cases} 1 & ,\ j < t \\ (\bar{b} - \sum_{k=1}^{t-1} \bar{a}_{\tau_k})/\bar{a}_{\tau_t} & ,\ j = t \\ 0 & ,\ j > t \end{cases}.$$

By Proposition 4.6, $x(\delta^\star)$ is an optimal solution for the non-convex relaxation when $\delta^\star > 0$. In addition, $g(x(\delta^\star)) = b$. Similarly, we can define an optimal solution for the SK when $\delta^\star = 0$, $x(0)$. Let $\tau(0) = (\tau_1, ..., \tau_{n'})$ be the sequence of variables in N' sorted in descending order of c_j/a_j. Then, $x(0)$ can be constructed in the same

manner with $x(\delta^\star)$ where $x_j(0) = 0$ for each $j \in N \setminus N'$. By Proposition 4.6, $x(0)$ is also an optimal solution for the non-convex relaxation when $\delta^\star = 0$.

We can generalize the sequence $\tau(\delta^\star)$ to some $\delta > 0$. Let $\tau(\delta) = (\tau_1, \ldots, \tau_n)$ be the sequence of variables sorted in descending order of $p_j(\delta)$ where

$$p_j(\delta) = \frac{c_j}{a_j + \frac{\Phi^{-1}(\epsilon)}{2\sqrt{\delta}}\sigma_j^2}.$$

Ties in the values of $p_j(\delta)$'s are broken in descending order of σ_j. In addition, let $x(\delta)$ be a feasible solution of the non-convex relaxation, which is defined as, for $t = \min\{k \in N : \sum_{j=1}^{k} a_j + \Phi^{-1}(\epsilon)\sqrt{\sum_{j=1}^{k} \sigma_j^2} > b\}$, $x_{\tau_j}(\delta) = 1$ for each $j < t$ and $x(\delta)_{\tau_j} = 0$ for each $j > t$ while $x_{\tau_t}(\delta)$ is the solution of the following quadratic equation.

$$\Phi^{-1}(\epsilon)^2 \left(\sum_{j=1}^{t} \sigma_{\tau_j}^2 + \sigma_{\tau_t}^2 x_{\tau_t}(\delta) \right) = \left(b - \sum_{j=1}^{t-1} a_{\tau_j} - a_{\tau_t} x_{\tau_t}(\delta) \right)^2$$

Then, by definition, $x(\delta)$ has at most one fractional variable and $g(x(\delta)) = b$.

The non-convex relaxation can be reformulated using the notion of $x(\delta)$ as an optimization problem with a single variable as follows.

$$\begin{aligned} \text{NCR:} \quad \max \quad & \sum_{j \in N} c_j x_j(\delta) \\ \text{s.t} \quad & \delta \in \mathbb{R}_+, \end{aligned}$$

where the optimal solution is $\delta^\star$. Now, we propose an efficient algorithm to solve the above optimization problem.

4.4.2 Algorithm to solve the reformulated non-convex relaxation

By definition, $\tau(\delta)$ changes as δ increases. However, it does not change continuously. For example, let $N = \{1, 2\}$, $\Phi^{-1}(\epsilon) = 2$, $(c_1, a_1, \sigma_1^2) = (1, 2, 3)$, and $(c_2, a_2, \sigma_2^2) = (2, 3, 1)$. Then, $\tau(\delta)$ is determined by

$$p_1(\delta) = \frac{1}{2 + \frac{3}{\sqrt{\delta}}} \text{ and } p_2(\delta) = \frac{2}{3 + \frac{1}{\sqrt{\delta}}}.$$

We note that $p_1(\delta) = p_2(\delta)$ when $\delta = 25$, and $p_1(\delta) < p_2(\delta)$ when $\delta < 25$ while $p_1(\delta) \leq p_2(\delta)$, otherwise. Hence, $\tau(\delta) = (2, 1)$ if $\delta < 25$, $\tau(\delta) = (1, 2)$ if $\delta \geq 25$.

Based on this observation, we first show that there exists a finite number of different $\tau(\delta)$'s for all $\delta \in \mathbb{R}_+$.

Proposition 4.7. *There exist at most* $\left(\frac{n(n-1)}{2} + 2\right)$ *different* $\tau(\delta)$*'s for all* $\delta \in \mathbb{R}_+$.

Proof. As shown in the above example, $\tau(\delta)$ changes only before and after δ such that $p_k(\delta) = p_l(\delta)$ for some $\{k, l\} \in N$. Hence, there exist at most $\frac{n(n-1)}{2}$ such δ's with their corresponding $\tau(\delta)$'s. Let $\delta_{\min}$ be the minimum value among such δ's. Then, we can obtain another variable sequence $\tau(\delta_{\min} - \varepsilon)$ where $0 < \varepsilon < \delta_{\min}$. We note that $\tau(\delta) = \tau(\delta_{\min} - \varepsilon)$ for all $\delta < \delta_{\min}$. In addition, we have $\tau(0)$. Therefore, $\left(\frac{n(n-1)}{2} + 2\right)$ different $\tau(\delta)$'s can exist. □

For some $k \in N$ and $l \in N$ such that $k \neq l$, let δ_{kl} be the solution of $p_k(\delta) = p_l(\delta)$ if such δ exists.From the proof of Proposition 4.7, different $\tau(\delta)$'s can be derived from $\delta \in \Delta$ where

$$\Delta = \{0, \delta_{\min} - \varepsilon\} \cup \{\delta_{kl} : \{k, l\} \subseteq N, k < l\},$$

and $\delta_{\min} = \min\{\delta_{kl} : \{k, l\} \subseteq N, k < l\}$. Because only different $\tau(\delta)$'s can yield

different $x(\delta)$'s by definition, it is sufficient to consider $x(\delta)$'s such that $\delta \in \Delta$ to solve the NCR. This observation leads to a polynomial-time algorithm to solve the NCR, described as follows.

Algorithm 10 Enumeration method for the NCR

$\Delta \leftarrow \{0\}$;
$\delta_{\min} \leftarrow \infty$ and $z^\star \leftarrow 0$;
for $k, l \in N$ **do**
    $\delta_{kl} \leftarrow$solve $p_k(\delta) = p_l(\delta)$;
    $\Delta \leftarrow \Delta \cup \{\delta_{kl}\}$;
    $\delta_{\min} \leftarrow \delta_{kl}$ if $\delta_{kl} < \delta_{\min}$;
end for
for $\delta \in \Delta$ **do**
    Construct $\tau(\delta)$ and $x(\delta)$;
    $z^\star \leftarrow \sum_{j \in N} c_j x_j(\delta)$ if $z^\star < \sum_{j \in N} c_j x_j(\delta)$;
end for
return $z^\star$;

Theorem 4.1. *The NCR can be solved in* $O(n^3 \log n)$.

Proof. In Algorithm 10, it requires $O(n^2)$ computations to construct Δ while $x(\delta)$ for each $\delta \in \Delta$ can be obtained in $O(n \log n)$. Therefore, the result follows. □

Although the NCR can be solved in polynomial time, constructing Δ requires $O(n^2)$ computation time, which can be time-consuming for large n. Therefore, we propose an adaptive method to solve the NCR without enumerating elements of Δ.

For a given $q \in \mathbb{R}^+$, let $\bar{\delta} = \min\{\delta \in \Delta : \delta > q,\ x(\delta) \neq x(q)\}$. We first characterize the candidates for $\bar{\delta}$. For the sake of brevity, let $\tau(q) = (1, \ldots, n)$. Then, for some $f \in N$, $x(q)$ can be described as $x_j(q) = 1$ for each $j \in Q$ where $Q = \{1, \ldots, f-1\}$, $0 \leq x_f(q) < 1$, and $x_j(q) = 0$ for each $j \in N \setminus Q$.

Proposition 4.8. $\bar{\delta} \in \{\delta_{fk} : k \in N \setminus \{f\}\}$.

Proof. Suppose that $\bar{\delta} = \delta_{ij}$ such that $\{i, j\} \in Q$. Then, $\tau(\bar{\delta})$ yields a sequence of variables where only the orders of some variables in Q change from $\tau(q)$. Therefore, $x(\bar{\delta}) = x(q)$, which conflicts the definition of $\bar{\delta}$.

Now, suppose that $\bar{\delta} = \delta_{ij}$ such that $\{i, j\} \in N \setminus (Q \cup \{f\})$. $\tau(\bar{\delta})$ yields a sequence of variables where only the orders of some variables in $N \setminus (Q \cup \{f\})$ changes from $\tau(q)$, and $x(\bar{\delta}) = x(q)$ which conflicts the definition of $\bar{\delta}$.

Finally, suppose that $\bar{\delta} = \delta_{ij}$ such that $i \in Q$ and $j \in N \setminus (Q \cup \{f\})$, i.e., $i < f < j$. By definition, $p_i(\bar{\delta}) = p_j(\bar{\delta})$. We show that

$$\min\{\delta_{fi}, \delta_{fj}\} \leq \bar{\delta}.$$

If $p_f(\bar{\delta}) = p_i(\bar{\delta})$, it is clear that $\delta_{fi} = \bar{\delta}$, hence, $\bar{\delta}$ can be obtained from δ_{fi}. On the other hand, if $p_f(\bar{\delta}) > p_i(\bar{\delta})$, then $q < \delta_{fi} \leq \bar{\delta}$, because $p_i(q) \geq p_f(q)$ and $p_i(\bar{\delta}) < p_f(\bar{\delta})$ while $p_i(\delta)$ and $p_f(\delta)$ are continuous non-decreasing functions. Finally, if $p_f(\bar{\delta}) < p_i(\bar{\delta})$, then $p_f(\bar{\delta}) < p_j(\bar{\delta})$ while $p_f(q) \geq p_j(q)$. Therefore, $q < \delta_{fj} \leq \bar{\delta}$ because $p_j(\delta)$ is also a continuous non-decreasing function. These results imply that $\bar{\delta}$ can be obtained from δ_{fk}'s where $k \in N \setminus \{f\}$. □

Proposition 4.8 provides a cornerstone for our adaptive method to solve the NCR. For any given $q \in \mathbb{R}+$, we can find $\delta > q$ in $O(n)$ such that $\delta \in \Delta$ and $x(\delta)$ may differ from $x(q)$ by searching for

$$\min_{k \in N \setminus f} \{\delta_{fk} : \delta_{fk} > q\}$$

where f is the fractional variable in $x(q)$. Similarly, for the obtained δ, we can

find another $\delta' \in \Delta$ that leads to a different feasible solution for the non-convex relaxation by applying the above procedure with δ replacing q. This way, we can identify the elements of Δ greater than q that produce different feasible solutions for the NCR without explicitly enumerating Δ.

Our adaptive method explores the solutions $x(\delta)$ obtained by repeatedly applying the above procedure, starting from a suitable initial value $q \in \mathbb{R}_+$. If $q = 0$, the adaptive method can yield the optimal solution for the NCR. However, the method may require many iterations until the optimal solution is found. In the subsequent discussion, we propose the upper and lower bounds for the optimal solution $\delta^\star$ for the NCR, which can be used to reduce the number of iterations of the adaptive method.

Recall that $\delta^\star = \sum_{j \in N} \sigma_j^2 x_j^\star$ where $x^\star$ is the optimal solution for the non-convex relaxation. Then, the upper bounds for $\delta^\star$ can be obtained by solving the following optimization problem.

$$\delta^\star_{\max} = \max_{x \in [0,1]^n} \left\{ \sum_{j \in N} \sigma_j^2 x_j : g(x) = b \right\}, \tag{4.5}$$

Similarly, the lower bounds for $\delta^\star$ can be obtained by solving

$$\delta^\star_{\min} = \min_{x \in [0,1]^n} \left\{ \sum_{j \in N} \sigma_j^2 x_j : g(x) = b \right\}. \tag{4.6}$$

For the sake of brevity, let $N = \{1, \ldots, n\}$ satisfy

$$\frac{\sigma_1^2}{a_1} \geq \frac{\sigma_2^2}{a_2} \geq \cdots \geq \frac{\sigma_n^2}{a_n}.$$

Additionally, we define $t_1 = \min\{k \in N : \sum_{j=1}^{k} a_j + \Phi^{-1}(\epsilon)\sqrt{\sum_{j=1}^{k} \sigma_j^2} > b\}$ and $t_2 = \max\{k \in N : \sum_{j=k}^{n} a_j + \Phi^{-1}(\epsilon)\sqrt{\sum_{j=k}^{n} \sigma_j^2} > b\}$. Subsequently, we define $x^{\max}$ as $x_j^{\max} = 1$ for each $j < t_1$ and $x_j^{\max} = 0$ for each $j > t_1$ while $x_{t_1}^{\max}$ makes $g(x^{\max}) = b$. Similarly, $x^{\min}$ is defined as $x_j^{\min} = 0$ for each $j < t_2$ and $x_j^{\max} = 1$ for each $j > t_2$ while $x_{t_2}^{\max}$ makes $g(x^{\min}) = b$. By definition, $x^{\max}$ and $x^{\min}$ are feasible for the problems (4.5) and (4.6), respectively.

