Optimization Letters manuscript No.
(will be inserted by the editor)

On the exact separation of cover inequalities of maximum-depth

Daniele Catanzaro · Stefano Coniglio · Fabio Furini

Received: date / Accepted: date

Abstract We investigate the problem of separating cover inequalities of maximum-depth exactly. We propose a pseudopolynomial-time dynamic-programming algorithm for its solution, thanks to which we show that this problem is weakly $\mathcal{NP}$-hard (similarly to the problem of separating cover inequalities of maximum violation). We carry out extensive computational experiments on instances of the knapsack and the multi-dimensional knapsack problems with and without conflict constraints. The results show that, with a cutting-plane generation method based on the maximum-depth criterion, we can optimize over the cover-inequality closure by generating a number of cuts smaller than when adopting the standard maximum-violation criterion. We also introduce the Point-to-Hyperplane Distance Knapsack Problem (PHD-KP), a problem closely related to the separation problem for maximum-depth cover inequalities, and show how the proposed dynamic programming algorithm can be adapted for effectively solving the PHD-KP as well.

Keywords Knapsack Problem · Cover Inequalities · Dynamic Programming · Mixed Integer Nonlinear Programming · Cutting Plane Generation

1 Introduction

The separation of *maximum-violation* inequalities is the standard approach for generating valid inequalities in Mixed Integer Linear Programs (MILPs) [33]. In the literature, however, some authors have speculated that separating (fractional) solutions by using a criterion different from maximizing the *cut violation* could be beneficial for the performance of the cutting-plane methods [1, 3, 4, 6, 7, 14]. The maximization of the *cut depth* [6] (also referred to as "geometric distance" [7] or "efficacy" [4, 11]) is one such criterion. It is, in particular, one of the criteria used in, e.g., SCIP during the *cut-selection* phase in which a subset of cuts is selected from the cut pool to be added to strengthen the formulation of the problem [1]. Previous works, including [28, 34, 37], tackled the problem of separating inequalities of maximum depth only heuristically, due to its intrinsic difficulty.

In this work, we focus on the case of *cover inequalities* and address the problem of their exact separation according to the maximum-depth criterion. The main theoretical result of our paper (see Section 5) is the characterization of the computational complexity of this problem.

Daniele Catanzaro
Center for Operations Research and Econometrics (CORE), Université Catholique de Louvain, Louvain-la-Neuve, Belgium.
E-mail: daniele.catanzaro@uclouvain.be

Stefano Coniglio
Department of Mathematical Sciences, University of Southampton, Southampton, UK.
E-mail: s.coniglio@soton.ac.uk

Fabio Furini
Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", Consiglio Nazionale delle Ricerche (IASI-CNR), Roma, Italy.
E-mail: f.furini@iasi.cnr.it

To this end, we develop an exact pseudopolynomial-time dynamic programming algorithm by means of which we show that the maximum-depth separation problem is weakly $\mathcal{NP}$-hard. This result shows that, even if it entails (as we will show) the optimization of a hyperbolic objective function, the complexity class of this separation problem is the same as for the standard one based on the maximum-violation criterion. A second contribution of the paper (see Section 6) is the introduction of the Point-to-Hyperplane Distance Knapsack Problem (PHD-KP), which, similarly to the problem of separating maximum-depth inequalities, is a knapsack problem with a hyperbolic objective function. We show that even this problem is weakly $\mathcal{NP}$-hard and that it can be solved by an adaptation of the dynamic programming algorithm we propose for the separation problem of maximum depth.

Computational experiments (see Section 7) carried out on several classes of instances reveal that the adoption of the cut depth allows for optimizing over the cover-inequality closure by generating a number of cuts smaller than with the standard maximum-violation criterion. Interestingly, our experiments also reveal that a reduction in the number of cutting planes, inferior but similar to the one achieved with the maximum-depth criterion, can be obtained by separating maximum-violation inequalities and making the corresponding covers minimal via an *a posteriori* procedure. Finally, computational experiments (see Section 8) on several classes of PHD-KP instances show that our dynamic-programming algorithm can solve the problem more efficiently than a state-of-the-art nonlinear-programming solver.

2 On the maximum-violation cover inequalities

Given a set of items $N := \{1, 2, \dots, n\}$, a non-negative integer *weight* w_j for each item $j \in N$, and positive *knapsack capacity* c, consider the *knapsack constraint*

$$\sum_{j \in N} w_j \, y_j \le c,$$

where each variable $y_j \in \{0, 1\}$ encodes the decision of inserting (or not) item $j \in N$ into the knapsack.[1] One (or several) such constraints are present in countless real-world applications involving the solution of a MILP [23–25, 30].

A *cover* is a subset of items $C \subset N$ with a total weight exceeding the knapsack capacity. This notion plays a central role in determining strong valid inequalities for the convex hull of the solutions to optimization problems featuring one or more knapsack constraints. In particular, since no more than $|C| - 1$ items of a given cover $C \subset N$ can be simultaneously inserted into the knapsack, the following family of constraints, called *cover inequalities*, are valid for the convex hull of the problem:

$$\sum_{j \in C} y_j \le |C| - 1 \qquad \forall\, C \subset N : \sum_{j \in C} w_j \ge c + 1. \tag{1}$$

In either its original form (1) or in its lifted form (see the survey [23]), this exponentially-large family of constraints constitutes one of the earliest combinatorial inequalities adopted in a general-purpose 0-1 Integer Programming (IP) solver [17], and it it still routinely generated (heuristically) in most state-of-the-art MILP solvers.

Given a (fractional) point $\bar{y} \in [0, 1]^n$, the *Separation Problem* (SP) for the cover inequalities calls for an inequality of type (1) with a strictly positive *cut violation*, i.e., with

$$\sum_{j \in \bar{C}} \bar{y}_j - |\bar{C}| + 1 > 0,$$

or for a proof that no such inequality exists. The standard solution approach for solving the SP consists in searching for a cover with a cut violation as large as possible. By introducing, for each

[1] Non trivial instances have $0 \le w_j \le c$, for all $j \in N$, and $c < \sum_{j \in N} w_j$.

$j \in N$, a decision variable $z_j \in \{0,1\}$ equal to 1 if and only if item j belongs to the cover, this problem can be formulated in terms of the following *Integer Linear Program* (ILP):

$$\max_{z\in\{0,1\}^n} \left\{ \sum_{j\in N} (\bar{y}_j - 1)\, z_j + 1 : \quad \sum_{j\in N} w_j\, z_j \geq c+1 \right\}.$$

By *complementing* the z variables, i.e., by introducing a new set of x variables such that $x_j := 1 - z_j$ for each $j \in N$, the *Maximum-Violation Separation Problem* (MV-SP) can be formulated in terms of the following *Knapsack Problem* (KP):

$$v := \max_{x\in\{0,1\}^n} \left\{ \sum_{j\in N} q_j\; x_j + \Psi : \quad \sum_{j\in N} w_j\, x_j \leq \Phi \right\}, \tag{MV-SP}$$

where $q_j := 1 - \bar{y}_j$, for all $j \in N$, $\Psi := \sum_{j\in N} \bar{y}_j - n + 1$, $\Phi := \sum_{j\in N} w_j - (c+1)$. We refer to v as the *maximum cut violation*. Let x^* denote an optimal solution to the (MV-SP) and let $C^* := \{j \in N : x^*_j = 0\}$. If $v > 0$, the cover inequality corresponding to C^* is maximally violated by $\bar{y}$, whereas, if $v \leq 0$, we have a proof that no violated cover inequality exists.

