Enhancing the separation of rank-1 Chvátal-Gomory cuts from knapsack sets

Giacomo Maggiorano, Stefano Gualandi, Pasquale Avella, Michele Mele

Abstract

We present an exact method for separating Chvátal-Gomory cuts from binary knapsack sets, consisting of two steps: *i*) enumerating a finite set of possible optimal multipliers for the knapsack constraint; *ii*) for each candidate, adjusting optimally the remaining multipliers. We prove that *ii*) can be formulated as a binary knapsack problem, leading to a pseudopolynomial-time exact separation algorithm and two efficient greedy heuristics. Computational experiments on Generalized Assignment Problem instances confirm the effectiveness of the approach in terms of cut strength and computational efficiency.

1 Introduction

Mixed-integer programming (MIP) solvers have advanced significantly over the past few decades, becoming able to handle complex and large-scale problems with increased efficiency. A pivotal factor in the progress of MIP solvers has been the development of efficient techniques to tighten MIP formulations through the automatic generation of cutting planes [6].

Modern MIP solvers derive cutting planes from special substructures (i.e. subsets of variables and/or constraints) of mixed-integer programming problems [13, 21, 4]. The most commonly exploited substructures are knapsack sets [14], mixed knapsack sets [18], single-node flow sets [11, 17], and set packing (conflict graphs) [1]. Knapsack sets have been used as a base to derive Lifted Cover inequalities, whose relevance is evidenced by the great attention received in the literature [12]. The effectiveness demonstrated by Lifted Cover inequalities raises the question of whether more aggressive separation algorithms on knapsack sets can further strengthen MIP formulations. For this purpose, exact separation procedures have been tested in [2]. Exact separation enhancements are beneficial for small yet challenging problems; however, they remain too time-consuming for larger ones.

In an effort to find a good compromise between quality of the lower bounds and computational cost, previous works [15, 19] proposed a separation framework to generate cutting planes belonging to the first Chvátal-Gomory (CG) closure of knapsack sets. They introduce additional variables to reformulate the original knapsack set as a fixed-charge knapsack set, then they perform two main steps:

i) enumerate in a discrete set the multiplier values associated with the knapsack constraint, then ii) for each main multiplier adjust the multipliers associated with the bounds ≤ 1. They affirm that this leads to a $O(n^3b^3)$ separation procedure, without providing computational results [19]. Building on similar considerations, [16] proposes a heuristic to identify violated CG cuts.

Our contribution. In this paper (Section 2), inspired by the separation framework of [19], we develop an approach for separating rank-1 Chvátal-Gomory cuts directly from binary knapsack inequalities. First, we strengthen step i) showing that intermediate fixed-charge knapsack reformulations are unnecessary. This leads to a reduced and explicitly characterized discrete set of candidate multipliers (Theorem 2). More importantly, we propose a novel way of dealing with step ii) by proving that the multiplier-adjustment can be formulated as a binary knapsack problem (Theorem 3), thus providing a new structural perspective on the separation process. Building on these theoretical findings, we design a new pseudo-polynomial exact separation algorithm with complexity $O(n^3a_{MAX}^2)$, improving upon best previously known bounds [19], as well as two efficient greedy heuristics. Computational experiments on Generalized Assignment Problem instances (Section 3) demonstrate the practical effectiveness of the proposed methods.

2 Chvátal-Gomory Cuts from the Knapsack Substructure

Let $N = \{1, \dots, n\}$, $b \in \mathbb{Z}$, and $a_j \in \mathbb{Z}\ \forall j \in N$. We consider the binary knapsack set $BKP = \{x \in \{0,1\}^n : \sum_{j\in N} a_j x_j \leq b\}$, with $b > 0$, and $a_j > 0$, $a_j \leq b$, $j \in N$. We denote by KP_I the convex hull of BKP, and by KP the set obtained relaxing the integrality constraint in BKP, that is $KP \coloneqq \{x \in \mathbb{R}^n : \sum_{j\in N} a_j x_j \leq b, 0 \leq x_j \leq 1, j = 1, \dots, n\}$. Any inequality of the form:

$$\sum_{j\in N} \lfloor u_0 a_j + u_j \rfloor x_j \leq \lfloor b u_0 + \sum_{j\in N} u_j \rfloor \tag{1}$$

for some $u_0, u_1, \dots, u_n \in \mathbb{R}_+$, is a rank-1 Chvátal-Gomory (CG) inequality [10, 5]. The set of points in KP satisfying all rank-1 CG inequalities is called the first CG closure and is denoted by KP^1, and we have $KP_I \subseteq KP^1$[6]. For a given $\bar{x} = (\bar{x}_1, \dots, \bar{x}_n) \in KP$, the problem of computing a violated inequality of the form (1), or proving that this cannot be found (and hence $\bar{x} \in KP^1$) is referred to as *CG separation problem* (CGSEP) and is known to be NP-hard [7]. CGSEP can be rephrased as a Mixed Integer Linear Program [8]:

$$\begin{aligned} z^* \coloneqq \max \quad & \sum_{j\in N} \lfloor u_0 a_j + u_j \rfloor \bar{x}_j - \lfloor b u_0 + \sum_{j\in N} u_j \rfloor \\ \text{subject to} \quad & 0 \leq u_0 < 1 \\ & 0 \leq u_j < 1 \qquad\qquad \forall j \in N. \end{aligned} \tag{2}$$

$\bar{x} \in KP^1$ if and only if the optimal value $z^* \leq 0$. Otherwise, the optimal solution $(u_0^*, u_1^*, \ldots, u_n^*)$ defines a maximally violated CG inequality for $\bar{x}$. The additional upper bound on the values of u_j is a necessary condition for obtaining non-dominated inequalities [20]. The value of z^* is unaffected by fixing $u_j = 0$ when $\bar{x}_j = 0$; by complementation, a similar reasoning holds for $\bar{x}_j = 1$. Hence, in (2) N can be replaced with $J = \{j \in N : 0 < \bar{x}_j < 1\}$ [8].

Given $\bar{x} \in KP$, for any $u = (u_0, u_1, \ldots u_n) \in \mathbb{R}^{n+1}$ that identifies a CG cut, we denote the value of the objective function of (2) as:

$$z(u_0, u_1, \ldots, u_n) \coloneqq \sum_{j \in J} \lfloor u_0 a_j + u_j \rfloor \bar{x}_j - \lfloor b u_0 + \sum_{j \in J} u_j \rfloor. \tag{3}$$

Since $0 \leq u_j < 1$ for all $j \in J$, we have that $\lfloor u_0 a_j \rfloor \leq \lfloor u_0 a_j + u_j \rfloor \leq \lfloor u_0 a_j \rfloor + 1$. We denote as $I(u_0, u_1, \ldots, u_n)$ the set of indices such that adding u_j raises the value of $\lfloor u_0 a_j \rfloor$ to the following integer, i.e.

$$I(u_0, u_1, \ldots, u_n) \coloneqq \{j \in J : \lfloor u_0 a_j + u_j \rfloor > \lfloor u_0 a_j \rfloor\} = \{j \in J : \lfloor u_0 a_j + u_j \rfloor = \lfloor u_0 a_j \rfloor + 1\}.$$

We observe that the first term in (3), namely $\sum_{j \in J} \lfloor u_0 a_j + u_j \rfloor \bar{x}_j$, depends only on u_0 and the set of indices $I(u_0, u_1, \ldots, u_n)$. On the contrary, the second term $-\lfloor b u_0 + \sum_{j \in J} u_j \rfloor$ has a direct dependency on all $u_0, u_1, \ldots, u_n$. Therefore, when we consider $u = (u_0, u_1, \ldots, u_n)$, we can always improve the corresponding value of the objective function z by substituting the last n components of u with the minimal values $\tilde{u}_1, \ldots, \tilde{u}_n$ such that $I(u_0, \tilde{u}_1, \ldots, \tilde{u}_n) = I(u_0, u_1, \ldots, u_n)$. In this way, the first term will remain constant while the second one will be as small as possible. This can be formalized with the following proposition [19, 16].