Proposition 4.9. *$x^{\max}$ and $x^{\min}$ are optimal solutions for the problems* (4.5) *and* (4.6)*, respectively.*

Proof. We first show that $x^{\max}$ is the optimal solution for (4.5). Suppose that $x^{\max}$ is not the optimal solution, and there exists $x' \neq x^{\max}$ such that $g(x') = b$ and $\sigma(x') > \sigma(x^{\max})$ where

$$\sigma(x) = \sum_{j \in N} \sigma_j^2 x_j.$$

Let us consider the following problem.

$$\begin{aligned} \max \quad & \sum_{j \in N} \sigma_j^2 x_j \\ \text{s.t} \quad & \sum_{j \in N} a_j x_j \leq b - \Phi^{-1}(\epsilon)\sqrt{\sigma(x^{\max})} \\ & x \in [0,1]^n, \end{aligned}$$

which is the LP relaxation of a binary knapsack problem. x^{max} and x' are feasible for the above problem. Because $\sigma(x^{max}) < \sigma(x')$, $\sigma(x^{max})$ is not the optimal solution for the above problem. However, the optimal solution to the above problem should be $x^{\max}$ by its definition and the property of the LP relaxation of a binary knapsack

problem. Therefore, such x' cannot exist, and $x^{\max}$ is the optimal solution for the problem 4.5.

Now, we show that x^{min} is the optimal solution for (4.6). Suppose that there exists x' such that $g(x') = b$ and $\sigma(x') < \sigma(x^{\min})$. Then, $x_{\min}$ and x' are feasible for the following problem.

$$\begin{aligned} \min \quad & \sum_{j \in N} \sigma_j^2 x_j \\ \text{s.t} \quad & \sum_{j \in N} a_j x_j \geq b - \Phi^{-1}(\epsilon)\sqrt{\sigma(x^{\min})} \\ & x \in [0,1]^n, \end{aligned}$$

In addition, $\sigma(x^{\min}) > \sigma(x')$. However, for the same reason with $x^{\max}$, $x^{\min}$ is the optimal solution to the above problem. Therefore, the result follows. □

Proposition 4.9 implies that $\delta^\star_{\max}$ and $\delta^\star_{\min}$ can be obtained in $O(n \log n)$. On the other hand, $\delta^\star_{\min}$ can be used as an initial value q for the adaptive method while $\delta^\star_{\max}$ can be used for the termination condition. The overall adaptive method for the NCR using the bounds for $\delta^\star$ is described in Algorithm 11.

We note that each iteration in the loop of Algorithm 11 requires $O(n \log n)$ computation. Such iteration is repeated at most $O(n^2)$ times, hence, the algorithm also has $O(n^3 \log n)$ computational complexity.

While the upper bound of the CKP can be obtained through the non-convex relaxation using Algorithm 11, we can also obtain the lower bound for the CKP by constructing a feasible solution for the CKP from the optimal solution for the relaxation.

Algorithm 11 Adaptive method for the NCR

$z^\star \leftarrow \sum_{j \in N} c_j x_j(0)$;
$\delta^\star_{\max} \leftarrow \max_{x \in [0,1]^n} \{\sum_{j \in N} \sigma_j^2 x_j : g(x) = b\}$;
$\delta^\star_{\min} \leftarrow \min_{x \in [0,1]^n} \{\sum_{j \in N} \sigma_j^2 x_j : g(x) = b\}$;
$\delta \leftarrow \delta^\star_{\min}$;
while $\exists \delta$ and $\delta \leq \delta^\star_{\max}$ **do**
    Construct $\tau(\delta)$, $x(\delta)$, and $f \leftarrow$fractional variable index of $x(\delta)$;
    if $\sum_{j \in N} c_j x_j(\delta) > z^\star$ **then**
        $z^\star \leftarrow \sum_{j \in N} c_j x_j(\delta)$;
    end if
    $\delta \leftarrow \min\{\delta_{fk} > \delta : k \in N \setminus \{f\}\}$;
end while
return $z^\star$;

From the proof of Proposition 4.5, let us recall x^L and x^U defined from the optimal solution $x^\star$ for the non-convex relaxation.

Proposition 4.10. $\frac{1}{2} z_{OPT} \leq \max\{c^T x^L, c^T x^U\}$.

Proof. Since $z_{OPT} \leq \sum_{j \in N} c_j x_j^\star$, the following inequalities hold.

$$z_{OPT} \leq c^T x^L + c^T x^U \leq 2 \max\{c^T x^L, c^T x^U\}$$

Therefore, the result follows. □

Based on Proposition 4.10, we devise a polynomial time 1/2-approximation algorithm for the CKP using Algorithm 11, as described in Algorithm 12.

Algorithm 12 Polynomial time 1/2-approximation algorithm for CKP

$x^\star \leftarrow \arg\max\{c^T x : x \in \mathcal{P}_{NC}\}$ using Algorithm 11 ;
$f \leftarrow$fractional variable index of $x^\star$;
$x_j^L \leftarrow \lfloor x_j^\star \rfloor$ for all $j \in N$;
$x_f^U \leftarrow 1$ and $x_j^U \leftarrow 0$ for all $j \in N \setminus \{f\}$;
$x^A \leftarrow \arg\max\{c^T x^L, c^T x^U\}$;
return x^A ;

Therefore, we can obtain both tight upper and lower bounds for the CKP in polynomial time from the non-convex relaxation using Algorithm 11 and 12, respectively.

The following section proposes an efficient lifting heuristic for probabilistic cover inequalities based on the non-convex relaxation and the proposed adaptive method.

4.5 Lifting heuristic based on the non-convex relaxation

Let a probabilistic cover inequality be given as

$$\sum_{j \in C_P} x_j \leq |C_P| - 1,$$

where C_P is a probabilistic cover. As mentioned in Section 4.1, this inequality can be strengthened to a lifted probabilistic cover inequality which is described as

$$\sum_{j \in C_P} x_j + \sum_{j \in N \setminus C_P} \gamma_j x_j \leq |C_P| - 1.$$

The lifting coefficients γ_j's can be obtained using the sequential lifting technique. Let a sequence of variables in $N \setminus C_P$ be given as $(j_1, ..., j_{n-|C_P|})$. The sequential lifting technique computes each lifting coefficient in order of the given sequence of variables by solving lifting problems. For each $k = 1, ..., n - |C_P|$, the lifting problem to obtain γ_{j_k} can be formulated as another CKP described as follows (Joung & Park, 2017).

$$\beta_{j_k} = \max \quad \sum_{i \in C_P} x_i + \sum_{i \in C_P^{k-1} \setminus C_P} \gamma_i x_i + n x_{j_k}$$

$$\text{s.t} \quad \sum_{i \in C_P^j} a_i x_i + \Phi^{-1}(\epsilon) \sqrt{\sum_{i \in C_P^k} \sigma_i^2 x_i^2} \leq b \tag{4.7}$$
$$x_i \in \{0,1\},\ i \in C_P^k,$$

where $C_P^k = C_P \cup \{j_1, \ldots, j_k\}$ and $C_P^0 = C_P$. Using the optimal objective value β_{j_k}, the lifting coefficient γ_{j_k} is determined as

$$\gamma_{j_k} = |C_P| - 1 + n - \beta_{j_k}.$$

Rather than solving the lifting problem exactly, our lifting heuristic determines the lifting coefficients using the upper bounds of the lifting problems. Specifically, because the lifting problem (4.7) is another CKP, we use the upper bound of β_{j_k} that can be obtained from the non-convex relaxation of the lifting problem.

For each $k = 1, \ldots, n - |C_P|$, the non-convex relaxation of the lifting problem can be formulated as follows.

$$\hat{\beta}_{j_k} = \max \quad \sum_{i \in C_P} x_i + \sum_{i \in C_P^{k-1} \setminus C_P} \beta_i x_i + n x_{j_k}$$
$$\text{s.t} \quad \sum_{i \in C_P^k} a_i x_i + \Phi^{-1}(\epsilon) \sqrt{\sum_{i \in C_P^k} \sigma_i^2 x_i} \leq b \tag{4.8}$$
$$x_i \in [0,1],\ i \in C_P^k.$$

Then, our lifting heuristic determines the lifting coefficient as $\gamma_{j_k} = |C_P| - 1 + n - \lfloor \hat{\beta}_{j_k} \rfloor$. The overall procedure is described in Algorithm 13.

In our lifting heuristic, the relaxation (4.8) can be solved in $O(n^3 \log n)$ using Algorithm 11. Atamtürk & Narayanan (2009) also proposed a lifting heuristic that

Algorithm 13 Lifting heuristic for probabilistic cover inequalities

procedure LIFT(C_P)
  $(j_1, \ldots, j_{n-|C_P|}) \leftarrow N \setminus C_P$; ▷ Variable sequencing
  for $k = 1, \ldots, n - |C_P|$ **do**
    $\hat{\beta}_{j_k} \leftarrow$Solve the relaxation (4.8) using Algorithm 11 ;
    $\gamma_{j_k} \leftarrow |C_P| - 1 + n - \lfloor \hat{\beta}_{j_k} \rfloor$;
  end for
  return $\gamma = (\gamma_{j_1}, \ldots, \gamma_{j_{n-|C_P|}})$;
end procedure

solves this relaxation repeatedly. However, their parametric optimization-based algorithm for the relaxation (4.8) has $O(|C_P|n^3 \log n)$ computational complexity, which is worse than our adaptive method.

Through computational experiments, we demonstrate that our lifting heuristic significantly reduces the computation time spent on the lifting procedure compared to the existing lifting heuristics, including Atamtürk & Narayanan (2009), while the effectiveness of the resulting lifted probabilistic cover inequality remains competitive.

4.6 Computational test results

In this section, we present computational test results on the performance of the proposed lifting heuristic. We compared the performance with two other lifting heuristics proposed by Atamtürk & Narayanan (2009) and Joung & Park (2017), which are denoted as "PL" (Parametric LP) and "RO" (Robust optimization technique) based on the method they used, respectively.

We applied the methods to multi-dimensional chance-constrained binary knapsack problems (MCKP), which are described as

$$\begin{aligned} \text{MCKP:} \quad \max \quad & \sum_{j \in N} c_j x_j \\ \text{s.t} \quad & \sum_{j \in N} a_{kj} x_j + \Phi^{-1}(\epsilon) \sqrt{\sum_{j \in N} \sigma_{kj}^2 x_j^2} \leq b_k, \ k \in R^\star \\ & x \in \{0,1\}^n. \end{aligned}$$

where $|R^\star| = r^\star$. We note that the feasible solution set of the MCKP is defined with $r^\star$ different CKPs. For each n, $r^\star$, and $\Phi^{-1}(\epsilon)$, we generated 10 instances in the same manner with Atamtürk & Narayanan (2009). Then, we reported the average results.

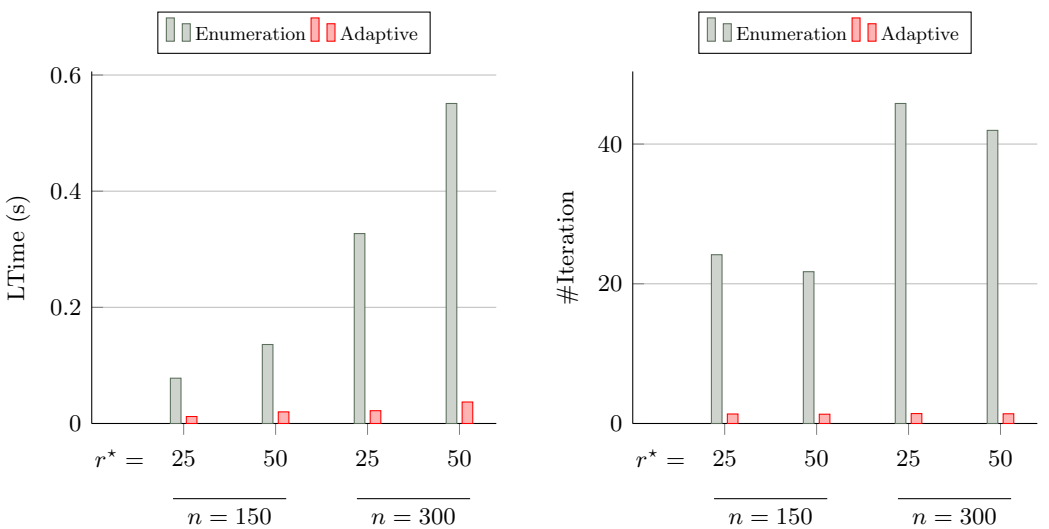
By relaxing the integrality restriction in the MCKP, a continuous relaxation can be obtained, represented as a convex optimization problem. We applied the cutting plane algorithm using probabilistic cover inequalities to the relaxation. In each iteration of the algorithm, at most $r^\star$ different probabilistic covers were obtained, then we applied the lifting methods to the probabilistic cover inequalities.

For the separation of probabilistic cover inequalities, we used the heuristic modifying the method proposed by Gu et al. (1998). Specifically, for a given incumbent

solution $\hat{x}$ in the cutting plane algorithm, we obtained the probabilistic covers in a greedy manner with respect to the values of $1 - \hat{x}_j$. We also used the variable ordering strategy proposed by Gu et al. (1998) to determine the lifting sequence.

We set 300 seconds time limit for the cutting plane algorithm. All algorithms were implemented using C++ with the linear programming solver provided by Xpress (Guéret et al., 2002). All tests were performed using a machine with an Intel Core i7, 3.10GHz CPU, and 16GB RAM.

In the proposed lifting heuristic, the non-convex relaxation of the lifting problem can be solved using both the adaptive method (Algorithm 11) and the enumeration method (Algorithm 10). We first evaluated the efficiency of our lifting heuristic using the adaptive method compared to the enumeration method. For each n and $r^\star$, we tested instances with $\Phi^{-1}(\epsilon) \in \{1, 3, 5\}$, and the average results are reported.