3 On the maximum-depth cover inequalities

Given a cover $C \subset N$, let

$$H_C := \left\{ y \in \mathbb{R}^n : \sum_{j\in C} y_j = |C| - 1 \right\}$$

be the hyperplane consisting in the set of points $y \in \mathbb{R}^n$ for which the inequality $\sum_{j\in C} y_j \leq |C| - 1$ is *tight*. Given a point $\bar{y} \in [0,1]^n$, the *depth* of the cover inequality corresponding to C is defined as the *Euclidean* (or 2-norm) *point-to-hyperplane distance* between $\bar{y}$ and the point belonging to H_C that is closest to $\bar{y}$ in Euclidean norm—such a point is often called the *projection* of $\bar{y}$ onto H_C. We denote this distance by $d_2(\bar{y}, H_C)$. Letting $z \in \{0,1\}^n$ be the characteristic vector of C, we have:

$$d_2(\bar{y}, H_C) := \frac{\left| \sum_{j\in N} \bar{y}_j z_j - |C| + 1 \right|}{||z||_2}, \tag{2}$$

where $||z||_2$ is the Euclidean norm (or 2-norm) of z. A derivation of this classical result using the tools of nonlinear optimization can be found in [29]. Besides the aforementioned works belonging to the cutting-plane literature in which the cut depth has been adopted, articles in which such a distance is either maximized or minimized include [2, 10, 13]. We report an illustration in the following example.

Example 1 Consider a point $\bar{y} = (0.75, 0.76, 0.5) \in \mathbb{R}^3$ and two covers $C_1 = \{3\}$ and $C_2 = \{1,2\}$. The cover inequalities corresponding to C_1 and C_2 are $y_3 \leq 0$ and $y_1 + y_2 \leq 1$. The corresponding hyperplanes H_{C_1} and H_{C_2} have equations $y_3 = 0$ and $y_1 + y_2 = 1$. The Euclidean point-to-hyperplane distance between $\bar{y}$ and the two hyperplanes is $d_2(\bar{y}, H_{C_1}) = \frac{|0.5-1+1|}{\sqrt{1}} = 0.5$ and $d_2(\bar{y}, H_{C_2}) = \frac{|0.75+0.76-2+1|}{\sqrt{2}} = \frac{0.51}{\sqrt{2}} \simeq 0.36$. Hence, H_{C_1} is farther away from $\bar{y}$ than H_{C_2} in terms of the Euclidean 2-norm point-to-hyperplane distance, whereas, in terms of cut violation, C_1 has a smaller violation (of $0.5 - 1 + 1 = 0.5$) than C_2 (which has a larger violation of $0.75 + 0.76 - 2 + 1 = 0.51$). A graphical representation is reported in Figure 1. □

The separation problem calling for a cover inequality of maximum Euclidean-norm point-to-hyperplane distance, i.e., of *maximum depth*, can be cast as the following Mixed Integer Non-Linear Program (MINLP):

$$\max_{z\in\{0,1\}^n} \left\{ \frac{\sum_{j\in N} (\bar{y}_j - 1)\; z_j + 1}{||z||_2} : \quad \sum_{j\in N} w_j\, z_j \geq c+1 \right\}. \tag{3}$$

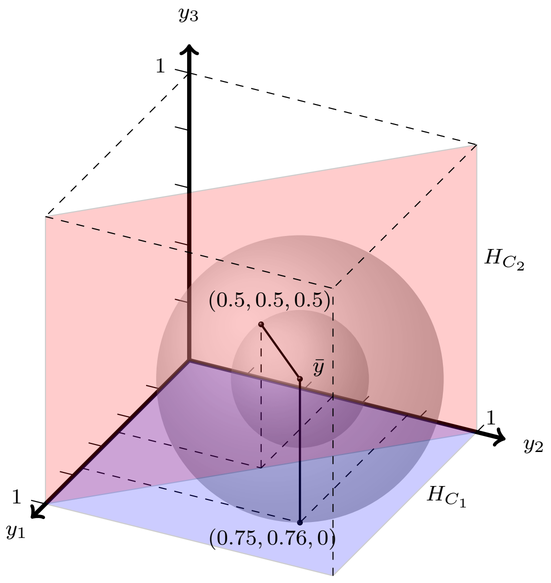

Fig. 1 An illustration of Example 1. The two solid segments connect $\bar{y}$ to the closest point in Euclidean-norm point-to-hyperplane distance on either hyperplane H_{C_1} and H_{C_2}. According to such distance, H_{C_1} is farther away from $\bar{y}$ than H_{C_2}.

We call the corresponding problem the *Maximum-Depth Separation Problem* (MD-SP). The objective function, obtained from (2) by dropping the absolute value, corresponds to the opposite of the *signed* 2-norm point-to-hyperplane distance between $\bar{y}$ and the set of all points $y \in \mathbb{R}^n$ that satisfy the cover inequality induced by $C := \{j \in N : z_j = 1\}$. Thus, it is equal to $-d_2(\bar{y}, H_C)$ if $\bar{y}$ satisfies such an inequality, and to $d_2(\bar{y}, H_C)$ otherwise.

Since the z variables are binary, $||z||_2 = \sqrt{\sum_{j\in N} |z_j|^2} = \sqrt{\sum_{j\in N} z_j}$. Complementing such variables as before, i.e., by introducing a binary variable $x_j := 1 - z_j$ for each $j \in N$, we obtain the following equivalent MINLP formulation:

$$\max_{x\in\{0,1\}^n} \left\{ \frac{\sum_{j\in N} q_j\ x_j + \Psi}{\sqrt{n - \sum_{j\in N} x_j}} : \quad \sum_{j\in N} w_j\, x_j \le \Phi \right\}.$$

The irrationality of the objective function prevents us from solving the problem exactly (i.e., up to an infinite precision) on a Turing machine. However, by discarding the solutions to the problem having a non-positive violation, we can, without loss of generality, raise the objective function to the second power. We obtain the following problem:

$$d := \max_{x\in\{0,1\}^n} \left\{ \frac{\left(\sum_{j\in N} q_j\ x_j + \Psi\right)^2}{n - \sum_{j\in N} x_j} : \quad \sum_{j\in N} w_j\, x_j \le \Phi, \quad \sum_{j\in N} q_j\ x_j + \Psi > 0 \right\}, \qquad \text{(MD-SP)}$$

where d is the square of the maximum cut-depth. Let x^* be an optimal solution to the (MD-SP) and let $C^* := \{j \in N : x^*_j = 0\}$. If the problem is feasible, which implies $d > 0$, C^* corresponds to a violated cover inequality of maximum depth. In contrast, if the problem is infeasible, we have a proof that no violated cover inequality exists. We note that the objective function is well-defined as its denominator is equal to 0 only for $x_j = 1$ for all $j \in N$, which is infeasible because $\sum_{j\in N} w_j \not\le \Psi = \sum_{j\in N} w_j - (c+1)$.