Proposition 1. *Let $(u_0, u_1, \ldots, u_n)$ be a feasible solution for (2), with $I = I(u_0, u_1, \ldots, u_n)$. Then $z(u_0, u_1, \ldots, u_n) \leq z(u_0, \tilde{u}_1, \ldots, \tilde{u}_n)$, where:*

$$\tilde{u}_j = \begin{cases} 1 - (u_0 a_j - \lfloor u_0 a_j \rfloor) & \text{if } j \in I, \\ 0 & \text{if } j \in J \setminus I. \end{cases} \tag{4}$$

Proof. We observe that $\lfloor u_0 a_j + u_j \rfloor = \lfloor u_0 a_j + \tilde{u}_j \rfloor \ \forall j \in J$. In fact if $j \in I$:

$$\begin{aligned} \lfloor u_0 a_j + u_j \rfloor &= \lfloor u_0 a_j \rfloor + 1 = \lfloor \lfloor u_0 a_j \rfloor + 1 \rfloor = \lfloor u_0 a_j - u_0 a_j + \lfloor u_0 a_j \rfloor + 1 \rfloor \\ &= \lfloor u_0 a_j + \tilde{u}_j \rfloor, \end{aligned}$$

while if $j \in J \setminus I$:

$$\lfloor u_0 a_j + u_j \rfloor = \lfloor u_0 a_j \rfloor = \lfloor u_0 a_j + 0 \rfloor = \lfloor u_0 a_j + \tilde{u}_j \rfloor.$$

This fact implies that:

$$\sum_{j \in J} \lfloor u_0 a_j + u_j \rfloor \bar{x}_j = \sum_{j \in J} \lfloor u_0 a_j + \tilde{u}_j \rfloor \bar{x}_j. \tag{5}$$

We recall that $0 \leq u_j < 1 \; \forall j \in J$: in particular $u_j \geq 0 = \tilde{u}_j \; \forall j \in J \backslash I$. Moreover, since $u_0 a_j + u_j \geq \lfloor u_0 a_j + u_j \rfloor = \lfloor u_0 a_j \rfloor + 1$ for all $j \in I$, we have $u_j \geq 1 - (u_0 a_j - \lfloor u_0 a_j \rfloor) = \tilde{u}_j$. Then $u_j \geq \tilde{u}_j \; \forall j \in J$, which in turn implies:

$$\lfloor b u_0 + \sum_{j \in J} u_j \rfloor \geq \lfloor b u_0 + \sum_{j \in J} \tilde{u}_j \rfloor. \tag{6}$$

Using (5) and (6) we can now conclude:

$$\begin{aligned} z(u_0, u_1, \ldots, u_n) &= \sum_{j \in J} \lfloor u_0 a_j + u_j \rfloor \bar{x}_j - \lfloor b u_0 + \sum_{j \in J} u_j \rfloor \\ &\leq \sum_{j \in J} \lfloor u_0 a_j + \tilde{u}_j \rfloor \bar{x}_j - \lfloor b u_0 + \sum_{j \in J} \tilde{u}_j \rfloor = z(u_0, \tilde{u}_1, \ldots, \tilde{u}_n). \end{aligned}$$

□

Thanks to Proposition 1, we can restrict to optimal solutions for (2) of the form (4). Moreover, for solutions of this form, the objective function can now be expressed as a function of the first component u_0 and the set of indices I, i.e. $z(u_0, I) \coloneqq z(u_0, \tilde{u}_1, \ldots, \tilde{u}_n)$ where $I = I(u_0, \tilde{u}_1, \ldots, \tilde{u}_n) = \{j \in J : \tilde{u}_j > 0\}$.

Problem (2) reduces to determining an optimal multiplier $u_0^* \in \mathbb{R}$ and the corresponding index set $I^* \subseteq J$ that maximize the function $z(u_0, I)$. Previous work [19] showed that an optimal value of u_0 can be identified by restricting attention to a finite discrete set, by considering the fixed-charged extension of the original knapsack problem (see Theorems 1–3 in [19]). In the following Theorem 2, we eliminate the need for this intermediate reformulation and prove a direct characterization of the relevant discrete set of candidate multipliers. Notably, the cardinality of this set does not depend explicitly on the knapsack capacity b, in contrast to the construction in [19].

Theorem 2. *There exists an optimal solution* $(u_0^*, u_1^*, \ldots, u_n^*)$ *to* (2) *such that* $u_0^* \in C = \{p/a_i : \; i \in J, \; p \in \mathbb{Z}_+, \; 0 \leq p < a_i\}$.

Proof. Let us denote $C \cup \{1\} = \{c_1, \ldots, c_{|C|+1} = 1\}$ with $c_k < c_{k+1}$ for all $k = 1, \ldots, |C| + 1$. Let us consider an optimal solution $(u_0^*, u_1^*, \ldots, u_n^*)$ to (2). Since $0 \leq u_0^* < 1$, then there exists $k \in \{1, \ldots, |C|\}$ such that $c_k \leq u_0^* < c_{k+1}$. If $u_0^* = c_k$ we have the thesis.

Let us consider the case $c_k < u_0^* < c_{k+1}$. Thanks to Proposition 1, we can assume that $(u_0^*, u_1^*, \ldots, u_n^*)$ is of the form (4), otherwise we can consider the corresponding optimal solution with the same u_0^*. We define $I^* = \{j \in J : \lfloor u_0^* a_j + u_j^* \rfloor = \lfloor u_0^* a_j \rfloor + 1\}$. We consider two cases:

(i) $b - \sum_{j \in I^*} a_j \geq 0$;

(ii) $b - \sum_{j \in I^*} a_j < 0$.