(a) Liting time (s)

(b) Number of the iterations

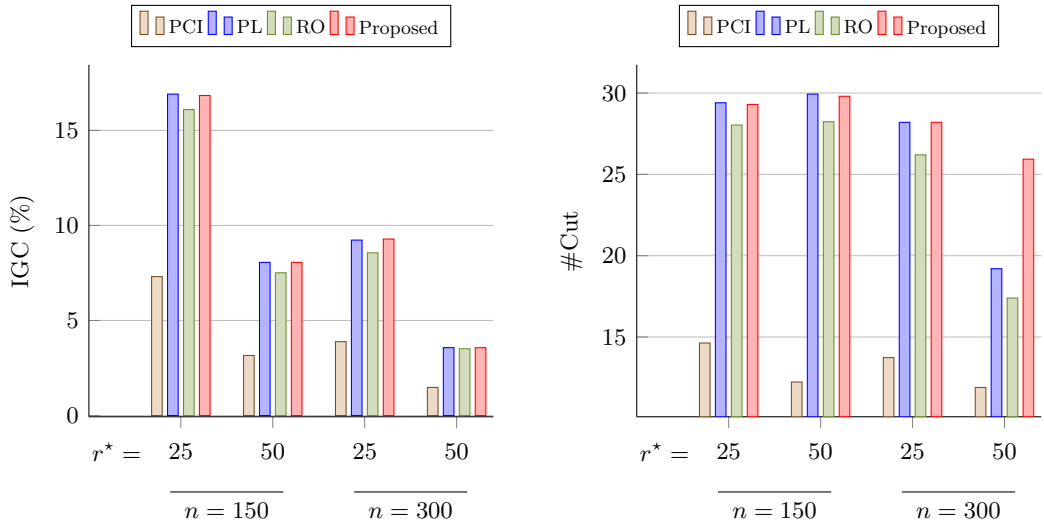
Figure 4.2: LTime (s) and #Iteration by lifting heuristics

Figure 4.2a shows the total computation time spent on the lifting procedure

(Lifting time, LTime) during the cutting plane algorithm. Both the enumeration method and the adaptive method exactly solve the non-convex relaxation for the lifting problem, resulting in the generation of the same lifted probabilistic cover inequalities in the lifting heuristics. However, using the adaptive method could significantly reduce the lifting time compared to using the enumeration method. The difference can be explained by Figure 4.2b, which represents the average number of iterations required to compute each lifting coefficient in each method, indicating the number of elements in Δ investigated in each method. Due to the upper and lower bounds for the optimal $\delta^\star$ proposed in Section 4.4, the adaptive method investigated far fewer elements in Δ compared to the enumeration method. Therefore, using the adaptive method could generate lifted probabilistic cover inequalities in significantly less time than the enumeration method, even though both methods have the same computational complexity in theory.

Next, we compared the performance of the proposed lifting heuristic with the others. Figure 4.3a and 4.3b show the integrality gap closed (%) by generated lifted probabilistic cover inequalities, and the number of generated cuts (#Cut), respectively. Here, "Proposed" refers to our lifting heuristic using the adaptive method, described in Algorithm 13. For comparison, we also reported the performance of probabilistic cover inequalities, denoted as "PCI". For each n and $r^\star$, we tested instances with $\Phi^{-1}(\epsilon) \in \{1, 3, 5\}$, and the average results are represented. The detailed result is presented in Appendix D.

As shown in Figure 4.3a, lifted inequalities can significantly enhance the formulations compared to probabilistic cover inequalities, regardless of the lifting heuristics employed. It is worth noting that the proposed lifting heuristic and PL yield the same



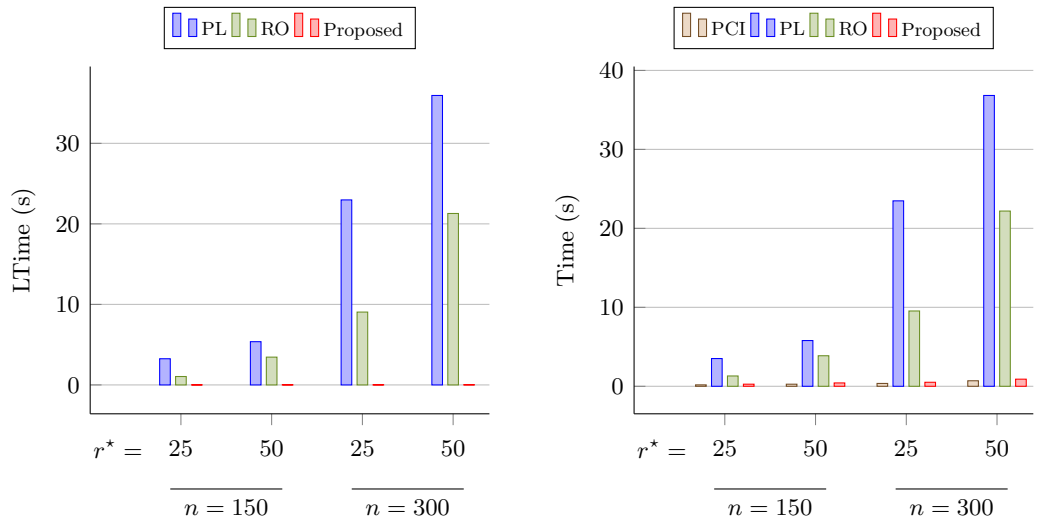
(a) Integrality gap closed (%)

(b) Number of generated cuts

Figure 4.3: IGC (%) and #Cut by lifting heuristics

lifted inequalities since they determine lifting coefficients by solving the same relaxation of lifting problems. While both the proposed lifting heuristic and PL improve the integrality gap more than RO, the difference between them was insignificant. The number of generated lifted probabilistic cover inequalities was also similar.

Figure 4.4a and 4.4b show the lifting and cutting plane time in seconds. As shown in Figure 4.4a, both PL and RO required considerable amounts of time to obtain lifted probabilistic cover inequalities. However, the proposed lifting heuristic was performed extremely quickly, taking less than 0.05 seconds even for $n = 300$. Therefore, the cutting plane algorithm using the proposed heuristic spent comparable computation time compared to using only probabilistic cover inequalities while significantly enhancing the formulations. In conclusion, the proposed lifting heuristic efficiently strengthens probabilistic cover inequalities, and the resulting lifted probabilistic cover inequalities remain competitive compared to those obtained using



(a) Lifting time

(b) Cutting plane time

Figure 4.4: LTime (s) and cutting plane time (s) by lifting heuristics

existing lifting methods.

4.7 Conclusion

In this study, we proposed an efficient lifting heuristic for probabilistic cover inequalities for the chance-constrained binary knapsack problem (CKP). We introduced a non-convex relaxation for the CKP and showed that the relaxation provides a tighter upper bound for the CKP compared to other continuous relaxations. Furthermore, the quality of the obtained upper bound is guaranteed at most twice the optimal objective value of the CKP. Even though the non-convex relaxation is a non-convex optimization problem, we devised a polynomial time algorithm for the relaxation based on its reformulation. We then proposed a lifting heuristic for probabilistic cover inequalities, which sequentially determines the lifting coefficients by solving the non-convex relaxations of the lifting problems using the proposed algorithm.

Computational test results indicated that our lifting heuristic outperforms the existing lifting methods in terms of computational efficiency. At the same time, the generated inequalities' effectiveness remains comparable to the existing lifting methods.

Even though our lifting heuristic has less computational complexity than the one proposed by Atamtürk & Narayanan (2009) in theory, its performance exceeded our expectations. It is necessary to conduct a more elaborate analysis of the computational complexity of the proposed heuristic with tight examples.

Additionally, computational complexities for the problems associated with the lifting procedure are still open questions. Because the lifting problem is a special case of the CKP, it may be solved efficiently as the lifting problem of the binary knapsack problem, even though the CKP is NP-hard. Also, the separation problem for probabilistic cover inequalities has not been seriously studied yet. Hence, this line of research may help tackle chance-constrained programs using cutting plane algorithms.

Chapter 5

Conclusion

5.1 Summary and contributions

In this thesis, we proposed cut generation methods for the two variants of the binary knapsack problem, which can be used in solving general binary integer programs: the binary knapsack problem with generalized upper bounds (GKP) and the chance-constrained binary knapsack problem (CKP).

In Chapter 2, we investigated valid inequalities for the GKP, which can be more effective in solving binary integer linear programs with generalized upper bounds compared to those for the binary knapsack problem. Specifically, we focused on rank-1 CG cuts, CG cuts for shorts, and the separation problem for the GKP, which have not been studied in the literature. We first characterized non-dominated CG cuts for the GKP and analyzed the computational complexity of the separation problem. Based on the results, we presented exact and heuristic algorithms for the separation problem. Although the CG cut separation problem for general integer linear programs is known to be strongly NP-hard, we demonstrated that the separation problem for the GKP can be solved in pseudo-polynomial time through the proposed exact separation algorithm. The performance of the proposed separation algorithms and the effectiveness of the generated CG cuts were evaluated through extensive com-

putational tests using instances for the GKP and general binary integer programs. The test results showed that the proposed separation algorithms outperformed the existing mixed-integer programming approach for the separation problem. Using the proposed heuristic separation algorithm could improve the comparable integrality gap with using the exact separation algorithm within significantly less time. Furthermore, CG cuts generated by the heuristic separation algorithm could provide more enhanced formulations than general lifted GUB cover inequalities within comparable computation times.

In Chapter 3, we presented a novel CG cut strengthening method for binary integer linear programs, which can yield more enhanced formulations using fewer cuts. We first defined maximal CG cuts for binary knapsack polytopes and proposed a method to derive one that dominates given CG cuts for binary integer linear programs. Subsequently, we introduced an extended knapsack polytope where the maximal CG cut can be interpreted as its lifted cover inequality. We then proposed a strengthening method for the maximal CG cut using the lifting function for cover inequalities. Through theoretical comparison, we showed that the obtained cut is at least as strong as the Gomory mixed-integer cut derived from the maximal CG cut. We also extended the method to CG cuts for binary integer linear programs with generalized upper bounds. Computational test results demonstrated that our CG cut strengthening method could provide more enhanced formulations using fewer cuts than using CG cuts. Furthermore, the proposed method could reduce the total computation time for the cutting plane algorithm by decreasing the number of iterations, even though additional computation time was required to strengthen CG cuts. However, for binary integer linear programs with generalized upper bounds,

the performance of the proposed method was indistinguishable from using Gomory mixed-integer cuts.

In Chapter 4, we proposed an efficient lifting heuristic for probabilistic cover inequalities for the CKP, which can be used to solve chance-constrained programs representing optimization problems under uncertainty. We introduced a non-convex relaxation for the CKP and showed that the relaxation provides a tighter upper bound for the CKP compared to existing continuous relaxations. Subsequently, we devised a polynomial time algorithm for the relaxation. We then proposed a lifting heuristic for probabilistic cover inequalities using the non-convex relaxation for the CKP and the proposed algorithm. Computational test results indicated that our lifting heuristic could generate lifted probabilistic cover inequalities in significantly less time than existing lifting methods, while the strength of the generated inequalities remains comparable.

5.2 Future research directions

The results of this thesis offer several future research directions. The heuristic CG cut separation algorithm proposed in Chapter 2 showed promising performance even though we selected the parameter arbitrarily. However, a more careful choice of the parameter by exploring more properties of effective CG cuts may improve the algorithm's performance. Furthermore, rank-1 CG cuts for the GKP can represent useful valid inequalities such as GUB cover inequalities, as shown in Chapter 2. Investigation into the properties of effective CG cuts for the GKP may lead to a new family of strong valid inequalities for the GKP. Our analysis results on the non-dominated CG cuts for the GKP generalized the results for the binary knapsack

problem by Park & Lee (2011). Similarly, one can further generalize our results to more complicated variants of the binary knapsack problem or other sub-structures of general binary integer linear programs.

We showed that the CG cut separation problem for the GKP can be solved in pseudo-polynomial time. This result implies that the separation problem is not strongly NP-hard. However, we failed to reveal the computational complexity of the separation problem, whether it is NP-hard or not. Studies on the computational complexity of the separation problem for the GKP can provide some important insights concerned with CG cuts. For example, studies on the NP-hardness of the CG cut separation problem for the GKP can provide an answer for the NP-hardness of the CG cut separation problem for general binary integer linear programs.

We showed that the strengthening method for maximal CG cuts for binary knapsack polytopes proposed in Chapter 3 yields a new family of valid inequalities for binary knapsack polytopes, namely SCG cuts. We extended the results to the case with generalized upper bounds; however, the effectiveness of the obtained SCG cuts was insignificant in that case. One possible explanation is that the generalized upper bounds are not incorporated explicitly in our method. Herefore, we suggest devising another strengthening method that incorporates the generalized upper bounds explicitly to yield more effective cuts in that case. Furthermore, the proposed strengthening methods can be applied to any CG cuts for binary integer linear programs with or without generalized upper bounds. However, we only presented computational results utilizing the CG cuts derived from each constraint of the test instances. One can evaluate the proposed methods' performances for CG cuts derived from other single-constraint relaxations, such as Gomory fractional cuts derived from the opti-

mal simplex tableau.

A closure using SCG cuts can be defined similarly to the Chvátal closure. Although we showed that the SCG cuts are at least as strong as the Gomory mixed-integer cuts derived from maximal CG cuts, it does not necessarily mean that their closure is tighter than the Gomory mixed-integer closure. Therefore, it may be helpful to analyze the strength of the closure to understand the effectiveness of SCG cuts. Additionally, investigating the separation problem for SCG cuts can be another research direction to enhance the capability of solving general binary integer linear programs.