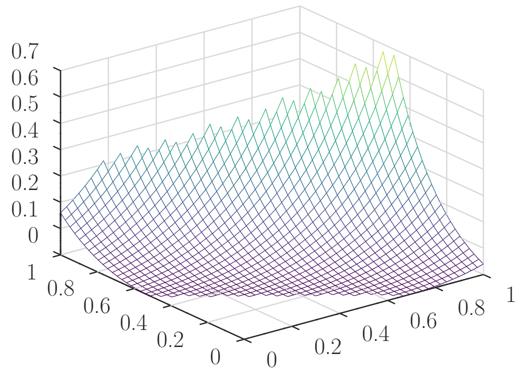

Fig. 2 An illustration of the non-concave objective function of the (MD-SP) for an instance with $n = 2$, $\bar{y}_1 = 0.4$, $\bar{y} = 0.2$, $q_1 = 0.6$, $q_2 = 0.8$, $\Psi = -0.4$, and $\Phi = 9$, subject to the constraint $5\,x_1 + 8\,x_2 \leq \Phi$. The objective function, which reads $\frac{(0.6\,x_1+0.8\,x_2-0.4)^2}{2-x_1-x_2}$, is plotted only for the $(x_1, x_2) \in [0,1]^2$ pairs which are feasible, i.e., which satisfy $5\,x_1 + 8\,x_2 \leq \Phi$ and $0.6\,x_1 + 0.8\,x_2 - 0.4 > 0$.

4 On the nature of cover inequalities of maximum-depth

The (MD-SP) can be interpreted as a variant of the (MV-SP) in which, rather than a single objective function, the ratio of two different objective functions is maximized: the item profit incremented by the constant Ψ (squared) and the cardinality of the cover, which corresponds to the difference between n and the number of items contained in a solution to the (MD-SP). Due to this ratio, in the (MD-SP) we look for solutions with a large numerator and a small denominator. This results in a non-concave objective function, as shown in Figure 2.

It is well-known that a cover inequality is non-dominated whenever the corresponding cover C is *inclusion-wise minimal*, i.e., whenever, for any $j \in C$, $C \setminus \{j\}$ is not a cover. When separating maximum-depth cover inequalities, the presence of the denominator of the (MD-SP) implies the following:

Proposition 1 *Any cover corresponding to an optimal solution to the* (MD-SP) *is minimal.*

Proof Let x^* be an optimal solution to the (MD-SP) and let $C^* := \{j \in N : x^*_j = 0\}$ be the corresponding cover. If C^* is not minimal, there exists a minimal cover $\tilde{C}$ with $\tilde{C} \subset C^*$. Let $\tilde{x}_j \in \{0,1\}^n$ such that $\tilde{C} = \{j \in N : \tilde{x}_j = 0\}$. As $\tilde{C} \subset C^*$, there is some $j \in N$ with $j \in C^*$ and $j \notin \tilde{C}$. Thus, $x^*_j = 0$ and $\tilde{x}_j = 1$, implying that $\tilde{x}$ achieves a numerator at least as large as x^* and a strictly smaller denominator, which shows that x^* is not optimal. □

Besides this minimality property, the (MD-SP) establishes a trade-off between the cut violation (in the numerator) and the cardinality of the cover (in the denominator), often leading to solutions with a less-than-optimal violation which is compensated by a smaller cardinality. This is illustrated by the following numerical example, which also shows how the optimal solutions to the (MD-SP) and the (MV-SP) can be different even when they correspond to minimal covers.

Example 2 Consider an instance of the (MV-SP) and the (MD-SP) with $n = 4$ items, weights $w_1 = 3$, $w_2 = 1$, $w_3 = 2$, $w_4 = 5$, profits $q_1 = 0.25$, $q_2 = 0.23$, $q_3 = 0.24$, $q_4 = 0.5$, capacity $c = 5$. With these values, we have $\Phi = 5$ and $\Psi = -0.22$. Consider two solutions, $\{1,3\}$ and $\{4\}$, which correspond to the covers $\{2,4\}$ and $\{1,2,3\}$, both of which are minimal. From the perspective of the (MV-SP), $\{1,3\}$ is suboptimal (it has value $0.25 + 0.24 - 0.22 = 0.27$), whereas $\{4\}$ is optimal (it has value $0.5 - 0.22 = 0.28$). From the perspective of the (MD-SP), $\{1,3\}$ is optimal (it has value $\frac{(0.25+0.24-0.22)^2}{4-2} \approx 0.036$), whereas $\{4\}$ is suboptimal (it has value $\frac{(0.5-0.22)^2}{4-1} \approx 0.026$). This difference is due to the fact that, in the (MD-SP), the smaller numerator of $\{1,3\}$ is compensated by a smaller denominator, whereas $\{4\}$, which has a larger numerator, is penalized by a larger denominator. □

5 An exact dynamic programming algorithm for the (MD-SP)

In this section, we discuss the computational complexity of the (MD-SP) and we present a solution algorithm based on *Dynamic Programming* (DP) to solve it to optimality. Thanks to such an algorithm, we also show that the problem is weakly $\mathcal{NP}$-hard.

The denominator of the objective function of the (MD-SP) takes discrete values in the set $\{1, \ldots, n-1\}$ (recall that the (MD-SP) is infeasible for $k = n$) as a function of the number of items in the solution. It is not difficult to see that, by removing any item j from an optimal solution with $k \in \{1, \ldots, n-1\}$ items, the remaining set of items in the solution must be optimal for the subproblem with capacity $\Phi - w_j$, item set $N \setminus \{j\}$, and containing $k-1$ items, as, if not, the solution with k items is not optimal. Thus, the (MD-SP) exhibits the so-called *optimal-substructure property*, which can be exploited by an approach relying on Bellman's recursion [25, 30].

Based on this observation, we now derive a Dynamic Programming (DP) algorithm for the (MD-SP). For each item $j \in N$, integer $k \in \{1, \ldots, n-1\}$, and capacity value $s \in \{0, \ldots, \Phi\}$, we denote by $f_j^k(s)$ the maximum total profit in terms of the q values increased by the constant Ψ (this corresponds to the numerator of the objective function of (MD-SP)) that is achievable by choosing k items belonging to $\{1, \ldots, j\} \subseteq N$ with a total weight no larger than $s \leq \Phi$. The following proposition holds:

Proposition 2 *The* (MD-SP) *can be solved in $O(n^2 \Phi)$, and its optimal solution value is:*

$$d = \max_{k \in \{1, \ldots, n-1\}} \left\{ \frac{f_n^k(\Phi)^2}{(n-k)} : f_n^k(\Phi) > 0 \right\},$$

whereas, if $f_n^k(\Phi) \leq 0$ for all $k \in \{1, \ldots, n-1\}$, the (MD-SP) *is infeasible.*

Proof For $j = 1$ and $s \in \{0, \ldots, \Phi\}$, the following holds for all $k \in \{1, \ldots, n-1\}$:

$$f_1^k(s) = \begin{cases} q_1 + \Psi & \text{if } k = 1 \text{ and } w_1 \leq s \\ -\infty & \text{if } k = 1 \text{ and } w_1 > s \\ -\infty & \text{if } k \geq 2. \end{cases} \tag{4}$$

The last case is due to the fact that no more than a single item can be chosen for $j = 1$.