In the following, to allow easier manipulations, we consider separately the positive and the negative term of the objective function:

$$z(u_0, I) = f(u_0, I) - g(u_0, I)$$

where

$$\begin{aligned} f(u_0, I) &= \sum_{j \in J} \lfloor u_0 a_j \rfloor \bar{x}_j + \sum_{j \in I} \bar{x}_j \\ g(u_0, I) &= \lfloor b u_0 + \sum_{j \in I} (1 - (u_0 a_j - \lfloor u_0 a_j \rfloor)) \rfloor \\ &= \lfloor b u_0 + \sum_{j \in I} \big(1 + \lfloor u_0 a_j \rfloor\big) - \sum_{j \in I} u_0 a_j \rfloor \\ &= |I| + \sum_{j \in I} \lfloor u_0 a_j \rfloor + \lfloor u_0 \Big(b - \sum_{j \in I} a_j \Big) \rfloor. \end{aligned}$$

(i) Let us define:

$$A = I^* \cap \{ j \in J : c_k a_j \in \mathbb{Z}_+ \}.$$

We show that a solution of the form (4) with $u_0 = c_k$ and $I = I^* \setminus A$ is also optimal, i.e. $z(c_k, I^* \setminus A) \geq z(u_0^*, I^*)$. We observe that, since $c_k < u_0^* < c_{k+1}$, we have $\lfloor u_0^* a_j \rfloor = \lfloor c_k a_j \rfloor$ for all $j \in J$.

$$\begin{aligned} f(c_k, I^* \setminus A) =& \sum_{j \in J} \lfloor c_k a_j \rfloor \bar{x}_j + \sum_{j \in I^* \setminus A} \bar{x}_j \\ =& \sum_{j \in J} \lfloor u_0^* a_j \rfloor \bar{x}_j + \sum_{j \in I^*} \bar{x}_j - \sum_{j \in A} \bar{x}_j \\ =& f(u_0^*, I^*) - \sum_{j \in A} \bar{x}_j. \\ g(c_k, I^* \setminus A) =& |I^* \setminus A| + \sum_{j \in I^* \setminus A} \lfloor c_k a_j \rfloor + \lfloor c_k \Big(b - \sum_{j \in I^* \setminus A} a_j \Big) \rfloor \\ =& |I^*| - |A| + \sum_{j \in I^*} \lfloor c_k a_j \rfloor - \sum_{j \in A} \lfloor c_k a_j \rfloor + \lfloor c_k \Big(b - \sum_{j \in I^*} a_j \Big) + \sum_{j \in A} c_k a_j \rfloor \\ =& |I^*| - |A| + \sum_{j \in I^*} \lfloor u_0^* a_j \rfloor - \sum_{j \in A} c_k a_j + \lfloor c_k \Big(b - \sum_{j \in I^*} a_j \Big) \rfloor + \sum_{j \in A} c_k a_j \\ =& |I^*| - |A| + \sum_{j \in I^*} \lfloor u_0^* a_j \rfloor + \lfloor c_k \Big(b - \sum_{j \in I^*} a_j \Big) \rfloor \\ \leq& |I^*| + \sum_{j \in I^*} \lfloor u_0^* a_j \rfloor + \lfloor u_0^* \Big(b - \sum_{j \in I^*} a_j \Big) \rfloor - |A| \end{aligned}$$

$$= g(u_0^*, I^*) - |A|\,.$$

To obtain the inequality, we used hypothesis (i) and the fact that $c_k < u_0^*$. Then recalling the previous results and the fact that $x_j \leq 1\ \forall j \in J$:

$$z(c_k, I^* \setminus A) \geq f(u_0^*, I^*) - \sum_{j\in A} \bar{x}_j - g(u_0^*, I^*) + |A| = z(u_0^*, I^*) + \Big(|A| - \sum_{j\in A} \bar{x}_j\Big) \geq z(u_0^*, I^*).$$

(ii) Let us define:

$$A = I^* \cap \{j \in J : c_{k+1}a_j \in \mathbb{Z}_+\},$$
$$B = (J \setminus I^*) \cap \{j \in J : c_{k+1}a_j \in \mathbb{Z}_+\}.$$

We show that a solution of the form (4) with $u_0 = c_{k+1}$ and $I = I^* \setminus A$ is also optimal, i.e. $z(c_{k+1}, I^* \setminus A) \geq z(u_0^*, I^*)$. We observe that, since $c_k < u_0^* < c_{k+1}$, we have $\lfloor c_{k+1}a_j \rfloor = \lfloor u_0^* a_j \rfloor + 1$ for all j such that $c_{k+1}a_j \in \mathbb{Z}$ (i.e. $j \in A \cup B$), and $\lfloor c_{k+1}a_j \rfloor = \lfloor u_0^* a_j \rfloor$ otherwise.

$$\begin{aligned}
f(c_{k+1}, I^* \setminus A) &= \sum_{j\in J} \lfloor c_{k+1}a_j \rfloor \bar{x}_j + \sum_{j\in I^*\setminus A} \bar{x}_j \\
&= \sum_{j\in J} \lfloor u_0^* a_j \rfloor \bar{x}_j + \sum_{j\in A\cup B} \bar{x}_j + \sum_{j\in I^*} \bar{x}_j - \sum_{j\in A} \bar{x}_j \\
&= f(u_0^*, I^*) + \sum_{j\in B} \bar{x}_j.
\end{aligned}$$

$$\begin{aligned}
g(c_{k+1}, I^* \setminus A) &= |I^* \setminus A| + \sum_{j\in I^*\setminus A} \lfloor c_{k+1}a_j \rfloor + \lfloor c_{k+1}\Big(b - \sum_{j\in I^*\setminus A} a_j\Big) \rfloor \\
&= |I^*| - |A| + \sum_{j\in I^*} \lfloor c_{k+1}a_j \rfloor - \sum_{j\in A} \lfloor c_{k+1}a_j \rfloor + \lfloor c_{k+1}\Big(b - \sum_{j\in I^*} a_j\Big) + \sum_{j\in A} c_{k+1}a_j \rfloor \\
&= |I^*| - |A| + \sum_{j\in I^*} \lfloor u_0^* a_j \rfloor + |A| - \sum_{j\in A} c_{k+1}a_j + \lfloor c_{k+1}\Big(b - \sum_{j\in I^*} a_j\Big) \rfloor + \sum_{j\in A} c_{k+1}a_j \\
&= |I^*| + \sum_{j\in I^*} \lfloor u_0^* a_j \rfloor + \lfloor c_{k+1}\Big(b - \sum_{j\in I^*} a_j\Big) \rfloor \\
&\leq |I^*| + \sum_{j\in I^*} \lfloor u_0^* a_j \rfloor + \lfloor u_0^*\Big(b - \sum_{j\in I^*} a_j\Big) \rfloor = g(u_0^*, I^*).
\end{aligned}$$

To obtain the inequality, we used hypothesis (ii) and the fact that $u_0^* < c_{k+1}$. We can now observe the following:

$$z(c_{k+1}, I^* \setminus A) \geq f(u_0^*, I^*) + \sum_{j\in B} \bar{x}_j - g(u_0^*, I^*) = z(u_0^*, I^*) + \sum_{j\in B} \bar{x}_j \geq z(u_0^*, I^*).$$

We are left to rule out the case $c_{k+1} = 1$, otherwise the new solution would be unfeasible for (2). If this was indeed the case, we would have, using the fact that $a_j \in \mathbb{Z}_+$ and $0 \leq u_j < 1$ for all $j \in J$:

$$\begin{aligned} z(1, u_1, \ldots, u_n) &= \sum_{j \in J} \lfloor a_j + u_j \rfloor \bar{x}_j - \lfloor b + \sum_{j \in J} u_j \rfloor = \sum_{j \in J} a_j \bar{x}_j - b - \lfloor \sum_{j \in J} u_j \rfloor \\ &\leq \sum_{j \in J} a_j \bar{x}_j - b \leq 0 \end{aligned}$$

but if we chose $u_0 = 0$ the objective function becomes:

$$z(0, u_1, \ldots, u_n) = \sum_{j \in J} \lfloor 0 + u_j \rfloor \bar{x}_j - \lfloor 0 + \sum_{j \in J} u_j \rfloor \leq 0 = z(0, 0, \ldots, 0),$$

Hence $(0, 0, \ldots, 0)$ can be selected as optimal (feasible) solution. □

The maximization problem (2) can be then decomposed in $|C|$ subproblems, corresponding to fixing $u_0 = p/a_i$ and determining the set of indices $I^* \subseteq J$ that maximizes $z(p/a_i, I)$. The case $u_0 = 0$ can be excluded as the corresponding optimal objective value $z(0, I^*) = 0$ would imply that no CG inequality is violated by $\bar{x}$. The following Theorem 3 introduces a novel way to compute the remaining optimal multipliers, leading to a new approach to compute maximally violated CG cuts for KP_I, forming the basis of the exact separation algorithm which is the major contribution of this paper.