Regarding the lifting of probabilistic cover inequalities, developing an efficient exact algorithm for the lifting problem may be possible. Moreover, it is necessary to develop exact solution approaches for the CKP that can provide optimal solutions for problems with many items in a reasonable time. In this study, we only considered down-liftings for probabilistic cover inequalities, but introducing up-liftings can lead to more effective cuts that can be used to solve chance-constrained programs. Additionally, we proposed an approximation algorithm for the CKP in this study. Exploring its applications or comparing it with other approximation algorithms are also interesting research directions.

The valid inequalities for the CKP studied in this thesis can be applied to chance-constrained programs under specific distributions of random variables. Valid inequalities for more general chance-constrained programs can also be studied. For example, Küçükyavuz (2012) investigated sub-structures for scenario-based chance-constrained programs and derived a useful family of cuts. This line of research may improve the solvability of optimization problems under uncertainty.

Throughout this thesis, we proposed cut generation methods that can be applied to single-constraint relaxations for general binary integer programs. However, we did not discuss how to obtain single-constraint relaxations. We only focused on those derived from each constraint of the problem. One possible research direction is to propose an adaptive method for constructing the single-constraint relaxation by aggregating the constraints through the cutting plane algorithm. For instance, dual optimal solutions obtained in each iteration of the cutting plane algorithm can be used for binary integer linear programs. Then, more effective cuts may be derived from the relaxation using our cut generation methods, which may significantly improve the performance of general-purpose optimization solvers. Furthermore, it is necessary to evaluate how much the proposed cut generation methods improve the solvability of binary integer linear programs by implementing them in the branch-and-cut framework.

The studies on the binary knapsack problem have been extended to cases with general integral or continuous decision variables, and the generalized results have been a cornerstone in enhancing the capability of solving integer or mixed-integer programs. We hope the results presented in this thesis can also be extended to more general knapsack problems.

Bibliography

Ahmed, Shabbir (2014). "Convex relaxations of chance constrained optimization problems". *Optimization Letters* 8, pp. 1–12.

Atamtürk, Alper & Avinash Bhardwaj (2018). "Network design with probabilistic capacities". *Networks* 71.1, pp. 16–30.

Atamtürk, Alper & Oktay Günlük (2010). "Mingling: mixed-integer rounding with bounds". *Mathematical Programming* 123, pp. 315–338.

Atamtürk, Alper & Kiavash Kianfar (2012). "n-step mingling inequalities: new facets for the mixed-integer knapsack set". *Mathematical programming* 132, pp. 79–98.

Atamtürk, Alper & Vishnu Narayanan (2008). "Polymatroids and mean-risk minimization in discrete optimization". *Operations Research Letters* 36.5, pp. 618–622.

— (2009). "The submodular knapsack polytope". *Discrete Optimization* 6.4, pp. 333–344.

Balas, Egon (1975). "Facets of the knapsack polytope". *Mathematical programming* 8.1, pp. 146–164.

Balas, Egon, Sebastian Ceria, Gérard Cornuéjols & N Natraj (1996). "Gomory cuts revisited". *Operations Research Letters* 19.1, pp. 1–9.

Balas, Egon & Eitan Zemel (1978). "Facets of the knapsack polytope from minimal covers". *SIAM Journal on Applied Mathematics* 34.1, pp. 119–148.

Balintfy, Joseph L., G. Terry Ross, Prabhakant Sinha & Andris A. Zoltners (1978). "A mathematical programming system for preference and compatibility maximized menu planning and scheduling". *Mathematical Programming* 15.1, pp. 63–76.

Basu, Amitabh, Robert Hildebrand & Matthias Köppe (2016a). "Light on the infinite group relaxation I: foundations and taxonomy". *4OR* 14, pp. 1–40.

— (2016b). "Light on the infinite group relaxation II: sufficient conditions for extremality, sequences, and algorithms". *4OR* 14, pp. 107–131.

Beasley, John E. (1990). "OR-Library: distributing test problems by electronic mail". *Journal of the operational research society* 41.11, pp. 1069–1072.

Ben-Tal, Aharon, Laurent El Ghaoui & Arkadi Nemirovski (2009). *Robust optimization*. Vol. 28. Princeton university press.

Bertsimas, Dimitris, David B. Brown & Constantine Caramanis (2011). "Theory and Applications of Robust Optimization". *SIAM Review* 53.3, pp. 464–501.

Bertsimas, Dimitris & Melvyn Sim (2004). "The price of robustness". *Operations research* 52.1, pp. 35–53.

Birge, John R & Francois Louveaux (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Bixby, Robert E, Sebastian Ceria, Cassandra M McZeal & Martin WP Savelsbergh (1998). *An updated mixed integer programming library: MIPLIB 3.0*. Tech. rep.

Bixby, Robert E., Mary Fenelon, Zonghao Gu, Ed Rothberg & Roland Wunderling (2004). "Mixed-integer programming: A progress report". *The sharpest cut: the impact of Manfred Padberg and his work*. SIAM, pp. 309–325.

Burdet, Claude-Alain & Ellis L Johnson (1975). "A subadditive approach to solve linear integer programs". *Annals of Discrete Mathematics* 1, pp. 117–144.

Cacchiani, Valentina, Manuel Iori, Alberto Locatelli & Silvano Martello (2022). "Knapsack problems-An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems". *Computers & Operations Research*, p. 105693.

Caprara, Alberto & Matteo Fischetti (1996). "{0, 1/2}-Chvátal-Gomory cuts". *Mathematical Programming* 74, pp. 221–235.

Caprara, Alberto & Adam N Letchford (2003). "On the separation of split cuts and related inequalities". *Mathematical Programming* 94, pp. 279–294.

Cavalcante, Cristina C. B., C. Carvalho De Souza, Martin W. P. Savelsbergh, Yaoguang Wang & Laurence A. Wolsey (2001). "Scheduling projects with labor constraints". *Discrete Applied Mathematics* 112.1-3, pp. 27–52.

Charnes, Abraham, William W Cooper & Gifford H Symonds (1958). "Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil". *Management science* 4.3, pp. 235–263.

Chvátal, Vasek (1973). "Edmonds polytopes and a hierarchy of combinatorial problems". *Discrete mathematics* 4.4, pp. 305–337.

Conforti, Michele, Gérard Cornuéjols, Aris Daniilidis, Claude Lemaréchal & Jérôme Malick (2015). "Cut-generating functions and S-free sets". *Mathematics of Operations Research* 40.2, pp. 276–391.

Conforti, Michele, Cornuéjols Gérard & Giacomo Zambelli (2014). *Integer Programming*. Springer International Publishing.

Cook, William, Ravindran Kannan & Alexander Schrijver (1990). "Chvátal closures for mixed integer programming problems". *Mathematical Programming* 47.1-3, pp. 155–174.

Cornuéjols, Gérard et al. (2007). "Revival of the Gomory cuts in the 1990's." *Annals of Operations Research* 149.1, pp. 63–66.

Crowder, Harlan, Ellis L Johnson & Manfred Padberg (1983). "Solving large-scale zero-one linear programming problems". *Operations Research* 31.5, pp. 803–834.

D'Andreagiovanni, Fabio, Carlo Mannino & Antonio Sassano (2013). "GUB covers and power-indexed formulations for wireless network design". *Management Science* 59.1, pp. 142–156.

Dantzig, George B (1955). "Linear programming under uncertainty". *Management science* 1.3-4, pp. 197–206.

Dey, Santanu S & Jean-Philippe Richard (2009). "Linear-programming-based lifting and its application to primal cutting-plane algorithms". *INFORMS Journal on Computing* 21.1, pp. 137–150.

Dey, Santanu S & Andrea Tramontani (2009). "Recent developments in multi-row cuts". *Optima* 80, pp. 2–8.

Edmond, Jack (1970). "Submodular functions, matroids, and certain polyhedra". *Combinatorial Structures and Their Applications*. Gordon and Breach, Louvain, pp. 69–87.

Edmonds, Jack (1965). "Maximum matching and a polyhedron with 0, 1-vertices". *Journal of research of the National Bureau of Standards B* 69.125-130, pp. 55–56.

Eisenbrand, Friedrich (1999). "Note-on the membership problem for the elementary closure of a polyhedron". *Combinatorica* 19.2, pp. 297–300.

Fischetti, Matteo & Andrea Lodi (2007). "Optimizing over the first Chvátal closure". *Mathematical Programming* 110.1, pp. 3–20.

Gabrel, Virginie, Cécile Murat & Aurélie Thiele (2014). "Recent advances in robust optimization: An overview". *European journal of operational research* 235.3, pp. 471–483.

Garey, Michael R. & David S. Johnson (1979). *Computers and intractability*. Vol. 174. freeman San Francisco.

Glover, Fred & Darwin Klingman (1979). "A O(nlogn) algorithm for LP Knapsacks with GUB constraints". *Mathematical Programming* 17.1, pp. 345–361.

Glover, Fred, Hanif D. Sherali & Youngho Lee (1997). "Generating Cuts from Surrogate Constraint Analysis for Zero-One and Multiple Choice Programming". *Computational Optimization and Applications* 8.2, pp. 151–172.

Gomory, Ralph (1958). "Outline of an algorithm for integer solutions to linear programs". *Bulletin of the American Mathematical Society* 64.5, pp. 275–278.

— (1960). "An Algorithm for the mixed Integer Problem". *Technical Report RM-2597, The Rand Corporation*.

Gomory, Ralph E & Ellis L Johnson (1972a). "Some continuous functions related to corner polyhedra". *Mathematical Programming* 3, pp. 23–85.

— (1972b). "Some continuous functions related to corner polyhedra, II". *Mathematical Programming* 3, pp. 359–389.

Goyal, Vineet & R. Ravi (2010). "A PTAS for the chance-constrained knapsack problem with random item sizes". *Operations Research Letters* 38.3, pp. 161–164.

Gu, Zonghao, George L Nemhauser & Martin WP Savelsbergh (1998). "Lifted cover inequalities for 0-1 integer programs: Computation". *INFORMS Journal on Computing* 10.4, pp. 427–437.

— (2000). "Sequence independent lifting in mixed integer programming". *Journal of Combinatorial Optimization* 4, pp. 109–129.

Guéret, Christelle, Christian Prins & Marc Sevaux (2002). *Applications of optimization with Xpress-MP*. Dash Optimization Limited.

Hammer, Peter L, Ellis L Johnson & Uri N Peled (1975). "Facet of regular 0–1 polytopes". *Mathematical Programming* 8.1, pp. 179–206.

Han, Jinil, Kyungsik Lee, Chungmok Lee, Ki-Seok Choi & Sungsoo Park (2016). "Robust optimization approach for a chance-constrained binary knapsack problem". *Mathematical Programming* 157.1, pp. 277–296.

Hojny, Christopher, Tristan Gally, Oliver Habeck, Hendrik Lüthen, Frederic Matter, Marc E Pfetsch & Andreas Schmitt (2020). "Knapsack polytopes: a survey". *Annals of Operations Research* 292, pp. 469–517.

Johnson, Ellis L & Manfred W Padberg (1981). "A note of the knapsack problem with special ordered sets". *Operations Research Letters* 1.1, pp. 18–22.

Joung, Seulgi & Kyungsik Lee (2020). "Robust optimization-based heuristic algorithm for the chance-constrained knapsack problem using submodularity". *Optimization Letters* 14.1, pp. 101–113.

Joung, Seulgi & Sungsoo Park (2017). "Lifting of probabilistic cover inequalities". *Operations Research Letters* 45.5, pp. 513–518.

Kall, Peter, Stein W Wallace & Peter Kall (1994). *Stochastic programming*. Vol. 6. Springer.

Kaparis, Konstantinos & Adam N Letchford (2010). "Separation algorithms for 0-1 knapsack polytopes". *Mathematical programming* 124, pp. 69–91.

Kellerer, H., U. Pferschy & D. Pisinger (2004). *Knapsack Problems*. Springer, Berlin, Germany.

Kim, Junyoung, Byungju Goo, Youngjoo Roh, Chungmok Lee & Kyungsik Lee (2023). "A branch-and-price approach for airport gate assignment problem with chance constraints". *Transportation Research Part B: Methodological* 168, pp. 1–26.

Klabjan, Diego, George L Nemhauser & Craig Tovey (1998). "The complexity of cover inequality separation". *Operations Research Letters* 23.1-2, pp. 35–40.

Küçükyavuz, Simge (2012). "On mixing sets arising in chance-constrained programming". *Mathematical programming* 132.1-2, pp. 31–56.

Land, A. H. & A. G. Doig (1960). "An Automatic Method of Solving Discrete Programming Problems". *Econometrica* 28.3, pp. 497–520.

Lee, Dabeen (2019). "On the NP-hardness of deciding emptiness of the split closure of a rational polytope in the 0, 1 hypercube". *Discrete Optimization* 32, pp. 11–18.

Lee, Kyungsik (2012). "Separation Heuristic for the Rank-1 Chvatal-Gomory Inequalities for the Binary Knapsack Problem". *Journal of Korean Institute of Industrial Engineers* 38.2, pp. 74–79.

Letchford, Adam N & Andrea Lodi (2002). "Strengthening Chvátal–Gomory cuts and Gomory fractional cuts". *Operations Research Letters* 30.2, pp. 74–82.

Letchford, Adam N & Georgia Souli (2019). "On lifted cover inequalities: A new lifting procedure with unusual properties". *Operations Research Letters* 47.2, pp. 83–87.