For $j \in \{2, \ldots, n\}$, $s \in \{w_j, \ldots, \Phi\}$, and $k = 1$, we have that either (i) item j is not chosen, leading to $f_j^1(s) = f_{j-1}^1(s)$, or (ii) item j is the only chosen item, leading to $f_j^1(s) = q_j + \Psi$. Therefore:

$$f_j^1(s) = \max\left\{ f_{j-1}^1(s), q_j + \Psi \right\}. \tag{5}$$

Instead, for $j \in \{2, \ldots, n\}$, $s \in \{0, \ldots, w_j - 1\}$, and $k = 1$ we have $f_j^1(s) = f_{j-1}^1(s)$ as item j does not fit.

For $j \in \{2, \ldots, n\}$, $s \in \{w_j, \ldots, \Phi\}$, and $k \in \{2, \ldots, n-1\}$, we have that either (i) item j is not chosen, leading to $f_j^k(s) = f_{j-1}^k(s)$, or (ii) item j is chosen, leading to $f_j^k(s) = f_{j-1}^{k-1}(s - w_j) + q_j$. Hence:

$$f_j^k(s) = \max\left\{ f_{j-1}^{k-1}(s - w_j) + q_j, f_{j-1}^k(s) \right\}. \tag{6}$$

Instead, for $j \in \{2, \ldots, n\}$, $s \in \{0, \ldots, w_j - 1\}$, and $k \in \{2, \ldots, n-1\}$ we have $f_j^k(s) = f_{j-1}^k(s)$ as item j does not fit.

The square of the optimal solution value of the (MD-SP) is obtained by evaluating the largest value taken by $\frac{(f_n^k(\Phi))^2}{(n-k)}$ over $k \in \{1, \ldots, n-1\}$, and by ignoring any case with $f_n^k(\Phi) \leq 0$. This last step takes $O(n)$. If $f_n^k(\Phi) \leq 0$ for all k, the problem is infeasible and $d = -\infty$. Computing $f_j^k(s)$ for all $j \in N$, $k \in \{1, \ldots, n-1\}$, and $s \in \{0, \ldots, \Phi\}$ via (4), (5), and (6) takes $O(n^2 \Phi)$. As, by storing the values of $f_j^k(s)$ for all $j \in N$ and $s \in \{0, \ldots, \Phi\}$, an optimal solution can be reconstructed in linear time by backtracking. The statement follows. □

As a consequence of the pseudopolynomial-time dynamic programming algorithm illustrated in Proposition 2, the following holds:

Table 1 Execution of the DP algorithm on the instance of Example 3. For each item of index $j \in N$, the corresponding part of the table reports the value of $f_j^k(s)$ for all values of s and k.

	$j=1$							$j=2$					
	$s=0$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$		$s=0$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$
$k=1$	$-\infty$	$-\infty$	$-\infty$	0.03	0.03	0.03	$k=1$	$-\infty$	0.01	0.01	0.03	0.03	0.03
$k=2$	-	-	-	-	-	-	$k=2$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0.26	0.26
	$j=3$							$j=4$					
	$s=0$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$		$s=0$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$
$k=1$	$-\infty$	0.01	0.02	0.03	0.03	0.03	$k=1$	$-\infty$	0.01	0.02	0.03	0.03	0.28
$k=2$	$-\infty$	$-\infty$	$-\infty$	0.25	0.26	0.27	$k=2$	$-\infty$	$-\infty$	$-\infty$	0.25	0.26	0.27

Corollary 1 *The* (MD-SP) *is weakly $\mathcal{NP}$-hard.*

Proof The separation problem calling for a cover inequality of strictly positive violation (or for a proof that no such inequality exists) is known to be weakly $\mathcal{NP}$-hard [26]. As any algorithm for the solution of the (MD-SP) implicitly solves such a separation problem as well, the (MD-SP) is *at least* weakly $\mathcal{NP}$-hard. As the DP algorithm we proposed runs in pseudopolynomial time, we deduce that the problem is weakly $\mathcal{NP}$-hard. □

We note that it is not necessary to compute $f_j^k(s)$ for all the values of $k \in \{1, \dots, n-1\}$. In fact, it suffices to consider $k \in \{1, \dots, \bar{k}\}$ where

$$\bar{k} := \max_{x \in \{0,1\}^n} \left\{ \sum_{j \in N} x_j : \sum_{j \in N} w_j \, x_j \le \Phi \right\}.$$

This value can be computed in $O(n \log n)$ by sorting the items in N in non-increasing order of weights and by adding them to the solution in that order until the capacity Φ is saturated. This observation reduces the complexity of the algorithm to $O(n\,\bar{k}\,\Phi)$. An additional further speed-up can be obtained by considering, in the recursion, values of k up to the minimum between $\bar{k}$ and the index j of the item currently under examination.

We conclude this section by showing an example of execution of the DP algorithm.

Example 3 Consider again the instance of Example 2, with $n = 4$, weights $w_1 = 3$, $w_2 = 1$, $w_3 = 2$, $w_4 = 5$, profits $q_1 = 0.25$, $q_2 = 0.23$, $q_3 = 0.24$, $q_4 = 0.5$, $\Phi = 5$, and $\Psi = -0.22$. The maximum number of items that can be simultaneously chosen is $\bar{k} = 2$. The values of the recursive function $f_j^k(s)$ are shown in Table 1. For each item $j \in N$, the values corresponding to $k > j$ (which can be neglected) are denoted by "-". In order to retrieve the optimal solution value, it is sufficient to examine the values of $f_j(s)^k$ for $j = 4$ (the last item) and $s = \Phi = 5$ for all the values of k. In particular, we have a solution of value $d = \frac{(0.5-0.22)^2}{4-1} \approx 0.026$ for $k = 1$ and one of value $d = \frac{(0.25+0.24-0.22)^2}{4-2} = 0.036$ for $k = 2$. The optimal solution value, obtained for $k = 2$, corresponds to the solution $x^* = (1, 0, 1, 0)$, which encodes the cover $C^* := \{j \in N : x_j^* = 0\} = \{2, 4\}$.

6 The point-to-hyperplane distance knapsack problem

In this section, we extend our study by introducing a problem related to the (MD-SP) which arises when maximizing a hyperbolic function over the feasible region of the knapsack problem, and show that the DP algorithm presented in Section 5 can be easily adapted to also solve such a problem.