Theorem 3. *There exists an optimal solution $(u_0^*, u_1^*, \ldots, u_n^*)$ to (2) such that $u_0^* = p/a_i$ for some $i \in J$, $0 \leq p < a_i$, and $u_j^* = w_j^* \cdot \alpha_j / a_i \ \forall j \in J$ where:*

$$\begin{aligned} \beta = \beta(a_i, p) &:= pb \mod a_i, \\ \alpha_j = \alpha_j(a_i, p) &:= a_i - (pa_j \mod a_i) \qquad \forall j \in J \\ w^* = \arg \max_{w^k \ s.t. \ 0 \leq k \leq \lfloor (\beta + \sum_{j \in J} \alpha_j)/a_i \rfloor} & \sum_{j \in J} \bar{x}_j w_j^k - k \end{aligned}$$

and $w^k = (w_j^k)_{j \in J}$ is the optimal solution to

$$\begin{aligned} \max \quad & \sum_{j \in J} \bar{x}_j w_j \\ \textit{subject to} \quad & \sum_{j \in J} \alpha_j w_j \leq (k+1) a_i - 1 - \beta \\ & w_j \in \{0, 1\} \qquad \forall j \in J. \end{aligned} \tag{7}$$

Proof. For a given u_0 and a set of indices $I \subseteq J$, the value of the objective function $z(u_0, I)$ is given by

$$\begin{aligned} z(u_0, I) &= \sum_{j\in J} \lfloor u_0 a_j \rfloor \bar{x}_j + \sum_{j\in I} \bar{x}_j - \lfloor b u_0 + \sum_{j\in I} \big(1 - (u_0 a_j - \lfloor u_0 a_j \rfloor)\big) \rfloor \\ &= \sum_{j\in J} \lfloor u_0 a_j \rfloor \bar{x}_j + \sum_{j\in I} \bar{x}_j - \lfloor u_0 b - \lfloor u_0 b \rfloor + \lfloor u_0 b \rfloor + \sum_{j\in I} \big(1 - (u_0 a_j - \lfloor u_0 a_j \rfloor)\big) \rfloor \\ &= \left(\sum_{j\in J} \lfloor u_0 a_j \rfloor \bar{x}_j - \lfloor u_0 b \rfloor \right) + \sum_{j\in I} \bar{x}_j - \lfloor u_0 b - \lfloor u_0 b \rfloor + \sum_{j\in I} \big(1 - (u_0 a_j - \lfloor u_0 a_j \rfloor)\big) \rfloor. \end{aligned}$$

By Theorem 2, we can furthermore assume that $u_0 = p/a_i$ for some $i \in J$ and $p \in \mathbb{Z}_+$, $1 \le p \le a_i - 1$. Therefore:

$$\begin{aligned} u_0 b - \lfloor u_0 b \rfloor &= \frac{1}{a_i}(pb \mod a_i) = \frac{\beta}{a_i}, \\ 1 - (u_0 a_j - \lfloor u_0 a_j \rfloor) &= \frac{1}{a_i}(a_i - (p a_j \mod a_i)) = \frac{\alpha_j}{a_i}. \end{aligned}$$

and

$$z\left(\frac{p}{a_i}, I\right) = \left(\sum_{j\in J} \lfloor \frac{p}{a_i} a_j \rfloor \bar{x}_j - \lfloor \frac{p}{a_i} b \rfloor \right) + \sum_{j\in I} \bar{x}_j - \lfloor \frac{1}{a_i} \left(\beta + \sum_{j\in I} \alpha_j \right) \rfloor. \tag{8}$$

The first term does not depend on I, while the last one is such that $0 \le \lfloor \frac{1}{a_i} \left(\beta + \sum_{j\in I} \alpha_j \right) \rfloor \le |J|$ since $0 \le \beta < a_i$ and $0 < \alpha_j \le a_i \ \forall j \in J$. If we suppose to fix this last term to some $k \in \{0, \ldots, |J|\}$, the task of maximizing (8) is equivalent to finding $I \subseteq J$ that maximizes $\sum_{j\in I} \bar{x}_j$ subject to

$$\frac{1}{a_i}\Big(\beta + \sum_{j\in I} \alpha_j\Big) < k + 1$$

or equivalently

$$\sum_{j\in I} \alpha_j \le (k+1) a_i - 1 - \beta.$$

This task can be decomposed in $O(|J|) = O(n)$ knapsack problems of the form:

$$\begin{aligned} \max \quad & \sum_{j\in J} \bar{x}_j w_j \\ \text{subject to} \quad & \sum_{j\in J} \alpha_j w_j \le (k+1) a_i - 1 - \beta \\ & w_j \in \{0, 1\} \qquad \forall j \in J, \end{aligned} \tag{9}$$

where w_j is a binary variable that is equal to 1 if and only if $j \in I$. □

We have $|C| = O(na_{MAX})$ subproblems, where $a_{MAX} \coloneqq \max_{i \in J} a_i$. Solving each problem (7) individually requires $O(|J|\,((k+1)a_i - 1 - \beta))$ operations using dynamic programming [20]. Since $k \leq |J| \leq n$ and $a_i \leq a_{MAX}$, this amounts to $O(n^2 a_{MAX})$ operations. This leads to a new algorithm for computing a maximally violated CG cut in $O(na_{MAX} \cdot n \cdot n^2 a_{MAX}) = O(n^4 a_{MAX}^2)$ operations.

However, for given a_i and p, the sequence of the knapsack subproblems (7) only differs for the value of the RHS (corresponding to the "capacity" of the knapsack). Therefore it is possible to compute the dynamic programming table only once for the highest capacity $\lfloor(\beta + \sum_{j \in I} \alpha_j)/a_i\rfloor$ and from that retrieve the optimal solutions for all the problems (7) with lower capacities [20]. This consideration allows us to reduce the overall complexity of computing a maximally violated CG cut to $O(n^3 a_{MAX}^2)$ operations, thus strictly improving the complexity bound for similar approaches [19]. This approach is illustrated by Algorithm 1 and provides a technique to optimize a function over the first CG closure KP^1.