Luedtke, James, Shabbir Ahmed & George L Nemhauser (2010). "An integer programming approach for linear programs with probabilistic constraints". *Mathematical programming* 122.2, pp. 247–272.

Marchand, Hugues, Alexander Martin, Robert Weismantel & Laurence Wolsey (2002). "Cutting planes in integer and mixed integer programming". *Discrete Applied Mathematics* 123.1-3, pp. 397–446.

Marchand, Hugues & Laurence A Wolsey (2001). "Aggregation and mixed integer rounding to solve MIPs". *Operations research* 49.3, pp. 363–371.

Martello, Silvano & Paolo Toth (1990). *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc.

Miller, Bruce L & Harvey M Wagner (1965). "Chance constrained programming with joint constraints". *Operations Research* 13.6, pp. 930–945.

Nemhauser, George L & Laurence A Wolsey (1990). "A recursive procedure to generate all cuts for 0–1 mixed integer programs". *Mathematical Programming* 46.1-3, pp. 379–390.

Nemhauser, George L. & Pamela H. Vance (1994). "Lifted cover facets of the 0–1 knapsack polytope with GUB constraints". *Operations Research Letters* 16.5, pp. 255–263.

Nemirovski, Arkadi (2012). "On safe tractable approximations of chance constraints". *European Journal of Operational Research* 219.3, pp. 707–718.

Nemirovski, Arkadi & Alexander Shapiro (2007). "Convex approximations of chance constrained programs". *SIAM Journal on Optimization* 17.4, pp. 969–996.

Padberg, Manfred W (1973). "On the facial structure of set packing polyhedra". *Mathematical programming* 5.1, pp. 199–215.

— (1975). "A note on zero-one programming". *Operations Research* 23.4, pp. 833–837.

Park, Kyungchul & Kyungsik Lee (2011). "On the Separation of the Rank-1 Chvatal-Gomory Inequalities for the Fixed-Charge 0-1 Knapsack Problem". *Journal of the Korean Operations Research and Management Science Society* 36.2, pp. 43–50.

Powell, Warren B (2019). "A unified framework for stochastic optimization". *European Journal of Operational Research* 275.3, pp. 795–821.

Prékopa, András (1970). "On probabilistic constrained programming". *Proceedings of the Princeton symposium on mathematical programming*. Vol. 113. Princeton, NJ, p. 138.

— (2013). *Stochastic programming*. Vol. 324. Springer Science & Business Media.

Ruszczyński, Andrzej (2002). "Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra". *Mathematical Programming* 93, pp. 195–215.

Sankaran, Jayaram K., Dennis L. Bricker & Shuw-Hwey Juang (1999). "A strong fractional cutting-plane algorithm for resource-constrained project scheduling". *International Journal of Industrial Engineering* 6, pp. 99–111.

Schrijver, A (1980). "On cutting planes". *Annals of Discrete Mathematics* 9, pp. 291–296.

Shapiro, Alexander, Darinka Dentcheva & Andrzej Ruszczynski (2021). *Lectures on stochastic programming: modeling and theory*. SIAM.

Sherali, Hanif D. & Youngho Lee (1995). "Sequential and simultaneous liftings of minimal cover inequalities for generalized upper bound constrained knapsack polytopes". *SIAM Journal on Discrete Mathematics* 8.1, pp. 133–153.

Sinha, Prabhakant & Andris A. Zoltners (1979). "The multiple-choice knapsack problem". *Operations Research* 27.3, pp. 503–515.

Sousa, Jorge P & Laurence A Wolsey (1992). "A time indexed formulation of non-preemptive single machine scheduling problems". *Mathematical programming* 54, pp. 353–367.

Tanner, Matthew W. & Lewis Ntaimo (2010). "IIS branch-and-cut for joint chance-constrained stochastic programs and application to optimal vaccine allocation". *European Journal of Operational Research* 207.1, pp. 290–296.

Van Roy, Tony J & Laurence A Wolsey (1987). "Solving mixed integer programming problems using automatic reformulation". *Operations Research* 35.1, pp. 45–57.

Wolsey, Laurence A (1977). "Valid inequalities and superadditivity for 0–1 integer programs". *Mathematics of Operations Research* 2.1, pp. 66–77.

— (1975). "Faces for a linear inequality in 0–1 variables". *Mathematical Programming* 8.1, pp. 165–178.

— (1990). "Valid inequalities for 0–1 knapsacks and mips with generalised upper bound constraints". *Discrete Applied Mathematics* 29.2-3, pp. 251–261.

Yang, Fan & Nilanjan Chakraborty (2018). "Algorithm for optimal chance constrained knapsack problem with applications to multi-robot teaming". *2018*

IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 1043–1049.

Zemel, Eitan (1978). "Lifting the facets of zero–one polytopes". *Mathematical Programming* 15, pp. 268–277.

— (1989). "Easily computable facets of the knapsack polytope". *Mathematics of Operations Research* 14.4, pp. 760–764.

Appendix A

Summary of benchmark instances

In this appendix, we give the summary of benchmark instances used in experiments.

We refer to the column headed 'z_{LP}' and 'z_{OPT}' as the optimal objective values of the LP relaxation and the optimal objective value to the problem. If z_{OPT} is unknown for some instances, we report 'z_B' and 'z_U' which means the best solution and best upper bound, respectively, instead of z_{OPT}. 'z_B' and 'z_U' are obtained after solving each instance with the MIP solver provided by Xpress for 1,800 seconds. The optimality gap (Gap) is computed as follows:

$$\text{Gap}(\%) = \frac{z_U - z_B}{z_U} * 100$$

Table A.1: Summary of MMKP instances

Name	m	n_i	$r^\star$	z_{LP}	z_B	z_U	Gap (%)
I01	5	5	5	182.7	173.0	173.0	0.00
I02	10	5	5	365.6	364.0	364.0	0.00
I03	15	10	10	1626.6	1602.0	1602.0	0.00
I04	20	10	10	3631.4	3597.0	3597.0	0.00
I05	25	10	10	3905.9	3905.7	3905.7	0.00
I06	30	10	10	4812.8	4799.3	4799.3	0.00
I07	100	10	10	24608.0	24595.0	24598.9	0.02
I08	150	10	10	36904.4	36889.0	36899.4	0.03
I09	200	10	10	49193.9	49175.0	49190.8	0.03
I10	250	10	10	61486.3	61474.0	61483.9	0.02
I11	300	10	10	73797.7	73779.0	73795.6	0.02
I12	350	10	10	86100.5	86091.0	86098.0	0.01
I13	400	10	10	98448.6	98431.0	98446.7	0.02

Table A.2: Summary of MIPLIB instances

Name	n	m	$r^\star$	z_{LP}	z_{OPT}
bm23	27	27	20	20.57	34.00
l152lav	1989	1321	97	4656.36	4722.00
lp4l	1086	812	85	2942.50	2967.00
lseu	89	39	28	834.68	1120.00
manna81	3321	3321	6480	-13297.00	-13164.00
misc03	160	133	96	1910.00	3360.00
misc07	260	221	212	1415.00	2810.00
mitre	10724	383	2054	114782.47	115155.00
mod008	319	319	6	290.93	307.00
mod010	2655	1229	146	6532.08	6548.00
p0033	33	23	16	2520.57	3089.00
p0040	40	10	23	61796.55	62027.00
p0201	201	41	133	6875.00	7615.00
p0282	282	275	241	176867.50	258411.00
p0291	291	285	252	1705.13	5223.75
p0548	548	520	176	539.16	8691.00
p2756	2756	2007	755	2698.95	3124.00
pipex	48	16	25	773.75	788.26
protfold	1835	169	2112	-41.96	-25.00
sentoy	60	60	30	-7839.28	-7772.00
sp97ar	14101	1212	1761	652560384.00	660705645.50
stein27	27	27	118	13.00	18.00
stein45	45	45	331	22.00	30.00

Table A.3: Summary of GAP instances of A, B, and C classes

Name	Class	n	$r^\star$	z_{LP}	z_B	z_U	Gap (%)
a05100	A	5	100	3402.273	3402	3402	0.00
a05200	A	5	200	6965.261	6965	6965	0.00
a10100	A	10	100	3741.443	3740	3740	0.00
a20200	A	20	200	7862.673	7861	7861	0.00
b05100	B	5	100	3276.211	3265	3265	0.00
b05200	B	5	200	6652.588	6648	6648	0.00
b10100	B	10	100	3699.328	3693	3693	0.00
b10200	B	10	200	7384.949	7373	7373	0.00
b20100	B	20	100	3944.819	3934	3934	0.00
b20200	B	20	200	7868.862	7861	7861	0.00
c05100	C	5	100	3177.175	3170	3170	0.00
c05200	C	5	200	6749.235	6744	6744	0.00
c10100	C	10	100	3712.99	3698	3698	0.00
c10200	C	10	200	7404.592	7394	7394	0.00
c10400	C	10	400	14808.9	14803	14804	0.01
c15900	C	15	900	34563.43	34558	34561	0.01
c20100	C	20	100	3881.013	3857	3857	0.00
c201600	C	20	1600	62801.44	62793	62799	0.01
c20200	C	20	200	7823.095	7809	7809	0.00
c20400	C	20	400	15625.85	15618	15619	0.01
c30900	C	30	900	35925.32	35912	35919.12	0.02
c401600	C	40	1600	64460.68	64448	64457.15	0.01
c40400	C	40	400	16168.02	16155	16156	0.01
c60900	C	60	900	36584.96	36549	36576.1	0.07
c801600	C	80	1600	65317	65193	65317	0.19

Table A.4: Summary of GAP instances of C and D classes

Name	Class	n	$r^\star$	z_{LP}	z_B	z_U	Gap (%)
d05100	D	5	100	5654.587	5647	5647	0.00
d05200	D	5	200	11063.8	11058	11058	0.00
d10100	D	10	100	5676.544	5650	5659.587	0.17
d10200	D	10	200	11781.64	11760	11775.56	0.13
d10400	D	10	400	23444.01	23428	23441.91	0.06
d15900	D	15	900	53499.53	53468	53498.39	0.06
d20100	D	20	100	5857.47	5777	5830.282	0.91
d201600	D	20	1600	95778.65	95655	95777.71	0.13
d20200	D	20	200	11982.31	11918	11973.92	0.47
d20400	D	20	400	23847.56	23767	23841.96	0.31
d30900	D	30	900	54071.25	53839	54068.96	0.43
d401600	D	40	1600	96495	96218	96495	0.29
d40400	D	40	400	24052.39	23797	24051	1.06
d60900	D	60	900	54349	53970	54349	0.70
d801600	D	80	1600	96566	96106	96566	0.48
e05100	E	5	100	87458.58	87419	87427	0.01
e05200	E	5	200	175078	175056	175073	0.01
e10100	E	10	100	88456.95	88423	88430	0.01
e10200	E	10	200	176906.1	176880	176896	0.01
e10400	E	10	400	354660.8	354626	354655	0.01
e15900	E	15	900	798483.4	798435	798480	0.01
e20100	E	20	100	91740.42	91658	91666	0.01
e201600	E	20	1600	1420960	1420909	1420956	0.00
e20200	E	20	200	177844.1	177821	177822	0.00
e20400	E	20	400	355538.2	355508	355524	0.00
e30900	E	30	900	800486.7	800444	800473	0.00
e401600	E	40	1600	1423317	1423280	1423307	0.00
e40400	E	40	400	355876.6	355808	355842	0.01
e60900	E	60	900	800796.7	800728	800753	0.00
e801600	E	80	1600	1424819	1424752	1424781	0.00

Appendix B

Detailed experiment results in Chapter 2

B.1 Small-sized GKP instances

We report the detailed experiment results in Section 2.6.1.