Given a set of items $S \subseteq N$, let y be the *characteristic vector* of S. For any $\ell \in \mathbb{N} \cup \{\infty\}$, we denote by $||y||_\ell$ the *ℓ-norm* of y, defined as $||y||_\ell = \sqrt[\ell]{\sum_{j=1}^n |y_j|^\ell} = \sqrt[\ell]{\sum_{j=1}^n y_j}$, if $\ell \in \mathbb{N}$, and $||y||_\ell = \lim_{\ell \to \infty} \sqrt[\ell]{\sum_{j=1}^n |y_j|^\ell} = \max_{j \in 1,\dots,n} |y_j|$, if $\ell = \infty$. We introduce the following problem:

Point-to-Hyperplane Distance Knapsack Problem. Given a set N of n items, a weight $w_j \in \mathbb{Z}_+$, a profit $p_j \in \mathbb{Z}_+$ for each item $j \in N$, a knapsack capacity $c \in \mathbb{Z}_+$, a constant $\alpha \in \mathbb{Z}_-$, and a (non-necessarily finite) positive integer $\ell \in \mathbb{N} \cup \{\infty\}$, find a subset of items $S \subseteq N$ of total weight $w(S) := \sum_{j \in S} w_j$ no larger than c and of nonnegative total profit $p(S) := \sum_{j \in S} p_j + \alpha$ such that ratio between $p(S)$ and $||y||_\ell$ is maximized. [2]

The problem calls for the maximization of the opposite of the *signed* ℓ-norm point-to-hyperplane distance between the point p (i.e., the profit vector) and the hyperplane $H_S := \big\{y \in \mathbb{R}^n : \sum_{j \in S} y_j + \alpha = 0\big\}$. Such a distance reads

$$\frac{\sum_{j \in N} p_j y_j + \alpha}{||y||_{\ell'}},$$

where ℓ and ℓ' satisfy $\frac{1}{\ell} + \frac{1}{\ell'} = 1$ (the corresponding norms are *dual* in the sense of Hölder's inequality). Figure 3 provides a geometric insight of such a signed distance.

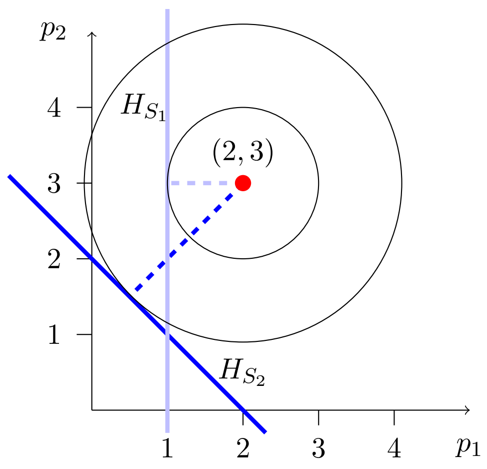

Fig. 3 An example of an instance of the Point-to-Hyperplane Distance Knapsack Problem represented in the p-space, with $|N| = 2$, $(p_1, p_2) = (2, 3)$, $\alpha = -1$ and $\ell = 3$. The example consider two possible solutions, namely $S_1 = \{1\}$, corresponding to the hyperplane $p_1 = 1$ (H_{S_1}), and $S_2 = \{1, 2\}$, corresponding to the hyperplane $p_1 + p_2 = 1$ (H_{S_2}). The dashed lines correspond to the segment connecting p to the closest point on each of the two hyperplanes, i.e., $d_2(p, H_{S_1})$ and $d_2(p, H_{S_2})$. The 3-norm length of the segments corresponds to the objective function value of the corresponding PHD-KP$_\ell$ solution: 1 for S_1 and $2^{\frac{8}{3}}$ for S_2.

By using an approach similar to the one that led to definition of the (MD-SP) in Section 3, we directly cast the Point-to-Hyperplane Distance Knapsack Problem as the following MINLP:

$$\max_{y \in \{0,1\}^n} \left\{ \frac{\left(\sum_{j \in N} p_j y_j + \alpha\right)^\ell}{\sum_{j \in N} y_j} : \sum_{j \in N} w_j\, y_j \leq c, \quad \sum_{j \in N} p_j y_j + \alpha \geq 0 \right\}. \qquad \text{(PHD-KP}_\ell\text{)}$$

An illustration of the objective function of the (PHD-KP$_\ell$) is given in Figure 4. The following proposition characterizes the computational complexity of the problem.

Proposition 3 *An optimal solution to the* (PHD-KP$_\ell$) *can be computed in* $O(n^2\, c)$ *and the problem is weakly* $\mathcal{NP}$*-hard.*

Proof The statement follows by observing that the (MD-SP) reduces to the (PHD-KP$_\ell$). Specifically, by setting $p_j = q_j$ for all $j \in N$, $\alpha = \Psi$, $c = \Phi$, and $\ell = 2$, and by observing that $\sum_{j \in N} y_j > 0$, it holds that, if the optimal solution value of the (PHD-KP$_\ell$) is strictly positive, then the value d of an optimal solution to the (MD-SP) is strictly positive as well, leading to a violated cover. Alternatively, if the (PHD-KP$_\ell$) is infeasible, then no cover exists for the (MD-SP). Now, observe

[2] Notice that, if $\alpha \leq 0$, the empty solution $y = 0$ is optimal, as it achieves a value of $\frac{\alpha}{0} = \infty$.

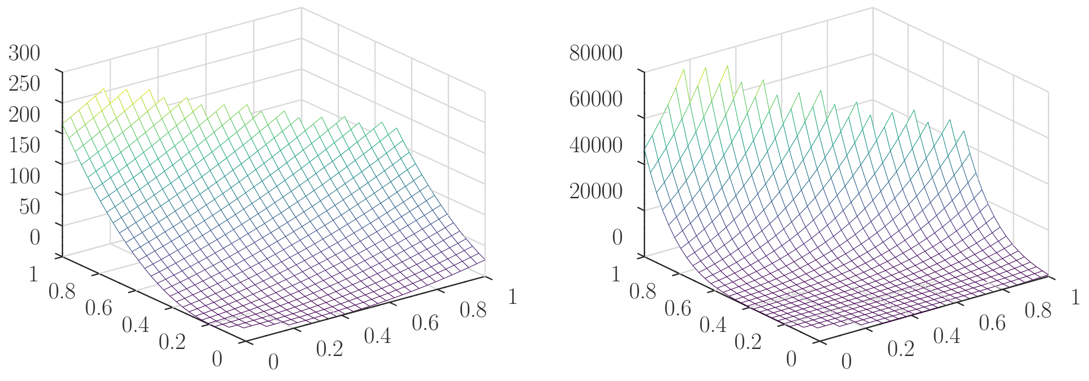

Fig. 4 An illustration of the non-concave objective function of the (PHD-KP$_\ell$) for an instance with $n = 2$, $p_1 = 4$, $p_2 = 7$, $\alpha = -1$, and $c = 9$, subject to the constraint $5\,x_1 + 8\,x_2 \leq c$, with $\ell = 3$ (left) and $\ell = 6$ (right). The objective function, which reads $\frac{(4\,y_1+7\,y_2+\alpha)^3}{y_1+y_2}$ (left) and $\frac{(4\,y_1+7\,y_2+\alpha)^6}{y_1+y_2}$ (right), is plotted only for the $(x_1, x_2) \in [0,1]^2$ pairs which are feasible, i.e., which satisfy $5\,x_1 + 8\,x_2 \leq c$ and $4\,y_1 + 7\,y_2 - 1 > 0$.

that the DP algorithm for the (MD-SP) presented in Section 5 can be easily adapted to solve the (PHD-KP$_\ell$). In particular, by denoting $g_n^k(\cdot)$ as the function $f_n^k(\cdot)$ in Section 5 in which p replaces q and α replaces Ψ, the (PHD-KP$_\ell$) can be solved in $O(n^2\,c)$. Assuming that the problem is feasible, its optimal solution value is:

$$\max_{k \in \{1,\dots,n-1\}} \left\{ \frac{g_n^k(c)^\ell}{k} : g_n^k(c) > 0 \right\}.$$

Otherwise, $g_n^k(c) \leq 0$ for all $k \in \{1, \dots, n-1\}$ and the problem is infeasible. As for Proposition 2, by storing the values of $g_j^k(s)$ for all $j \in N$ and $s \in \{0, \dots, \Phi\}$, an optimal solution can be reconstructed in linear time by backtracking. $\square$

Similarly to the DP algorithm for the (MD-SP), a speed-up technique can be applied also when solving the (PHD-KP$_\ell$). By computing $\bar{k}$ as done in Section 5 and substituting c for Φ, this leads to a complexity of $O(n\,\bar{k}\,c)$.