Algorithm 1 Explicit resolution of the Knapsack subProblems (EKP)

Require: $a_1, \ldots, a_n, b \in \mathbb{Z}_+$, $\bar{x} \in KP$
Ensure: $u_0^*, u_1^*, \ldots, u_n^* \in [0, 1)$, $z^* \in \mathbb{R}$
$u_0^* = 0, u_1^* = 0, \ldots, u_n^* = 0, z^* = 0$
for $i \in J = \{j : 0 < \bar{x}_j < 1\}$ **do**
  for $p \in \{1, \ldots, a_i - 1\}$ **do**
    $\beta = pb \mod a_i$
    $\alpha_j = a_i - (pa_j \mod a_i)$ for all $j \in J$
    Compute the DP table for (7) with RHS $\lfloor(\beta + \sum_{j \in N_2} \alpha_j)/a_i\rfloor$
    for $k \in \{0, \ldots, \lfloor(\beta + \sum_{j \in N_2} \alpha_j)/a_i\rfloor\}$ **do**
      Retrieve the optimal w to (7) for α_j, β, and k using DP table
      $z = \sum_{j \in J} \lfloor pa_j/a_i \rfloor \bar{x}_j - \lfloor pb/a_i \rfloor + \sum_{j \in J} \bar{x}_j w_j - k$
      if $z > z^*$ **then**
        $z^* = z$
        $u_0^* = p/a_i$
        $u_j^* = w_j \cdot \alpha_j / a_i \; \forall j \in J$

Solving this family of linear programs explicitly can be computationally demanding. Algorithm 2 exploits the standard greedy heuristic for the knapsack problem [6] to speed up this step. The CG cuts obtained in this way are not ensured to be maximally violated, but exhibit a better trade-off between computation time and performance in branch-and-cut setting (Sec. 3).

In a similar spirit, another approach can be devised by using a different heuristic for finding, for all given $u_0 \in C$, the set of indices $I^* \subseteq J$ that maximizes $z(u_0, I^*)$. Let us consider the following expression for the objective:

$$z(u_0, I) = \sum_{j \in J} \lfloor u_0 a_j \rfloor \bar{x}_j + \sum_{j \in I} \bar{x}_j - |I| - \Big\lfloor u_0 b - \sum_{j \in I} (u_0 a_j - \lfloor u_0 a_j \rfloor) \Big\rfloor.$$

First, we observe that $\sum_{j \in J} \lfloor u_0 a_j \rfloor \bar{x}_j$ depends only on u_0, and therefore can be seen as a constant value F once u_0 is fixed. For a given cardinality $|I|$, $z(u_0, I)$ increases when either $\sum_{j \in I} \bar{x}_j$ increases or $\lfloor u_0 b - \sum_{j \in I} (u_0 a_j - \lfloor u_0 a_j \rfloor) \rfloor$ decreases,

Algorithm 2 Heuristic 1 (H1)

Require: $a_1, \ldots, a_n, b \in \mathbb{Z}_+$, $\bar{x} \in KP$
Ensure: $u_0^*, u_1^*, \ldots, u_n^* \in [0, 1)$, $z^* \in \mathbb{R}$
$u_0^* = 0, u_1^* = 0, \ldots, u_n^* = 0, z^* = 0$
for $i \in J = \{j : 0 < \bar{x}_j < 1\}$ **do**
  for $p \in \{1, \ldots, a_i - 1\}$ **do**
    $\beta = pb \mod a_i$
    $\alpha_j = a_i - (pa_j \mod a_i)$ for all $j \in J$
    Sort $J = \{j_1, \ldots, j_{|J|}\}$ in decreasing order of $\bar{x}_j/\alpha_j$
    for $k \in \{0, \ldots, |J|\}$ **do**
      $I = \{j_l : l = 1, \ldots, k\}$
      $z = \sum_{j \in J} \lfloor pa_j/a_i \rfloor \bar{x}_j - \lfloor pb/a_i \rfloor + \sum_{j \in I} \bar{x}_j - \lfloor (\beta + \sum_{j \in I} \alpha_j)/a_i \rfloor$
      if $z > z^*$ **then**
        $z^* = z$
        $u_0^* = p/a_i$
        $u_j^* = \alpha_j/a_i \ \forall j \in I$
        $u_j^* = 0 \ \forall j \in J \setminus I$

i.e. $\sum_{j \in I}(u_0 a_j - \lfloor u_0 a_j \rfloor)$ increases. To identify the best set of indices $I \subseteq J$ with a fixed cardinality $|I|$ we can add j to I in a greedy way, starting with the indices with a larger value of $\bar{x}_j + \lambda(u_0 a_j - \lfloor u_0 a_j \rfloor)$, for some $\lambda \in \mathbb{R}_+$. The coefficient λ allows to weigh differently the contribution from $\bar{x}_j$, which would directly increase the objective function, and the one from $u_0 a_j - \lfloor u_0 a_j \rfloor$, that would affect it only indirectly, as could be "canceled" after applying lower integer part. This procedure is described in Algorithm 3.

Algorithm 3 Heuristic 2 (H2)

Require: $a_1, \ldots, a_n, b \in \mathbb{Z}_+$, $\bar{x} \in KP$, $\lambda \in \mathbb{R}_+$
Ensure: $u_0^*, u_1^*, \ldots, u_n^* \in [0, 1)$, $z^* \in \mathbb{R}$
$u_0^* = 0, u_1^* = 0, \ldots, u_n^* = 0, z^* = 0$
for $i \in J = \{j : 0 < \bar{x}_j < 1\}$ **do**
  for $p \in \{1, \ldots, a_i - 1\}$ **do**
    $u_0 = p/a_j$
    Sort $J = \{j_1, \ldots, j_{|J|}\}$ in decreasing order of $\bar{x}_j + \lambda(u_0 a_j - \lfloor u_0 a_j \rfloor)$
    for $k \in \{0, \ldots, |J|\}$ **do**
      $I = \{j_l : l = 1, \ldots, k\}$
      $z = \sum_{j \in J} \lfloor u_0 a_j \rfloor \bar{x}_j + \sum_{j \in I} \bar{x}_j - k - \lfloor u_0 b - \sum_{j \in I}(u_0 a_j - \lfloor u_0 a_j \rfloor) \rfloor$
      if $z > z^*$ **then**
        $z^* = z$
        $u_0^* = u_0$
        $u_j^* = 1 - (u_0 a_j - \lfloor u_0 a_j \rfloor) \ \forall j \in I$
        $u_j^* = 0 \ \forall j \in J \setminus I$

3 Computational Results

To assess their performances, the algorithms in Section 2 were embedded into a cut-and-branch framework. To allow comparisons with similar studies in the literature [16, 2], we adopted as test-bed the most challenging instances –namely D and E– of the Generalized Assignment Problem (GAP) taken from the OR library [3]. We

considered separately the three main approaches presented in Section 2 for computing maximally violated CG cuts:

1. Explicit resolution of the Knapsack subproblems of the form (7) using dynamic programming (EKP), corresponding to Algorithm 1.

2. Heuristic based on Algorithm 2 (H1).

3. Heuristic based on Algorithm 3 (H2).

To assess the performances of EKP, H1, and H2, we also considered the performance of Gurobi (a commercial solver) and SCIP (a non-commercial solver), disabling generation of all cuts except cover cuts. For Gurobi, we varied the parameter for cut aggressiveness, comparing moderate and aggressive cut generation. The results obtained are the same for the two values.

In accordance with the literature [16], as a measure of improvement with respect to the solution of the linear relaxation, we used the closed gap defined as $1 - (LB_{final} - LB_{opt})/(LB_{init} - LB_{opt})$, where LB_{init} is the value of the objective for the linear relaxation, LB_{final} is the value of the relaxation strengthened by adding all the cuts, and LB_{opt} is the optimal value when available or otherwise the best known lower bound. For all instances, the values of LB_{opt} were taken from [2]. For selecting the parameter λ required by H2, a preliminary analysis shows that the highest closed gap was achieved for $\lambda = 1$ in almost all instances. We witness a consistent behavior, with smaller closed gaps for values of λ greater or less than 1. Therefore, we only consider H2 with $\lambda = 1$ when comparing with other methods. For a few instances (namely d401600, d60900, d801600), the linear relaxation already yields the optimal value, thus we did not consider them.