Table B.1: Integrality gap closed (%) and separation time (s) for each separation algorithm

n	m	n_i	$\bar{a} = 100$					$\bar{a} = 200$				
			MIP	ECG	HCG			MIP	ECG	HCG		
					S1	S2	S3			S1	S2	S3
50	5	10	99.83	99.72	99.27	99.14	99.60	100.00	100.00	100.00	100.00	100.00
	10	5	98.86	98.86	96.86	96.77	98.30	99.56	99.56	98.14	96.19	98.46
100	5	20	100.00	100.00	99.58	99.41	100.00	99.93	99.93	99.52	97.44	99.66
	10	10	*98.56	98.81	97.48	96.41	98.21	98.83	98.83	98.47	96.94	98.76
	20	5	*63.97	99.38	96.85	92.84	98.60	*66.20	98.43	96.13	93.21	97.98
Average IGC %			92.24	99.35	98.01	96.91	98.94	92.91	99.35	98.45	96.76	98.97

n	m	n_i	$\bar{a} = 100$					$\bar{a} = 200$				
			MIP	ECG	HCG			MIP	ECG	HCG		
					S1	S2	S3			S1	S2	S3
50	5	10	0.127	0.084	0.000	0.000	0.000	0.180	0.302	0.000	0.000	0.000
	10	5	0.397	0.264	0.000	0.000	0.000	0.447	1.046	0.000	0.000	0.000
100	5	20	0.273	0.169	0.000	0.000	0.000	0.410	0.717	0.000	0.000	0.000
	10	10	*1.079	0.497	0.000	0.000	0.001	1.081	1.976	0.000	0.000	0.001
	20	5	*122.9	1.580	0.000	0.001	0.001	*123.8	7.219	0.000	0.001	0.001
Average time (s)			28.041	0.519	0.000	0.000	0.001	25.182	2.252	0.000	0.000	0.001

Table B.2: Number of generated cuts and cutting plane time (s) for each cutting plane algorithm

n	m	n_i	$\bar{a} = 100$					$\bar{a} = 200$				
			MIP	ECG	HCG			MIP	ECG	HCG		
					S1	S2	S3			S1	S2	S3
50	5	10	7.7	8.2	8.2	8.0	8.0	7.2	7.2	6.5	6.6	6.3
	10	5	9.4	8.4	7.6	7.1	7.6	8.8	9.1	8.2	7.6	7.9
100	5	20	13.2	10.7	12.5	11.6	11.8	14.7	15.1	14.8	14.4	14.4
	10	10	*26.0	11.1	10.4	10.7	10.9	10.8	10.8	10.5	9.7	10.4
	20	5	*5.1	14.6	12.4	12.5	12.9	*5.3	10.4	10.2	9.8	10.0
Average #Cut			12.3	10.6	10.2	10.0	10.2	9.4	10.5	10.0	9.6	9.8

n	m	n_i	$\bar{a} = 100$					$\bar{a} = 200$				
			MIP	ECG	HCG			MIP	ECG	HCG		
					S1	S2	S3			S1	S2	S3
50	5	10	1.01	0.80	0.04	0.03	0.04	1.11	2.27	0.03	0.03	0.03
	10	5	4.11	2.23	0.03	0.02	0.03	3.31	9.62	0.03	0.02	0.03
100	5	20	5.06	2.37	0.05	0.04	0.05	8.94	14.81	0.06	0.07	0.07
	10	10	*32.24	5.84	0.05	0.05	0.05	12.39	22.94	0.05	0.04	0.05
	20	5	*547.56	23.38	0.06	0.04	0.06	*599.26	73.38	0.04	0.04	0.05
Average time(s)			134.36	6.93	0.04	0.04	0.05	125.00	24.60	0.04	0.04	0.05

B.2 Large-sized GKP instances

We report the detailed experiment results in Section 2.6.2.

Table B.3: Integrality gap closed (%) by cutting plane algorithms

n	m	n_i	$\bar{a} = 100$				$\bar{a} = 200$			
			LGCI	HCG			LGCI	HCG		
				S1	S2	S3		S1	S2	S3
200	10	20	49.02	95.31	90.68	97.83	39.37	96.48	91.49	98.11
	20	10	37.55	95.05	88.96	96.57	36.94	92.55	87.52	95.86
500	10	50	35.06	89.82	76.93	91.73	37.01	92.38	83.02	95.90
	20	25	31.04	86.09	68.86	90.89	31.46	93.38	83.73	95.77
	50	10	34.49	92.07	84.68	95.37	25.17	93.28	85.04	94.92
1000	10	100	55.17	88.54	80.60	91.42	38.84	89.00	74.87	91.60
	20	50	34.88	91.85	77.29	93.04	39.07	87.38	74.62	91.38
	50	20	31.23	90.66	79.54	94.19	33.54	93.39	83.49	94.88
	100	10	30.42	88.94	78.96	92.34	21.83	85.12	72.86	88.59
2000	10	200	33.61	77.08	56.93	78.60	35.13	81.71	70.33	86.35
	20	100	23.35	66.04	39.93	68.89	37.70	82.70	63.72	88.63
	50	40	14.43	91.94	82.14	92.63	18.57	82.94	63.12	87.62
	100	20	20.68	84.12	70.63	86.42	23.58	74.83	57.86	79.53
	200	10	13.08	66.93	54.59	70.12	11.07	49.62	41.21	51.60
5000	10	500	33.42	*56.89	43.90	*58.91	35.19	78.55	*47.21	81.20
	20	250	25.25	*53.68	41.65	*53.42	43.26	*72.57	53.44	*75.56
	50	100	10.07	*70.48	*70.83	*73.78	15.00	*69.29	53.57	*72.46
	100	50	10.14	*92.74	*86.00	*93.20	9.79	*62.87	*51.13	*63.55
	200	25	6.29	*78.71	*63.10	*78.71	*8.69	*37.89	32.20	*38.66
	500	10	*6.07	*22.27	20.24	*22.15	*7.50	*27.99	21.37	*29.06
Average			26.76	78.96	67.82	81.01	27.44	77.20	64.59	80.06

Table B.4: Cutting plane time (s)

n	m	n_i	$\bar{a} = 100$				$\bar{a} = 200$			
			LGCI	HCG			LGCI	HCG		
				S1	S2	S3		S1	S2	S3
200	10	20	0.0001	0.0005	0.0005	0.0008	0.0001	0.0004	0.0005	0.0008
	20	10	0.0004	0.0008	0.0009	0.0016	0.0004	0.0009	0.0010	0.0018
500	10	50	0.0002	0.0009	0.0008	0.0013	0.0002	0.0010	0.0009	0.0014
	20	25	0.0007	0.0023	0.0025	0.0040	0.0007	0.0028	0.0030	0.0048
	50	10	0.0049	0.0030	0.0039	0.0066	0.0048	0.0036	0.0046	0.0078
1000	10	100	0.0004	0.0012	0.0012	0.0016	0.0004	0.0019	0.0015	0.0024
	20	50	0.0013	0.0033	0.0034	0.0048	0.0013	0.0043	0.0043	0.0065
	50	20	0.0079	0.0077	0.0094	0.0143	0.0078	0.0097	0.0116	0.0178
	100	10	0.0380	0.0086	0.0118	0.0200	0.0384	0.0115	0.0158	0.0263
2000	10	200	0.0008	0.0027	0.0023	0.0031	0.0008	0.0033	0.0029	0.0039
	20	100	0.0024	0.0059	0.0055	0.0076	0.0024	0.0070	0.0067	0.0090
	50	40	0.0144	0.0212	0.0234	0.0324	0.0145	0.0295	0.0320	0.0459
	100	20	0.0606	0.0234	0.0286	0.0448	0.0629	0.0328	0.0413	0.0632
	200	10	0.3037	0.0225	0.0325	0.0529	0.3052	0.0346	0.0502	0.0814
5000	10	500	0.0018	0.0066	0.0055	0.0073	0.0018	0.0077	0.0061	0.0094
	20	250	0.0055	0.0133	0.0123	0.0152	0.0056	0.0144	0.0136	0.0168
	50	100	0.0337	0.0481	0.0501	0.0619	0.0332	0.0651	0.0679	0.0842
	100	50	0.1418	0.0869	0.0985	0.1321	0.1414	0.1282	0.1435	0.1922
	200	25	0.6071	0.0811	0.1028	0.1499	0.6056	0.1329	0.1696	0.2464
	500	10	5.3871	0.0638	0.0969	0.1526	5.3986	0.1216	0.1880	0.2952
Average			0.3306	0.0202	0.0246	0.0357	0.3313	0.0306	0.0382	0.0559

Table B.5: Number of generated cuts

n	m	n_i	$\bar{a} = 100$				$\bar{a} = 200$			
			LGCI	HCG			LGCI	HCG		
				S1	S2	S3		S1	S2	S3
200	10	20	9.1	20.5	17.1	21.7	10.9	15.7	15.1	17.4
	20	10	9.9	17.2	13.7	17.4	8.0	17.0	14.1	19.9
500	10	50	13.5	32.7	24.5	36.4	10.6	29.5	20.2	34.6
	20	25	14.2	31.4	22.5	35.8	11.9	27.2	19.7	31.2
	50	10	10.4	17.7	15.8	19.6	9.9	17.7	14.4	18.6
1000	10	100	12.2	35.2	27.6	39.4	18.2	44.0	27.5	48.8
	20	50	18.4	42.1	25.0	44.1	12.5	36.2	23.0	45.8
	50	20	17.2	24.6	17.9	32.3	10.4	20.5	15.7	21.7
	100	10	10.2	19.7	13.0	20.1	11.1	21.2	16.3	23.4
2000	10	200	20.3	73.8	45.3	70.7	16.4	58.5	40.0	62.3
	20	100	21.6	105.1	45.6	131.9	20.1	56.0	37.4	66.4
	50	40	15.8	76.1	38.5	76.8	18.9	60.3	27.4	62.3
	100	20	15.0	73.7	38.5	90.6	15.1	31.1	20.0	32.1
	200	10	16.4	46.7	26.3	62.1	10.5	25.2	17.1	29.5
5000	10	500	27.5	*163.5	75.6	*185.1	32.8	123.3	*52.1	143.7
	20	250	22.3	*193.8	99.0	*212.3	15.6	*117.2	64.7	*144.7
	50	100	28.3	*215.8	*118.7	*197.7	25.9	*176.1	92.0	*188.7
	100	50	26.9	*116.0	*79.1	*97.3	29.0	*104.5	*65.4	*100.1
	200	25	20.2	*157.2	*94.6	*155.1	*20.2	*65.8	37.5	*68.9
	500	10	*8.9	*68.1	36.8	*80.6	*9.6	*41.7	22.5	*47.1
Average			16.9	76.5	43.8	81.4	15.9	54.4	32.1	60.4

Table B.6: Separation time (s)

n	m	n_i	$\bar{a} = 100$				$\bar{a} = 200$			
			LGCI	HCG			LGCI	HCG		
				S1	S2	S3		S1	S2	S3
200	10	20	0.001	0.012	0.010	0.022	0.001	0.008	0.009	0.016
	20	10	0.004	0.015	0.013	0.031	0.003	0.016	0.015	0.039
500	10	50	0.003	0.034	0.022	0.056	0.003	0.035	0.021	0.063
	20	25	0.010	0.077	0.055	0.162	0.008	0.087	0.062	0.185
	50	10	0.051	0.053	0.061	0.129	0.048	0.065	0.065	0.149
1000	10	100	0.005	0.049	0.036	0.077	0.008	0.102	0.045	0.158
	20	50	0.023	0.156	0.088	0.240	0.016	0.165	0.102	0.348
	50	20	0.135	0.188	0.163	0.455	0.081	0.204	0.182	0.397
	100	10	0.386	0.169	0.152	0.396	0.431	0.241	0.253	0.607
2000	10	200	0.016	0.224	0.107	0.258	0.013	0.228	0.118	0.285
	20	100	0.051	0.624	0.249	1.101	0.047	0.444	0.245	0.673
	50	40	0.226	1.678	0.890	2.543	0.271	1.859	0.872	2.976
	100	20	0.903	1.704	1.072	3.985	0.949	0.997	0.799	1.968
	200	10	4.988	1.037	0.840	3.224	3.209	0.869	0.851	2.385
5000	10	500	0.049	*1.176	0.425	*1.479	0.058	1.062	*0.322	1.616
	20	250	0.122	*2.594	1.184	*3.259	0.087	*1.780	0.885	*2.726
	50	100	0.962	*10.449	*5.920	*12.467	0.860	*11.572	6.219	*16.108
	100	50	3.806	*9.721	*7.454	*12.345	4.077	*13.290	9.157	*19.008
	200	25	12.275	*12.805	*9.666	*23.319	*12.282	*8.694	*6.270	*16.884
	500	10	*47.725	*4.196	3.434	*11.913	*51.834	*4.902	4.110	*13.452
Average			3.587	2.348	1.592	3.873	3.714	2.331	1.530	4.002

Appendix C

Detailed experiment results in Chapter 3

C.1 GAP instances

We report the detailed experiment results using GAP instances in Section 3.6.

Table C.1: Results on GAP instances of A, B, and C classes using maximal CG cuts

Name	HCG			MCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
a05100	100.00	1	0.006	0.00	0.00	-1.82	0.000
a05200	100.00	3	0.020	0.00	0.00	4.62	0.000
a10100	100.00	18	0.049	0.00	0.00	1.23	0.000
a20200	100.00	27	0.380	0.00	-18.52	2.42	0.000
Average				0.00	-4.63	1.61	0.000
b05100	53.81	58	0.077	-1.09	-12.07	20.08	0.000
b05200	33.78	65	0.237	-1.49	0.00	-0.42	0.000
b10100	100.00	52	0.069	0.00	-21.15	42.03	0.000
b10200	79.07	149	1.243	0.57	3.36	-12.43	0.000
b20100	100.00	87	0.360	0.00	-1.15	-9.30	0.000
b20200	91.62	146	1.226	-0.06	-6.85	11.34	0.000
Average				-0.34	-6.31	8.55	0.000
c05100	64.61	61	0.118	0.40	13.11	-15.01	0.000
c05200	58.10	61	0.232	0.00	-3.28	15.06	0.000
c10100	84.81	89	0.136	0.00	-2.25	-10.30	0.000
c10200	75.55	136	1.107	0.16	0.00	-5.18	0.000
c10400	68.71	112	1.934	0.08	8.04	5.00	0.000
c15900	39.96	201	7.290	0.23	5.47	-11.76	0.000
c20100	93.23	163	0.871	0.00	-7.36	-24.46	0.000
c201600	29.67	332	24.925	0.19	2.71	-7.65	0.000
c20200	89.79	248	1.598	-0.10	-10.48	35.64	0.000
c20400	70.19	274	4.400	-0.38	-2.19	29.29	0.000
c30900	45.92	427	18.825	0.08	4.68	-22.19	0.000
c401600	30.14	653	64.707	0.16	-1.84	-12.80	0.000
c40400	84.09	390	7.706	0.62	-3.59	-11.59	0.000
c60900	25.46	847	42.798	-0.13	-1.77	14.92	0.000
c801600	0.38	1307	234.882	0.00	4.67	-30.20	0.001
Average				0.09	0.40	-3.42	0.000

Table C.2: Results on GAP instances of A, B, and C classes using Gomory mixed-integer cuts