Two special cases of the (PHD-KP$_\ell$) are worth mentioning. First, we note that, if $\ell = 1$ and if the problem admits a solution containing at least an item, $||y||_{\ell'} = ||y||_\infty = 1$. In such a case, the (PHD-KP$_\ell$) coincides with the classical KP and, thus, it can be solved in $O(n\,c)$. We also note that, if $\ell = \infty$ and $\alpha = 0$, then $||y||_{\ell'} = ||y||_1$ coincides with the cardinality of the solution. Hence, the optimal solution to the problem can be obtained by selecting a single item of maximum profit, in which case the problem is solved in $O(n)$.

Finally, we note that the solutions to the (PHD-KP$_\ell$) (i.e., the subsets S) are in general not maximal, especially when $p_j = 0$ for some $j \in N$. In contrast, the solutions to the MD-SP are maximal by construction, even if the instance contains items with $q_j = 0$ for some $j \in N$. This is due to the fact that the denominator of the (PHD-KP$_\ell$) enforces solutions with a small cardinality, whereas the one of the (MD-SP) enforces solutions with a large cardinality. Thus, even if the (PHD-KP$_\ell$) can be solved, in principle, for the separation of the cover inequalities, the covers that one would find may not be minimal and, hence, could be dominated.

7 Experiments when optimizing over the cover-inequality closure

In this section, we investigate the impact of optimizing over the cover-inequality closure by generating maximum-depth cover inequalities within a cutting-plane method where the (MD-SP) is solved to optimality. In line with previous works such as [3, 14, 15, 38], our goal is to assess whether the exact solution of a separation problem based on the cut depth may lead to the generation of a smaller number of cuts, and to quantify how large such a reduction can be. For this reason, we adopt a pure cutting-plane method in which all the separation problems are solved to optimality.

As such, the computing times are not our primary concern in the experiment that we are about to present. Moreover, in order to assess the impact of the maximum-depth criterion in a clean setting, we generate a single cut at a time, without resorting to branching, nor to any heuristic separation method. In particular, we remark that assessing the number of cuts generated by heuristic separation methods (which are used in state-of-the-art MILP solvers such as `CPLEX`, `Gurobi`, or `Xpress`; also see the methods proposed in [24]), would be of scarce interest, as such methods do not allow for optimizing exactly over the cover-inequality closure.

We consider two main pure cutting-plane generation methods, `MaxD` and `MaxV`. The first one, `MaxD`, consists in the generation of cover inequalities of maximum depth by solving the (MD-SP) via the DP algorithm we proposed in Section 5. We recall that, due to Proposition 1, every cover corresponding to an optimal solution to the (MD-SP) is minimal. The second one, `MaxV`, is based on the generation of cover inequalities of maximum violation by solving the (MV-SP) via the standard $O(n\Phi)$ DP algorithm for the KP described in, e.g., [25, 30]. We note that, if some component $j \in N$ of the point $\bar{y}$ being separated is equal to 1 (which leads to a coefficient $q_j = 0$ in the objective function), the solution to the (MV-SP) may not be maximal, which implies that the corresponding cover may not be minimal. In particular, this is the case in most of the algorithm for solving the KP, including [25, 30, 31], where every item with $q_j = 0$ can be discarded.

For this reason, we also consider a third method, which we refer to as `MaxV+`, in which, after a cover inequality of maximum violation has been obtained by solving the (MV-SP) via the standard DP algorithm, we turn the corresponding cover into a minimal one with an *a posteriori* linear-time procedure. From a maximum-depth perspective, we note that the separation algorithm featured in `MaxV+` can be interpreted as a heuristic for solving the (MD-SP) (which still allows for optimizing over the closure exactly). Indeed, the separation algorithm in `MaxV+` computes, first, a solution which maximizes the cut violation (which corresponds to the numerator of the objective function in the (MD-SP) and, then, it removes items with $q_j = 0$ from the cover to make it minimal. As the items are removed in order of non-increasing weight, this second operation corresponds to minimizing the denominator in the (MD-SP) for a fixed value of the numerator. Overall, this leads to a solution which heuristically maximizes the cut depth.

All the exact separation algorithms that we consider are based on dynamic programming and process the items in a given pre-defined order. To limit the impact of such an order on the tie-breaking choices, in our implementations the items are always taken into account in non-increasing order of weight.

As a case study, we consider two combinatorial problems featuring one or more knapsack constraints: (i) the *Knapsack Problem* (KP) and (ii) the *Multi-dimensional Knapsack Problem* (MKP). This family of problems can be cast as the following MILP:

$$\max_{y \in \{0,1\}^n} \left\{ \sum_{j \in N} p_j \, y_j : \sum_{j \in N} w_{ij} \, y_j \leq c_i \quad i \in M \right\},$$

where y_j is a binary variable equal to 1 if and only if item $j \in N$ is chosen, p_j is the item profit, and M is a set of m knapsack constraints, with capacities c_i ($i \in M$) and item weights w_{ij} ($i \in M,\ j \in N$). The KP, as well as its numerous variants, has been extensively studied in the literature for over a century [25, 30], with works dating as early as 1897 [32]. We refer the reader to [18, 20–22, 25, 30] for more details and the KP and its variants, and to [5, 23, 24] for works focused on its polyhedral aspects. The MKP is a multi-dimensional extension of the KP featuring $m > 1$ knapsack constraints. This problem has also been extensively studied in the literature, see e.g., [25, 30, 36]. In addition, for both problems we also consider the generalization where conflicts between the items are present. Let $G = (N, E)$ be a *conflict graph* where N is the set of items and E is the set of edges representing the conflicts. To prevent the simultaneous selection of any pair of items i and j with $\{i, j\} \in E$, the following constraints are imposed:

$$y_i + y_j \leq 1 \qquad \{i, j\} \in E.$$

The generalization of the KP to the case with conflicts has been extensively studied in the literature, see, e.g., [9, 16, 35].

Table 2 Total number of cover inequalities generated for optimizing over the corresponding closure for the considered classes of instances.