As we may expect, EKP always performs better than all the other approaches in terms of closed gap, probably because of the improvement deriving from using maximally violated CG cuts. For all considered instances, SCIP and Gurobi achieve much lower closed gaps, with Gurobi showing especially poorer performance, possibly because of early stopping, as suggested by the extremely low running times. EKP exhibits running times that are comparable (and sometimes shorter) to both SCIP and Gurobi.

The heuristics H1 and H2 obtain closed gaps that are always lower but close to EKP and generate fewer cuts, especially in the case of large instances. The running times of the heuristics are similar to each other and slightly shorter than those relative to EKP. The magnitude of this difference might have been influenced by the values of the coefficients that are relatively small in the instances considered, thus favoring EKP. Even if the two heuristics use different criteria in their greedy approach, the final performances are similar.

When comparing with similar heuristics present in the literature [16], both H1 and H2 yield consistently better results in terms of closed gap for all instances for which we have data. For [16] we do not have access to runtimes and number of

	SCIP			Gurobi		EKP		
Instance	Cl. Gap	Time	Cuts	Cl. Gap	Time	Cl. Gap	Time	Cuts
e05100	33.04	<1	20	10.25	<1	76.41	<1	110
e05200	29.11	<1	16	21.75	<1	57.03	<1	62
e10100	38.17	<1	22	2.19	<1	65.66	<1	137
e10200	45.58	<1	39	7.03	<1	60.59	<1	169
e10400	53.32	1	38	20.24	1	71.50	<1	119
e15900	35.34	4	38	54.45	2	73.74	1	245
e20100	62.28	1	76	<1.45	<1	91.26	<1	317
e201600	39.35	14	60	21.50	4	66.82	3	378
e20200	55.33	1	69	1.30	1	85.14	<1	242
e20400	50.23	3	63	13.02	1	87.21	1	315
e30900	56.72	13	125	19.74	5	94.91	5	615
e401600	49.64	38	117	35.06	13	83.59	18	897
e40400	56.26	10	132	10.56	3	84.87	3	717
e60900	49.82	50	231	1.62	4	83.28	29	1470
e801600	53.28	207	289	10.93	35	86.37	105	2036
Min	29.11	<1	16	<1.45	<1	57.03	<1	62
Max	62.28	207	289	54.45	35	94.91	105	2036
Mean	47.16	23	89	15.34	5	77.89	11	522
Median	49.82	3	63	10.93	1	83.28	1	315

Table 1: Exact separation vs. solvers for class E. Time in seconds, Closed Gap in %.

	EKP			H1			H2			HL [16]
Instance	Cl. Gap	Time	Cuts	Cl. Gap	Time	Cuts	Cl. Gap	Time	Cuts	Cl. Gap
e05100	76.41	<1	110	76.35	<1	94	74.83	<1	103	63.1
e05200	57.03	<1	62	54.69	<1	63	55.57	<1	66	50.0
e10100	65.66	<1	137	64.17	<1	129	63.37	<1	131	63.4
e10200	60.59	<1	169	57.50	<1	124	57.25	<1	118	55.7
e10400	71.50	<1	119	71.16	<1	100	71.32	<1	121	70.6
e15900	73.74	1	245	72.48	1	200	72.05	1	213	
e20100	91.26	<1	317	90.39	<1	294	90.37	<1	329	86.3
e201600	66.82	3	378	66.08	2	337	65.51	2	308	
e20200	85.14	<1	242	84.52	<1	230	84.36	<1	219	84.4
e20400	87.21	1	315	85.80	1	288	85.42	1	290	84.2
e30900	94.91	5	615	93.87	4	548	92.81	4	522	
e401600	83.59	18	897	80.67	15	793	80.50	15	782	
e40400	84.87	3	717	83.80	3	641	82.64	3	594	81.1
e60900	83.28	29	1470	81.47	24	1283	81.05	24	1222	
e801600	86.37	105	2036	84.95	89	1818	84.10	89	1685	
Min	57.03	<1	62	54.69	<1	63	55.57	<1	66	
Max	94.91	105	2036	93.87	89	1818	92.81	89	1685	
Mean	77.89	11	522	76.53	9	462.8	76.08	9	447	
Median	83.28	1	315	80.67	1	288	80.50	1	290	

Table 2: Exact vs. heuristic separation for class E. Time in seconds, Closed Gap in %.

cuts generated; since results for larger instances are not available, no statistics is computed.

Computational experiments were conducted using a Dell Precision 7960 workstation, equipped with an Intel(R) Xeon(R) W-3495X CPU featuring 56 physical cores and 128 GB of RAM. The operating system used was Ubuntu 22.04 LTS (64-bit).

All the algorithms are implemented in Python (v3.10) using Gurobi (v12.0) as a branch-and-cut framework. The EKP separation algorithm, which requires access to the dynamic programming table to solve the knapsack (sub)problems more efficiently (line 10 in Algorithm 1), is implemented in C++ for efficiency reasons (a first implementation of the very same EKP algorithm in pure Python was two orders of magnitude slower).

The procedure is terminated if an integer solution was obtained or if the violation of all the new CG inequalities was less than 10^{-2}. A maximal number of cut generation rounds was fixed to 200, and no new cut generation round started if the runtime had already exceeded 2 hours.

4 Concluding remarks

In this paper, we introduced a new more efficient exact algorithm and two novel heuristics to optimize over the first Chvátal-Gomory closure of a knapsack set. Our techniques identify maximally violated CG cuts by solving a sequence of related knapsack problems. Extensive computational experiments on benchmark GAP instances from OR-Library show that the proposed approach closes a larger optimality gap compared to a commercial MIP solver. Moreover, the heuristics outperform the state-of-the-art heuristics for optimizing the first CG closure [16].

As a direction for future research, the approaches presented could be combined. For instance, the two heuristics (H1 and H2) could be applied first to generate a violated inequality, using the more expensive exact knapsack procedure (EKP) only to certify that no further violated cut exists. This strategy could accelerate the separation process without compromising its effectiveness in reducing the optimality gap.

Although the present paper focused on GAP instances to facilitate comparison with the literature, future computational studies could evaluate the performance of the proposed methods on a broader class of MILP instances with knapsack constraints, such as those from the MIPLIB library [9].

Acknowledgments

The authors sincerely thank Adam Letchford for bringing to their attention the paper [16] on rank-1 Chvátal-Gomory inequalities for knapsack problems. His recommendation, along with the insightful discussions that followed, provided valuable inspiration for this research.