Name	HCG			GMI			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
a05100	100.00	1	0.006	0.00	0.00	0.00	0.000
a05200	100.00	3	0.020	0.00	0.00	2.05	0.000
a10100	100.00	18	0.049	0.00	-16.67	-18.48	0.000
a20200	100.00	27	0.380	0.00	-7.41	1.50	0.000
Average				0.00	-6.02	-3.73	0.000
b05100	53.81	58	0.077	0.06	-8.62	26.68	0.000
b05200	33.78	65	0.237	-0.02	0.00	-6.57	0.000
b10100	100.00	52	0.069	-3.16	-7.69	25.36	0.000
b10200	79.07	149	1.243	-0.19	4.03	-6.84	0.001
b20100	100.00	87	0.360	0.00	-1.15	-13.60	0.000
b20200	91.62	146	1.226	0.46	-6.16	1.02	0.001
Average				-0.48	-3.27	4.34	0.000
c05100	64.61	61	0.118	0.00	-8.20	-20.44	0.000
c05200	58.10	61	0.232	-0.04	-3.28	8.15	0.000
c10100	84.81	89	0.136	0.09	1.12	-8.76	0.000
c10200	75.55	136	1.107	-0.47	-2.21	2.05	0.001
c10400	68.71	112	1.934	0.11	14.29	17.64	0.001
c15900	39.96	201	7.290	0.18	9.45	7.09	0.004
c20100	93.23	163	0.871	-0.41	-14.11	-32.71	0.000
c201600	29.67	332	24.925	-0.13	5.12	2.18	0.012
c20200	89.79	248	1.598	0.43	-8.87	33.48	0.001
c20400	70.19	274	4.400	-0.33	-4.38	-4.64	0.002
c30900	45.92	427	18.825	-0.13	6.09	-1.69	0.009
c401600	30.14	653	64.707	-0.12	-5.05	-12.61	0.021
c40400	84.09	390	7.706	0.42	-4.10	-17.76	0.003
c60900	25.46	847	42.798	0.06	-1.18	17.90	0.016
c801600	0.38	1307	234.882	-0.01	1.61	-3.23	0.045
Average				-0.02	-0.91	-0.89	0.008

Table C.3: Results on GAP instances of A, B, and C classes using SCG cuts

Name	HCG			SCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
a05100	100.00	1	0.006	0.00	0.00	0.00	0.000
a05200	100.00	3	0.020	0.00	0.00	0.00	4.103
a10100	100.00	18	0.049	0.00	-33.33	0.00	-38.809
a20200	100.00	27	0.380	0.00	-7.41	0.00	2.526
Average				0.00	-10.19	0.00	-8.045
b05100	53.81	58	0.077	-0.12	-5.17	0.00	12.306
b05200	33.78	65	0.237	0.55	4.62	0.00	11.668
b10100	100.00	52	0.069	-3.65	-30.77	0.00	1.739
b10200	79.07	149	1.243	-0.18	-2.01	0.01	-17.301
b20100	100.00	87	0.360	0.00	-14.94	0.00	-21.143
b20200	91.62	146	1.226	0.60	-8.90	0.01	-5.277
Average				-0.47	-9.53	0.00	-3.001
c05100	64.61	61	0.118	0.00	-19.67	0.00	-34.351
c05200	58.10	61	0.232	-0.04	-3.28	0.00	8.585
c10100	84.81	89	0.136	0.11	-2.25	0.00	4.415
c10200	75.55	136	1.107	0.17	0.74	0.01	4.850
c10400	68.71	112	1.934	0.86	19.64	0.02	17.743
c15900	39.96	201	7.290	0.61	5.97	0.06	21.486
c20100	93.23	163	0.871	-0.41	-26.38	0.00	-45.366
c201600	29.67	332	24.925	0.12	-6.02	0.16	-15.665
c20200	89.79	248	1.598	0.06	-16.13	0.01	9.673
c20400	70.19	274	4.400	-0.03	-4.38	0.03	22.013
c30900	45.92	427	18.825	0.18	6.09	0.12	1.212
c401600	30.14	653	64.707	0.01	-0.92	0.31	-1.120
c40400	84.09	390	7.706	0.82	-8.46	0.04	-6.823
c60900	25.46	847	42.798	-0.02	-6.26	0.21	21.551
c801600	0.38	1307	234.882	0.00	0.84	0.62	-2.701
Average				0.16	-4.03	0.11	0.367

Table C.4: Results on GAP instances of D and E classes using maximal CG cuts

Name	HCG			MCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
d05100	44.15	84	0.175	-0.89	-1.19	-0.11	0.000
d05200	36.42	99	0.397	-0.24	-10.10	20.82	0.000
d10100	48.30	189	0.390	0.02	-3.17	18.70	0.000
d10200	21.21	207	1.724	0.47	-9.66	25.63	0.000
d10400	11.37	175	3.376	0.04	-4.57	5.81	0.000
d15900	4.52	311	13.383	0.01	-8.04	-5.08	0.000
d20100	32.80	424	3.945	-0.16	-0.71	-17.57	0.000
d201600	0.92	472	48.018	0.00	-0.21	-7.78	0.000
d20200	15.19	421	6.283	0.04	0.95	3.72	0.000
d20400	7.70	440	11.174	-0.01	-1.82	13.56	0.000
d30900	1.28	738	47.815	-0.01	-0.95	9.40	0.000
d401600	0.00	4575	300.000	0.00	-0.79	-0.14	0.003
d40400	0.50	3629	300.000	0.01	0.52	0.35	0.002
d60900	0.00	8252	300.000	0.00	1.38	-0.63	0.004
d801600	0.00	5880	300.000	0.00	-1.26	0.73	0.003
Average				-0.05	-2.64	4.49	0.001
e05100	64.75	79	0.153	0.00	0.00	-2.41	0.000
e05200	17.78	46	0.164	0.00	4.35	-9.20	0.000
e10100	62.52	124	0.227	0.07	-8.87	10.29	0.000
e10200	29.15	122	1.047	0.00	0.00	0.32	0.000
e10400	13.90	125	1.935	-0.05	-12.00	54.99	0.000
e15900	6.42	199	11.404	0.00	-1.01	-1.51	0.000
e20100	83.36	293	2.914	0.50	0.00	-5.66	0.000
e201600	6.05	308	39.606	0.04	8.77	-30.68	0.000
e20200	83.57	255	2.926	0.61	-13.33	28.78	0.000
e20400	42.63	281	7.971	-0.12	1.42	14.85	0.000
e30900	29.95	529	40.554	-0.23	-2.08	-6.86	0.000
e401600	22.22	767	112.206	-0.25	-5.22	18.97	0.000
e40400	45.27	568	300.000	0.23	44.72	-90.59	0.000
e60900	53.42	1233	176.784	0.09	-1.54	-4.59	0.001
e801600	47.26	1603	300.000	0.88	-2.00	0.17	0.001
Average				0.12	0.88	-1.54	0.000

Table C.5: Results on GAP instances of D and E classes using Gomory mixed-integer cuts

Name	HCG			GMI			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
d05100	44.15	84	0.175	-0.16	5.95	-6.75	0.000
d05200	36.42	99	0.397	-0.08	-13.13	-0.28	0.000
d10100	48.30	189	0.390	0.46	3.70	57.39	0.001
d10200	21.21	207	1.724	0.68	-6.28	38.15	0.001
d10400	11.37	175	3.376	0.09	6.29	12.47	0.002
d15900	4.52	311	13.383	0.00	-0.96	-0.94	0.006
d20100	32.80	424	3.945	0.19	-5.42	-9.18	0.001
d201600	0.92	472	48.018	0.00	-2.33	-0.58	0.015
d20200	15.19	421	6.283	0.08	-2.38	6.87	0.002
d20400	7.70	440	11.174	-0.01	3.64	21.68	0.004
d30900	1.28	738	47.815	-0.01	1.22	4.26	0.014
d401600	0.00	4575	300.000	0.00	-0.02	0.35	0.145
d40400	0.50	3629	300.000	-0.01	-10.94	-0.49	0.028
d60900	0.00	8252	300.000	0.00	1.42	0.10	0.153
d801600	0.00	5880	300.000	0.00	1.48	0.46	0.190
Average				0.08	-1.18	8.23	0.037
e05100	64.75	79	0.153	0.05	-6.33	-25.31	0.000
e05200	17.78	46	0.164	0.10	4.35	1.28	0.000
e10100	62.52	124	0.227	-0.55	-20.97	-21.78	0.000
e10200	29.15	122	1.047	0.20	4.10	38.01	0.001
e10400	13.90	125	1.935	-0.06	-6.40	8.01	0.001
e15900	6.42	199	11.404	0.05	-9.55	-2.21	0.004
e20100	83.36	293	2.914	0.72	-20.48	-31.73	0.001
e201600	6.05	308	39.606	0.01	-6.17	-43.87	0.010
e20200	83.57	255	2.926	0.29	-11.76	15.53	0.001
e20400	42.63	281	7.971	0.33	-7.83	-8.45	0.002
e30900	29.95	529	40.554	-0.19	-9.83	-19.21	0.010
e401600	22.22	767	112.206	0.03	-6.39	7.01	0.024
e40400	45.27	568	300.000	0.51	-9.33	-93.21	0.005
e60900	53.42	1233	176.784	2.29	-15.82	-30.67	0.021
e801600	47.26	1603	300.000	3.69	-10.85	2.29	0.052
Average				0.50	-8.88	-13.62	0.009

Table C.6: Results on GAP instances of D and E classes using SCG cuts

Name	HCG			SCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
d05100	44.15	84	0.175	0.14	-11.90	0.00	-25.243
d05200	36.42	99	0.397	-0.02	-15.15	0.01	-3.958
d10100	48.30	189	0.390	-0.02	4.76	0.01	29.669
d10200	21.21	207	1.724	0.68	-12.56	0.01	6.893
d10400	11.37	175	3.376	0.14	2.29	0.02	5.260
d15900	4.52	311	13.383	0.00	-0.64	0.09	4.478
d20100	32.80	424	3.945	0.59	-10.14	0.01	-23.733
d201600	0.92	472	48.018	0.00	-0.42	0.23	-3.325
d20200	15.19	421	6.283	0.30	3.09	0.03	-4.436
d20400	7.70	440	11.174	0.04	-5.45	0.05	-2.433
d30900	1.28	738	47.815	-0.01	-6.64	0.19	3.108
d401600	0.00	4575	300.000	0.00	-0.70	2.18	-0.042
d40400	0.50	3629	300.000	0.01	-7.25	0.41	0.938
d60900	0.00	8252	300.000	0.00	-0.21	2.18	0.327
d801600	0.00	5880	300.000	0.00	-1.00	2.75	-0.041
Average				0.12	-4.13	0.55	-0.836
e05100	64.75	79	0.153	1.80	-11.39	0.00	-17.417
e05200	17.78	46	0.164	-0.02	2.17	0.00	-1.888
e10100	62.52	124	0.227	1.13	-9.68	0.00	-3.476
e10200	29.15	122	1.047	-0.01	-1.64	0.01	50.200
e10400	13.90	125	1.935	-0.03	-16.80	0.02	33.972
e15900	6.42	199	11.404	0.02	-11.06	0.05	-21.105
e20100	83.36	293	2.914	1.36	-25.26	0.01	-27.359
e201600	6.05	308	39.606	0.01	8.77	0.17	-12.798
e20200	83.57	255	2.926	0.47	-12.55	0.01	10.627
e20400	42.63	281	7.971	0.22	-13.52	0.03	-3.796
e30900	29.95	529	40.554	0.10	-12.10	0.14	-12.800
e401600	22.22	767	112.206	0.22	-9.65	0.34	-14.886
e40400	45.27	568	300.000	0.71	-14.61	0.10	-94.440
e60900	53.42	1233	176.784	2.48	-18.33	0.26	-41.697
e801600	47.26	1603	300.000	3.73	-8.73	0.69	-1.245
Average				0.81	-10.29	0.12	-10.540

C.2 MIPLIB instances without consideration of generalized upper bounds

We report the detailed experiment results using MIPLIB instances without consideration of generalized upper bounds in Section 3.6.

Table C.7: Results on MIPLIB instances using maximal CG cuts and Gomory mixed-integer cuts

Name	HCG			MCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
lseu	71.33	46	0.112	0.00	0.00	1.87	0.001
mod008	83.35	78	3.116	0.17	-14.10	-20.87	0.009
p0033	87.42	46	0.014	0.00	0.00	5.84	0.000
p0282	8.36	213	2.554	-0.04	-8.45	3.27	0.013
p0291	7.15	69	17.730	0.00	4.35	5.31	0.005
p0548	60.95	343	13.302	0.98	-11.37	-8.12	0.036
p2756	84.63	800	258.429	0.62	-3.00	3.43	0.462
pipex	79.70	44	0.027	0.00	-4.55	-8.12	0.001
sentoy	24.79	68	0.879	0.00	-2.94	-2.70	0.001
Average	56.41	189.67	32.907	0.19	-4.45	-2.23	0.06

Name	HCG			GMI			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
lseu	71.33	46	0.112	0.00	0.00	2.58	0.000
mod008	83.35	78	3.116	0.19	-3.85	-5.50	0.001
p0033	87.42	46	0.014	0.00	0.00	8.03	0.000
p0282	8.36	213	2.554	0.07	-10.33	7.54	0.001
p0291	7.15	69	17.730	0.00	1.45	-1.39	0.001
p0548	60.95	343	13.302	1.01	-10.50	-11.67	0.004
p2756	84.63	800	258.429	0.68	-3.75	-8.20	0.049
pipex	79.70	44	0.027	0.00	-6.82	-5.17	0.000
sentoy	24.79	68	0.879	0.00	-8.82	-2.51	0.000
Average	56.41	189.67	32.907	0.22	-4.73	-1.81	0.01

Table C.8: Results on MIPLIB instances using SCG cuts

Name	HCG			SCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
lseu	71.33	46	0.112	0.00	-2.17	1.52	0.001
mod008	83.35	78	3.116	0.17	-15.38	-25.77	0.008
p0033	87.42	46	0.014	0.00	-4.35	1.46	0.001
p0282	8.36	213	2.554	0.07	-10.33	8.08	0.013
p0291	7.15	69	17.730	0.00	1.45	-1.33	0.005
p0548	60.95	343	13.302	1.03	-12.54	-10.86	0.036
p2756	84.63	800	258.429	0.68	-3.75	-7.76	0.446
pipex	79.70	44	0.027	0.06	-15.91	-6.27	0.001
sentoy	24.79	68	0.879	0.00	-8.82	-2.68	0.001
Average	56.41	189.67	32.907	0.22	-7.98	-4.85	0.06

C.3 MIPLIB instances with consideration of generalized upper bounds

We report the detailed experiment results using MIPLIB instances with consideration of generalized upper bounds in Section 3.6.