				`MaxV`		`MaxD`		`MaxV+`	
conflicts	class	# const	# inst	avg	max	avg	max	avg	max
0%	*Uncorr.*	1	210	23.7	106	**14.2**	**37**	15.0	57
		5	185	20.6	43	**13.2**	**27**	13.4	28
	Weakly Corr.	1	181	22.7	63	**11.2**	**29**	12.6	31
		5	203	21.6	51	**15.0**	32	15.5	32
	Subset Sum	1	681	64.6	251	**23.1**	59	26.4	**56**
1%	*Uncorr.*	1	201	23.6	87	**13.9**	45	14.7	**42**
		5	176	20.1	43	**13.0**	**27**	13.2	28
	Weakly Corr.	1	159	22.7	66	**10.7**	**31**	12.2	36
		5	188	21.2	51	**14.5**	**32**	15.1	33
	Subset Sum	1	537	48.6	251	21.6	62	**21.5**	**50**

In line with the classes of KP instances proposed in [31], we generate knapsack items, weights, and profits considering the following three classes: *uncorrelated*, *weakly correlated*, and *subset sum.* [3] The instances depend on two user-supplied parameters: $\bar{c}$ and $\bar{R}$. For each knapsack constraint, the capacity c is defined as a percentage of the total item weight $W := \sum_{j=1}^{n} w_j$, and is set to $c := \lfloor \bar{c} \cdot W \rfloor + 1$. $\bar{R}$, which defines the order of magnitude of the item profits and weights, is set to 1,000. We generate 1 or 5 knapsack constraints for each instance, except for those of the *subset sum* class. For each KP constraint and for each class, we consider $n \in \{20, 40\}$, $\bar{c} \in \{10\%, 20\%\}$. We first generate 200 instances per combination of the parameters. In order to better assess the impact of the different cutting-plane methods, we drop any instance where the total number of cover inequalities generated by `MaxV` is smaller than 15. We notice that, in spite of this, the number of cover inequalities that are generated never exceeds a couple of hundreds. [4]

As far as the conflicts are concerned, thanks to extensive preliminary experiments we noticed that the number of generated cover inequalities decreases when increasing the density of the conflict graph. For this reason, we consider a set of conflict graphs generated by following the Erdös-Rény model with an edge probability of 1%. This way, we obtain a testbed of 2721 KP and MKP instances in which each instance is featured with or without a conflict graph.

The DP algorithms are implemented in `C++` and compiled with `gcc` v9.2.1 with flag `-O3`. The LP relaxations are solved with `CPLEX` 12.9. The experiments are run on an 8-core Intel i7-3770 @ 3.4 GHz, equipped with 16 GB of RAM, and running Ubuntu 64-bit, release 19.10. The source code of the algorithms used in the experiments can be downloaded from `https://github.com/fabiofurini/COVER_MAX_DEPTH`.

The results for the three methods `MaxD`, `MaxV`, and `MaxV+` are summarized in Table 2. Each row groups the results corresponding to $n \in \{20, 40\}$, $\bar{c} \in \{10\%, 20\%\}$, and 200 different seeds, i.e., 800 different instances. The column "conflict" reports the percentage of conflicts, the column "class" reports the class of the instances, the column "# const" reports the number of KP constraints. In the column "# inst", we report the number of instances (out of 800) in which `MaxV` generates at least 15 cover inequalities inequalities. In each row, the table reports the average and the maximum number of the cover inequalities that are generated to optimize exactly over the cover closure by the corresponding method.

Let us first compare `MaxD` to `MaxV`, thus considering the clean setting in which either the cut depth or the cut violation is considered. Our results show that `MaxV` generates, in total, 103,198 inequalities on the whole testbed (with an average of 37.92 inequalities per instance), whereas

[3] Preliminary experiments on other classes proposed in [31] revealed that, on the *strongly correlated*, *inversely correlated*, and *almost strongly correlated* classes, the number of cover inequalities that is generated is extremely small (2 or 3 cuts on average).

[4] We also tested the nine classes of MKP instances belonging to the OR Lib [8], introduced in [12]. Experimentally, we observed that the number of inequalities generated on these instances is extremely small, often smaller than 5.

MaxD only generates 47,316 (17.38 on average), for a reduction of more than a factor of 2. The maximum difference per instance in the number of generated cuts is 231, and the average one is of 20.54. The *Subset Sum* instances are the class of instances in which the number of inequalities generated by either algorithm is more substantial (more than twice the number of inequalities that are generated for the instances of the other two classes). The number of instances where MaxV generates at least 15 cover inequalities is substantially larger. We observe that MaxV generates a total of 70,081 inequalities (57.53 on average), in contrast to MaxD, which generates a total of 27,325 inequalities (22.43 on average), for a total reduction of more than a factor of 2.5. As Table 2 shows, the presence of conflict constraints leads to an overall reduction in the number of cover inequalities, which also results in a less pronounced difference between the two methods. The total number of inequalities that are generated on the instances where no conflicts are present (1460 instances) is 26,209 for MaxD and 61,248 for MaxV; for the instances with conflicts (1261 instances), 41,950 inequalities are generated by MaxV and 21,107 cover inequalities for MaxD. A similar phenomenon is observed when the number of KP constraints is increased from 1 (1969 instances) to 5 (752 instances). The total number of inequalities that are generated on the instances with a single KP constraints is 87,499 for MaxV and 36,830 for MaxD; for the instances with 5 KP constraints, 15,699 inequalities are generated by MaxV and 10,486 cover inequalities for MaxD.

When also taking the performance of MaxV+ into consideration, we observe that the latter leads to a very large reduction in the number of cover inequalities w.r.t. MaxV. Interestingly, such a reduction seems to be almost comparable, albeit slightly inferior, to the one obtained with MaxD. Indeed, MaxV+ generates, in total, 50,629 inequalities (18.6 on average), whereas MaxD generates 47,316 inequalities (17.38 on average). As we illustrated before, from a cut-depth perspective such a performance can be explained by interpreting the separation algorithm featured in MaxV+ as a heuristic for solving the separation problem in MaxD. The superior performance of MaxD and MaxV+ confirms that, as can be expected, the impact of generating minimal covers is substantial. Nevertheless, the experiments we carried out on our testbed show that, when maximizing the ratio between the numerator and the denominator of the (MD-SP) exactly, as done in MaxD, it is possible to achieve a further reduction w.r.t. MaxV+ by 7%.

As the separation algorithm embedded in MaxD runs in $O(n\,\bar{k}\,\Phi)$, its computing time is larger than the one of the classical DP algorithm used in MaxV, which runs in $O(n\,\Phi)$. While this may be compensated by the reduction in the number of cutting-plane iterations (which also leads to solving a smaller number of LP relaxations), such reduction is, while substantial on the two testbeds we considered, not sufficient to obtain a cutting plane algorithm that is also faster in terms of computing time. Clearly, one of the main driver of the computational time is the number of items n. On average on the 2721 instances we considered, the time taken by MaxD is of ≈ 0.3 seconds for $n = 20$ and of ≈ 3.8 seconds for $n = 40$. The time taken by MaxV+ is of ≈ 0.02 seconds for $n = 20$ and ≈ 0.08 for $n = 40$. The time taken by MaxV is of ≈ 0.03 seconds for $n = 20$ and ≈ 0.16 for $n = 40$. For MaxD, we have, on average, $\bar{k} \simeq 17$ for $n = 20$ and $\bar{k} \simeq 36$ for $n = 40$. This shows that, with MaxV+, we obtain not only a reduction in the number of cutting planes, but also in the computing times. The larger computing times of MaxD are in line with the fact that the (MD-SP) solved in MaxD is intrinsically harder than the MV-SP due to being a non-linear variant of the KP with a fractional objective function. We further remark that most of the time is taken by solving the separation problem, and only a fraction of it is spent by reoptimizing the LP after a cut has been added. In a different context where the reoptimization problem that is solved after a cut is added is harder (such as, e.g., a MILP or a very large LP), a reduction in the number of cutting planes could have a beneficial effect on the computing time as well.