A Detailed results

Instance	LP	Optimum	SCIP				Gurobi			EKP			
			Bound	Cl. Gap	Time	Cuts	Bound	Cl. Gap	Time	Bound	Cl. Gap	Time	Cuts
d05100	6353	6345.41	6346.6	15.95	<1	22	6346.3	11.13	<1	6349.7	56.69	<1	141
d05200	12742	12736.20	12737.0	13.39	<1	29	12737.0	13.85	<1	12739.3	54.20	1	140
d10100	6347	6323.46	6328.8	22.59	<1	65	6324.9	5.93	<1	6338.5	63.69	1	273
d10200	12430	12418.36	12420.2	15.90	1	58	12419.7	11.72	<1	12424.3	51.30	1	266
d10400	24960	24955.99	24956.7	17.21	1	48	24956.1	2.03	<1	24958.2	54.21	1	263
d15900	55403	55400.47	55400.9	15.58	4	85	55401.0	21.04	1	55402.1	63.99	3	382
d20100	6185	6142.53	6155.0	29.39	1	118	6143.7	2.72	<1	6170.9	66.79	2	488
d201600	97823	97821.35	97821.8	26.77	11	120	97822.0	39.39	2	97822.6	74.61	8	550
d20200	12233	12217.69	12222.4	30.93	2	115	12218.0	2.00	<1	12228.5	70.56	3	510
d20400	24562	24552.44	24555.2	29.12	3	115	24553.0	5.89	1	24559.2	70.27	4	537
d30900	54833	54828.75	54830.0	28.49	11	115	54828.8	0.83	2	54832.0	75.65	15	900
d40400	24350	24347.61	24348.0	16.38	9	369	24347.6	0.01	1	24348.9	55.39	24	2447
Min				13.39	<1	22		0.01	<1		51.30	<1	140
Max				30.93	11	369		39.39	2		75.65	24	2447
Mean				21.81	4	105		9.71	1		63.11	5	575
Median				19.90	1	100		5.91	<1		63.84	2	435

Table 3: Exact separation vs. solvers for class D. Time in seconds, Closed Gap in %.

Instance	LP	Optimum	SCIP				Gurobi			EKP			
			Bound	Cl. Gap	Time	Cuts	Bound	Cl. Gap	Time	Bound	Cl. Gap	Time	Cuts
e05100	12681	12641.42	12654.5	33.04	<1	20	12645.5	10.25	<1	12671.7	76.41	<1	110
e05200	24930	24922.00	24924.3	29.11	<1	16	24923.7	21.75	<1	24926.6	57.03	<1	62
e10100	11577	11543.05	11556.0	38.17	<1	22	11543.8	2.19	<1	11565.3	65.66	<1	137
e10200	23307	23293.86	23299.8	45.58	<1	39	23294.8	7.03	<1	23301.8	60.59	<1	169
e10400	45746	45739.21	45742.8	53.32	1	38	45740.6	20.24	1	45744.1	71.50	<1	119
e15900	102421	102416.61	102418.2	35.34	4	38	102419.0	54.45	2	102419.8	73.74	1	245
e20100	8436	8359.58	8407.2	62.28	1	76	8359.9	0.45	<1	8429.3	91.26	<1	317
e201600	180645	180640.29	180642.1	39.35	14	60	180641.3	21.50	4	180643.4	66.82	3	378
e20200	22379	22355.93	22368.7	55.33	1	69	22356.2	1.30	1	22375.6	85.14	<1	242
e20400	44877	44861.76	44869.4	50.23	3	63	44863.7	13.02	1	44875.1	87.21	1	315
e30900	100427	100413.32	100421.1	56.72	13	125	100416.0	19.74	5	100426.3	94.91	5	615
e401600	178293	178282.62	178287.8	49.64	38	117	178286.3	35.06	13	178291.3	83.59	18	897
e40400	44561	44523.43	44544.6	56.26	10	132	44527.4	10.56	3	44555.3	84.87	3	717
e60900	100149	100103.26	100126.0	49.82	50	231	100104.0	1.62	4	100141.4	83.28	29	1470
e801600	176820	176780.99	176801.8	53.28	207	289	176785.3	10.93	35	176814.7	86.37	105	2036
Min				29.11	<1	16		0.45	<1		57.03	<1	62
Max				62.28	207	289		54.45	35		94.91	105	2036
Mean				47.16	23	89		15.34	5		77.89	11	522
Median				49.82	3	63		10.93	1		83.28	1	315

Table 4: Exact separation vs. solvers for class E. Time in seconds, Closed Gap in %.

Instance	LP	Optimum	EKP				H1				H2			
			Bound	Cl. Gap	Time	Cuts	Bound	Cl. Gap	Time	Cuts	Bound	Cl. Gap	Time	Cuts
d05100	6353	6345.41	6349.7	56.69	<1	141	6349.6	55.52	<1	106	6349.6	55.39	<1	125
d05200	12742	12736.20	12739.3	54.20	1	140	12739.3	54.03	<1	130	12739.3	54.10	<1	136
d10100	6347	6323.46	6338.5	63.69	1	273	6338.4	63.41	<1	240	6338.3	63.02	<1	257
d10200	12430	12418.36	12424.3	51.30	1	266	12424.2	50.46	<1	257	12424.2	50.29	<1	286
d10400	24960	24955.99	24958.2	54.21	1	263	24958.1	53.29	<1	231	24958.1	53.21	<1	244
d15900	55403	55400.47	55402.1	63.99	3	382	55402.1	63.20	2	375	55402.1	63.48	1	367
d20100	6185	6142.53	6170.9	66.79	2	488	6170.8	66.60	1	481	6170.8	66.51	1	487
d201600	97823	97821.35	97822.6	74.61	8	550	97822.6	73.58	5	514	97822.6	73.58	4	481
d20200	12233	12217.69	12228.5	70.56	3	510	12228.4	69.86	1	482	12228.4	69.73	1	509
d20400	24562	24552.44	24559.2	70.27	4	537	24559.1	69.90	2	519	24559.1	69.80	2	499
d30900	54833	54828.75	54832.0	75.65	15	900	54831.9	75.01	8	862	54831.9	74.71	8	860
d40400	24350	24347.61	24348.9	55.39	24	2447	24348.9	55.39	33	2973	24348.9	54.51	31	2846
Min				51.30	<1	140		50.46	<1	106		50.29	<1	125
Max				75.65	24	2447		75.01	33	2973		74.71	31	2846
Mean				63.11	5	575		62.52	4	598		62.36	4	591
Median				63.84	2	435		63.31	1	428		63.25	1	424

Table 5: Exact vs. heuristic separation for class D. Time in seconds, Closed Gap in %.

Instance	LP	Optimum	EKP				H1				H2			
			Bound	Cl. Gap	Time	Cuts	Bound	Cl. Gap	Time	Cuts	Bound	Cl. Gap	Time	Cuts
e05100	12681	12641.42	12671.7	76.41	<1	110	12671.6	76.35	<1	94	12671.0	74.83	<1	103
e05200	24930	24922.00	24926.6	57.03	<1	62	24926.4	54.69	<1	63	24926.4	55.57	<1	66
e10100	11577	11543.05	11565.3	65.66	<1	137	11564.8	64.17	<1	129	11564.6	63.37	<1	131
e10200	23307	23293.86	23301.8	60.59	<1	169	23301.4	57.50	<1	124	23301.4	57.25	<1	118
e10400	45746	45739.21	45744.1	71.50	<1	119	45744.0	71.16	<1	100	45744.1	71.32	<1	121
e15900	102421	102416.61	102419.8	73.74	1	245	102419.8	72.48	1	200	102419.8	72.05	1	213
e20100	8436	8359.58	8429.3	91.26	<1	317	8428.7	90.39	<1	294	8428.6	90.37	<1	329
e201600	180645	180640.29	180643.4	66.82	3	378	180643.4	66.08	2	337	180643.4	65.51	2	308
e20200	22379	22355.93	22375.6	85.14	<1	242	22375.4	84.52	<1	230	22375.4	84.36	<1	219
e20400	44877	44861.76	44875.1	87.21	1	315	44874.8	85.80	1	288	44874.8	85.42	1	290
e30900	100427	100413.32	100426.3	94.91	5	615	100426.2	93.87	4	548	100426.0	92.81	4	522
e401600	178293	178282.62	178291.3	83.59	18	897	178291.0	80.67	15	793	178291.0	80.50	15	782
e40400	44561	44523.43	44555.3	84.87	3	717	44554.9	83.80	3	641	44554.5	82.64	3	594
e60900	100149	100103.26	100141.4	83.28	29	1470	100140.5	81.47	24	1283	100140.3	81.05	24	1222
e801600	176820	176780.99	176814.7	86.37	105	2036	176814.1	84.95	89	1818	176813.8	84.10	89	1685
Min				57.03	<1	62		54.69	<1			55.57	<1	66
Max				94.91	105	2036		93.87	89			92.81	89	1685
Mean				77.89	11	522		76.53	9			76.08	9	447
Median				83.28	1	315		80.67	1			80.50	1	290