Table C.9: Results on MIPLIB instances using maximal CG cuts and Gomory mixed-integer cuts

Name	HCG			SCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
lseu	75.65	49	0.25	0.00	0.00	2.03	0.00
mitre	100.00	757	300.00	-0.50	0.66	0.01	0.13
mod008	83.14	75	5.38	-0.29	-37.33	-35.62	0.00
p0282	97.60	208	4.28	0.31	-1.92	22.67	0.00
p0291	98.03	67	33.63	0.00	-1.49	7.43	0.00
p0548	87.81	310	19.44	0.07	-5.48	-0.52	0.00
p2756	97.78	645	300.00	-0.23	-3.10	0.00	0.04
pipex	80.11	50	0.10	-2.22	-8.00	-20.29	0.00
sentoy	24.79	63	1.97	0.00	-4.76	0.66	0.00
sp97ar	3.47	122	16.43	0.00	0.00	1.50	0.04
Average	74.84	234.6	68.15	-0.28	-6.14	-2.21	0.02

Name	HCG			SCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
lseu	75.65	49	0.25	0.17	10.20	2.30	0.00
mitre	100.00	757	300.00	0.00	-14.40	-4.96	0.64
mod008	83.14	75	5.38	-0.29	-37.33	-37.75	0.01
p0282	97.60	208	4.28	0.39	-9.62	11.34	0.03
p0291	98.03	67	33.63	0.00	-1.49	7.54	0.01
p0548	87.81	310	19.44	0.02	-6.45	-3.20	0.09
p2756	97.78	645	300.00	0.61	-5.43	0.01	0.76
pipex	80.11	50	0.10	0.03	-24.00	-39.33	0.00
sentoy	24.79	63	1.97	0.00	-4.76	-13.83	0.00
sp97ar	3.47	122	16.43	0.00	-0.82	-15.72	0.23
Average	74.84	234.6	68.15	0.09	-9.41	-9.36	0.18

Table C.10: Results on MIPLIB instances using SCG cuts

Name	HCG			SCG			
	IGC (%)	#Cut	Time (s)	ΔIGC (%)	ΔCut (%)	ΔTime (%)	STime (s)
lseu	75.65	49	0.25	0.17	0.00	2.70	0.00
mitre	100.00	757	300.00	0.00	-6.21	-0.01	0.79
mod008	83.14	75	5.38	0.55	-37.33	-14.61	0.02
p0282	97.60	208	4.28	0.39	-7.69	8.89	0.05
p0291	98.03	67	33.63	0.00	-1.49	7.49	0.02
p0548	87.81	310	19.44	0.02	-6.45	-0.09	0.16
p2756	97.78	645	300.00	0.53	-4.34	0.00	1.29
pipex	80.11	50	0.10	-0.27	-26.00	-19.71	0.00
sentoy	24.79	63	1.97	0.00	-4.76	-8.44	0.00
sp97ar	3.47	122	16.43	0.00	-3.28	-8.98	0.32
Average	74.84	234.6	68.15	0.14	-9.76	-3.28	0.27

Appendix D

Detailed experiment results in Chapter 4

We report the detailed experiment results using MCKP instances in Section 4.6.

Table D.1: Results on MCKP instances using PL and RO

n	$r^\star$	$\Phi^{-1}(\epsilon)$	Probabilistic cover			PL			
			IGC (%)	#Cut	Time (s)	IGC (%)	#Cut	Time (s)	LTime (s)
150	25	1	10.28	20.8	0.193	22.82	32.4	3.768	3.535
		3	8.09	13.3	0.173	16.30	29.2	3.253	3.005
		5	3.56	9.8	0.175	11.60	26.6	3.517	3.213
	50	1	4.99	18.4	0.270	9.50	36.3	6.450	6.051
		3	3.44	12.9	0.275	9.46	29.1	5.559	5.123
		5	1.05	5.4	0.249	5.21	24.4	5.367	4.934
300	25	1	8.87	24.2	0.426	14.75	35.5	25.010	24.561
		3	1.85	11.0	0.346	7.06	24.4	23.804	23.288
		5	0.93	6.0	0.309	5.88	24.7	21.632	21.076
	50	1	2.81	21.0	0.754	5.96	37.3	46.916	45.970
		3	0.83	8.0	0.683	2.56	21.4	32.460	31.655
		5	0.79	6.7	0.658	2.20	19.2	31.105	30.207
Average			3.96	13.1	0.376	9.44	28.38	17.40	16.885

n	$r^\star$	$\Phi^{-1}(\epsilon)$	Probabilistic cover			RO			
			IGC (%)	#Cut	Time (s)	IGC (%)	#Cut	Time (s)	LTime (s)
150	25	1	10.28	20.8	0.193	22.40	33.3	1.453	1.198
		3	8.09	13.3	0.173	14.99	26.2	1.140	0.912
		5	3.56	9.8	0.175	10.89	24.6	1.325	1.022
	50	1	4.99	18.4	0.270	9.01	36.1	4.203	3.824
		3	3.44	12.9	0.275	8.85	27.1	3.701	3.283
		5	1.05	5.4	0.249	4.66	21.5	3.698	3.245
300	25	1	8.87	24.2	0.426	14.33	34.2	9.860	9.386
		3	1.85	11.0	0.346	6.09	22.2	9.378	8.913
		5	0.93	6.0	0.309	5.26	22.2	9.368	8.841
	50	1	2.81	21.0	0.754	5.99	36.9	29.140	28.173
		3	0.83	8.0	0.683	2.39	20.1	20.345	19.458
		5	0.79	6.7	0.658	2.16	17.4	17.087	16.238
Average			3.96	13.1	0.376	8.92	26.82	9.22	8.708

Table D.2: Results on MCKP instances using the proposed lifting heuristic

n	$r^\star$	$\Phi^{-1}(\epsilon)$	Probabilistic cover			Proposed			
			IGC (%)	#Cut	Time (s)	IGC (%)	#Cut	Time (s)	LTime (s)
150	25	1	10.28	20.8	0.193	22.82	32.4	0.237	0.007
		3	8.09	13.3	0.173	16.27	29.2	0.256	0.007
		5	3.56	9.8	0.175	11.41	26.3	0.307	0.008
	50	1	4.99	18.4	0.270	9.50	36.3	0.409	0.013
		3	3.44	12.9	0.275	9.48	28.9	0.431	0.013
		5	1.05	5.4	0.249	5.19	24.2	0.435	0.014
300	25	1	8.87	24.2	0.426	14.90	35.6	0.446	0.014
		3	1.85	11.0	0.346	7.07	24.4	0.520	0.015
		5	0.93	6.0	0.309	5.88	24.6	0.559	0.014
	50	1	2.81	21.0	0.754	5.96	37.3	0.955	0.028
		3	0.83	8.0	0.683	2.56	21.4	0.821	0.023
		5	0.79	6.7	0.658	2.19	19.1	0.927	0.024
Average			3.96	13.1	0.376	9.44	28.31	0.53	0.015

국문초록

다양한 산업에서 발생하는 의사결정문제는 이진정수최적화 문제로 모형화할 수 있으며, 지난 수십년간에 걸친 그 해법의 발전은 현실의 최적화 이슈들을 해결하는데 크게 공헌해왔다. 이진정수최적화 문제의 해법을 개선시킨 주요 요인 중 하나는 이진배낭문제에 대한 절단평면(cutting plane)들의 활용이다. 이러한 절단평면들은 주어진 문제의 완화된 해집합을 더욱 정교하게 정제하여, 완화(relaxation) 기반 최적화 알고리듬의 성능을 향상시킨다. 그러나, 급격한 산업의 발전은 더욱 복잡한 이진정수최적화 문제로 모형화되는 도전적인 운영 이슈들을 발생시키고 있으며, 그 문제들에 대응하기 위한 개선된 해법들이 요구되고 있다. 이에 대한 한 가지 해결책으로써 이진정수최적화 문제에 대한 더욱 효과적인 절단평면과 그 생성 기법들을 도출하기 위해 이진배낭문제의 변형에 대한 연구들이 활발히 진행되고 있다.

본 논문에서는 그러한 이진배낭문제의 두 가지 변형, 일반화된 상한제약이 있는 이진배낭문제(binary knapsack problem with generalized upper bounds, GKP)와 확률제약이 있는 이진배낭문제(chance-constrained binary knapsack problem, CKP)에 대한 효율적인 절단평면 생성 기법을 개발하여 이진정수최적화 문제에 대한 해결 능력을 제고한다. 먼저, 이진배낭문제보다 더욱 강력한 절단평면을 도출할 수 있는 GKP에 대해서, 일반적인 선형정수최적화 문제에서 정의될 수 있는 크바탈-고모리(Chvátal-Gomory, CG) 절단평면들을 다룬다. 일반적인 선형정수최적화 문제에 대한 CG 절단평면들을 생성하는 분리(separation) 문제는 강성 NP-hard임이 증명되었으나, GKP에 대한 분리 문제는 유사다항시간(pseudo-polynomial time) 내에 해결될 수 있음을 밝힌다. 또한, GKP에 대한 분리 문제의 분해 성질에 기반하여, 효율적으로 CG 절단평면들을 생성하는 휴리스틱 분리 알고리듬도 함께 제안한다. 계산실험결과를 통해,

제안된 분리 알고리듬으로 생성된 CG 절단평면들은 기존에 알려진 GKP의 절단평면들에 비해서 비슷한 시간 내에 선형이진정수최적화문제의 선형 완화를 현저히 개선함을 확인한다.

다음으로, 선형이진정수최적화 문제의 CG 절단평면들을 강화하는 새로운 기법을 제시하여 CG 절단평면들의 모형 강화 효과를 개선한다. 우선 주어진 선형이진정수최적화문제의 CG 절단평면과 이진배낭문제에 대한 커버 부등식(cover inequality) 사이의 관계성을 밝힌다. 이 관계성에 입각하여, 커버 부등식에 대한 리프팅 함수(lifting function)를 활용해, 주어진 선형이진정수최적화문제의 CG 절단평면보다 강한 절단평면을 생성해내는 강화 기법을 제안한다. 제안된 강화 기법은 일반화된 상한제약이 있는 선형이진정수최적화 문제의 CG 절단평면에 대해서 확장된다. 이론적 비교를 통해, 제안된 강화 기법은 기존에 제시된 기법들보다 더욱 강한 절단평면을 생성함을 보인다. 더불어, 계산실험을 통해서, 제안된 강화 기법은 더 적은 절단평면으로 정의된 더욱 강화된 모형을 도출할 수 있으며, 그 모형을 얻을 때까지 소요되는 계산 시간도 감소시킨다는 것을 보인다.

마지막으로, 불확실성이 존재하는 최적화 문제에 대한 확률제약모형(chance-constrained program)에서 나타나는 CKP를 다룬다. 본 논문에서는 아이템들의 무게가 서로 독립인 정규분포를 따른다고 가정하는데, 이러한 CKP는 비선형 이진정수최적화 문제로 모형화된다. 본 논문에서는 이 CKP에 대해 잘 알려진 절단평면인 확률적 커버 부등식(probabilistic cover inequality)의 효율적인 리프팅 휴리스틱을 제시한다. 먼저 CKP에 대한 비볼록 연속 완화(non-convex continuous relaxation)를 제시하고, 기존에 제시된 다른 연속 완화들보다 더 강한 상한을 제공함을 밝힌다. CKP에 대한 비볼록 연속 완화는 일반적으로 풀기 힘든 비볼록 최적화 문제로 표현되지만, 그 완화에 대한 다항 시간 알고리듬을 제시한다. 제안하는 리프팅 휴리스틱은 CKP에 대한 비볼록 연속 완화와 그 다항 시간 알고리듬을 활용한다. 계산실험결과는 제안된 리프팅 휴리스틱이

기존에 제시된 방법들보다 압도적으로 빠른 시간내에 리프팅을 수행을 하는 반면, 그 결과로 생성된 리프팅된 확률적 커버 부등식의 효과성이 여전이 유지됨을 보여준다.

주요어: 이진정수최적화, 배낭문제, 일반화된 상한제약, 확률제약, 범용 절단평면, 크바탈-고모리 절단평면, 커버 부등식, 분리 알고리듬, 리프팅

학번: 2017-23584