Table 6: Exact vs. heuristic separation for class E. Time in seconds, Closed Gap in %.

B Cut Generation Outline

In the following, we summarize the main steps for computing the coefficients $(u_0, u_1, \ldots, u_n)$ representing a violated CG inequality, arising from a constraint $\sum_{j \in N} a_j x_j \leq b$, given $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n) \in KP$.

1. We retrieve the set of indices $J_0 \coloneqq \{j \in N : \bar{x}_j = 0\}$. As mentioned at the beginning of Section 2, we can restrict to $N_0 \coloneqq N \setminus J_0$, and fix $u_j = 0 \; \forall j \in J_0$.

2. In a similar way we can discard the components such that $\bar{x}_j = 1$, by considering the complemented variables. We retrieve the set of indices $J_1 \coloneqq \{j \in N_0 : \bar{x}_j = 1\}$, then we update the value of b to $b' = b - \sum_{j \in J_1} a_j$ and we restrict to $N_1 \coloneqq N_0 \setminus J_1$. Once we have computed the optimal value $u_0^* = p^*/a_i^*$ for the first coefficient, we can compute $u_j \; \forall j \in J_1$ in the following way:

$$u_j = \begin{cases} 0 & \text{if } u_0^* a_j \in \mathbb{Z} \iff a_j p^* \mod a_i^* = 0 \\ 1 - (u_0 a_j - \lfloor u_0 a_j \rfloor) & \text{otherwise.} \end{cases}$$

3. We then consider each $a_i \in \{a_j : j \in N_1\}$, that is, if a value appears multiple times we consider it only once. We can furthermore ignore all a_i for which there exists a_j with $j \neq i$ such that $a_j/a_i = q \in \mathbb{Z}$. In fact, we can express any c_k of the form $c_k = p/a_i$ as $c_k = pq/a_j$.

4. For a fixed a_i, we consider each $p \in \{1, \ldots, a_i - 1\}$.

5. For a fixed a_i and p, we can restrict to $N_2 \coloneqq N_1 \setminus J_2$, where $J_2 = J_2(a_i, p) \coloneqq \{j \in N_1 : a_j p/a_i \in \mathbb{Z}\} = \{j \in N_1 : a_j p \mod a_i = 0\}$, since the proof of Theorem 2 shows that there exists an optimal solution with $u_j = 0 \; \forall j \in J_2$.

6. We apply either EKP, H1 or H2, considering a_j for $j \in N_2$, b'.

7. When applying EKP or H1, we vary the maximal capacity of the associated knapsack problem as $(k+1)a_i - 1 - \beta$ with k between 0 and $\lfloor (\beta + \sum_{j \in N_2} \alpha_j)/a_i \rfloor$.

We can implement step 3 and 4 as double for loop. However, since only the ratio u_0 is relevant for computing the coefficients, this implementation may lead to control the same value $u_0 = p/a$ multiple times, thus performing redundant operations. For instance, if $a_1 = 16$ and $a_2 = 24$, the values $2/16 = 3/24$, $4/16 = 6/24$, $6/16 = 9/24$, $8/16 = 12/24$, $10/16 = 15/24$, $12/16 = 18/24$, $14/16 = 21/24$ will be checked two times, thus leading in this case 18% of redundant coefficients. More complex implementations may avoid or reduce the number of these redundant operations.

References

[1] Achterberg, T. Conflict analysis in mixed integer programming. *Discrete Optimization 4*, 1 (2007), 4–20.

[2] Avella, P., Boccia, M., and Vasilyev, I. A computational study of exact knapsack separation for the generalized assignment problem. *Computational Optimization and Applications 45* (2010), 543–555.

[3] Beasley, J. E. OR-library: distributing test problems by electronic mail. *Journal of the Operational Research Society 41*, 11 (1990), 1069–1072.

[4] Bolusani, S., Besançon, M., Bestuzheva, K., Chmiela, A., Dionísio, J., Donkiewicz, T., van Doornmalen, J., Eifler, L., Ghannam, M., Gleixner, A., Graczyk, C., Halbig, K., Hedtke, I., Hoen, A., Hojny, C., van der Hulst, R., Kamp, D., Koch, T., Kofler, K., Lentz, J., Manns, J., Mexi, G., Mühmer, E., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Turner, M., Vigerske, S., Weninger, D., and Xu, L. The SCIP Optimization Suite 9.0. Technical report, Optimization Online, February 2024.

[5] Chvátal, V. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics 4*, 4 (1973), 305–337.

[6] Conforti, M., Cornuéjols, G., and Zambelli, G. *Integer Programming*. Springer, 2014.

[7] Eisenbrand, F. On the membership problem for the elementary closure of a polyhedron. *Combinatorica 19* (1999), 297–300.

[8] Fischetti, M., and Lodi, A. Optimizing over the first Chvátal closure. *Mathematical Programming 110*, 1 (2007), 3–20.

[9] Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation 13*, 3 (2021), 443–490.

[10] Gomory, R. E. Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. In *50 Years of Integer Programming 1958-2008: From the early years to the State-of-the-Art*. Springer, 2009, pp. 77–103.

[11] Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing 10*, 4 (1998), 427–437.

[12] Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming 85* (1999), 439–467.

[13] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[14] Kaparis, K., and Letchford, A. N. Separation algorithms for 0-1 knapsack polytopes. *Mathematical Programming 124* (2010), 69–91.

[15] Lee, K. A cutting-plane generation method for a variable-capacity (0, 1)-knapsack problem with general integer variables. *Management Science and Financial Engineering 10*, 1 (2004), 97–106.

[16] Lee, K.-S. Separation heuristic for the rank-1 Chvátal-Gomory inequalities for the binary knapsack problem. *Journal of Korean Institute of Industrial Engineers 38*, 2 (2012), 74–79.

[17] Letchford, A. N., and Souli, G. New valid inequalities for the fixed-charge and single-node flow polytopes. *Operations Research Letters 47*, 5 (2019), 353–357.

[18] Marchand, H., and Wolsey, L. A. Aggregation and mixed integer rounding to solve mips. *Operations Research 49*, 3 (2001), 363–371.

[19] Park, K.-C., and Lee, K.-S. 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 (2011), 43–50.

[20] Wolsey, L. A., and Nemhauser, G. L. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

[21] Xpress, F. Fico Xpress optimization suite.