In summary, our experiments indicate that, when optimizing over the closure of the cover inequalities, the cut depth is a valid measure whose inclusion can lead to a substantial reduction in the number of cutting planes that are generated. They also reveal that, rather than optimizing such a measure directly, which leads to a harder separation problem, it is better to take it into account indirectly in a heuristic method such as the one featured in MaxV+, whose complexity is the same as that for the standard method in MaxV.

8 Computational experiments for solving the (PHD-KP$_\ell$)

Table 3 Computational comparison between the DP algorithm of Section 6 and `BARON` in terms of the average solution time in seconds (t[s]) and the number of instances solved to optimality (#opt) for different values of $n, \bar{c}$, and ℓ withing a time limit of 600 seconds.

			DP algorithm						`BARON`					
			$\ell = 2$		$\ell = 3$		$\ell = 10$		$\ell = 2$		$\ell = 3$		$\ell = 10$	
n	$\bar{c}$	#inst	t[s]	#opt	t[s]	#opt	t[s]	#opt	t[s]	#opt	t[s]	#opt	t[s]	#opt
75	25	8	0.1	**8**	0.1	**8**	0.1	**8**	0.3	6	0.2	6	0.2	6
	50	8	0.2	**8**	0.2	**8**	0.2	**8**	0.2	6	0.2	6	0.8	6
	75	8	0.4	**8**	0.4	**8**	0.4	**8**	15.9	**8**	16.2	**8**	27.0	**8**
100	25	8	0.1	**8**	0.1	**8**	0.1	**8**	0.9	**8**	1.0	**8**	0.9	**8**
	50	8	0.5	**8**	0.5	**8**	0.5	**8**	0.4	**8**	0.4	**8**	0.8	**8**
	75	8	0.8	**8**	0.8	**8**	0.8	**8**	0.1	6	0.2	6	0.2	6
150	25	8	0.6	**8**	0.6	**8**	0.6	**8**	0.5	6	0.8	6	0.4	6
	50	8	1.8	**8**	1.7	**8**	1.8	**8**	0.4	6	0.4	6	0.7	6
	75	8	3.5	**8**	3.5	**8**	3.5	**8**	0.3	6	0.4	6	0.6	6
Avg/tot			0.9	**72**	0.9	**72**	0.9	**72**	2.1	60	2.2	60	3.5	60

With this last set of experiments, we focus on the (PHD-KP$_\ell$), comparing the performance of the DP algorithm we presented in Section 6 to `BARON` v19.12.7, which is one of the state-of-the-art spatial branch-and-bound solvers for global optimization.

Preliminary experiments by running `BARON` on the formulation (PHD-KP$_\ell$) reported in Section 6 revealed that the solved failed to solve most of the instances, especially for larger values of ℓ. This is due to the solver incurring in numerical issues caused by the nature of the numerator, which consists in a high-degree polynomial function. For this reason, the experiments in this section are carried out by using the following formulation:

$$\max_{y \in \{0,1\}^n} \left\{ \frac{\sum_{j \in N} p_j y_j + \alpha}{\sqrt[\ell]{\sum_{j \in N} y_j}} : \quad \sum_{j \in N} w_j \, y_j \leq c, \quad \sum_{j \in N} p_j y_j + \alpha \geq 0 \right\}.$$

We remark that, by solving this formulation with `BARON`, one does not have the guarantee that the problem is solved to infinite precision, differently from the case of our DP algorithm. For this reason, to limit as much as possible the impact of adopting a finite precision on the accuracy of the solution to the formulation reported above, we run `BARON` by setting the optimality tolerance to 0.

For these experiments, we generate the item profits p and weights w as proposed in [31] according to the following 8 different classes: *uncorrelated, weakly, strongly, inverse-strongly, and almost-strongly correlated, subset sum, even-odd subset sum*, and *even-odd strongly correlated.* We consider $n \in \{75, 100, 150\}$, $\bar{c} \in \{25\%, 50\%, 75\%\}$, which determines the knapsack capacity $c := \lfloor \bar{c} \cdot W \rfloor + 1$, $\bar{R} = 1000$, and $\alpha := -1$. We also consider $\ell \in \{2, 3, 10\}$, leading us to a total of 216 (PHD-KP$_\ell$) instances. We adopt a time limit of 600 seconds for each instance.

The results are reported in Table 3. Each row reports the average computing time in seconds for the instances of the 8 classes generated with the same values of n and $\bar{c}$ (t[s]) and the number of instances solved to optimality (#opt). The column # reports the number of instances per row. The former is computed by only considering the instances that have been solved to optimality by the associated method.

The table shows that the DP algorithm is able to solve all the 216 instances considered while `BARON` only solves 180 of them, i.e., $\approx 83\%$. It also shows that the performance of the DP algorithm is not affected at all by the value of ℓ, while the performance of `BARON` tends to worsen for high values of ℓ (see, in particular, the instances with $n = 75$ and $\bar{c} = 50, 75$). The performance of both methods degrades when increasing the number of items, but the impact of n on the performance of the DP algorithm is much smaller than on `BARON`, showing that the DP algorithm enjoys better scaling properties. In particular, for $n = 150$ and all the values of ℓ and $\bar{c}$, `BARON` always fails to solve two instances. Across the whole testbed, the instances BARON fails to solve are always of the categories *even-odd subset sum* and *even-odd strongly correlated.* The table also shows that the knapsack capacity has a negative impact on the performance of the DP algorithm, in line with its pseudopolynomial complexity $O(n\,\bar{k}\,c)$.

9 Conclusions

This work constitutes a first step on the exact separation of maximum-depth inequalities, focusing on the family of cover inequalities. We have proposed a pseudopolynomial-time Dynamic Programming (DP) algorithm for the maximum-depth separation of such inequalities, which has allowed us to established that the problem is weakly $\mathcal{NP}$-hard. Our experiments have indicated that a maximum-depth approach can lead to a reduction in the number of cuts that are generated when optimizing over the cover-inequality closure using a pure cutting-plane algorithm. Since, however, the computational complexity of the separation problem increases, this has not lead to an overall reduction of the computing times in our experiments. Interestingly, we have observed that a similar reduction in the number of cuts can be obtained via the separation of maximally-violated inequalities whose covers are then made minimal via an *a posteriori* procedure (which leads to a reduction of the overall computing time).

We have then introduced the Point-to-Hyperplane Distance Knapsack Problem (PHD-KP), which is a variant of the Knapsack Problem with a hyperbolic objective function similar to the one featured in the separation problem of maximum-depth cover inequalities. We have shown how such a problem can be solved to optimality via an extension of the DP algorithm we proposed for the separation problem. For the PHD-KP, our experiments have revealed that our DP algorithm is the most-effective approach when compared to a state-of-the-art spatial branch-and-bound solver in terms of both the computing time and the number of instances solved.

With this letter, we hope to stimulate the research of algorithms for the separation of other families of valid inequalities, such as, e.g., clique constraints or 0-$\frac{1}{2}$ Gomory cuts [4, 27], adopting the maximum-depth criterion. Such a research may also include the adoption of this criterion in the context of *Column Generation* methods such as those for solving the Set-Partitioning formulation of the Bin Packing Problem, see, e.g., [15, 19], by generating columns corresponding to violated dual constraints of maximum depth, whose pricing problem coincides with the PHD-KP.

Acknowledgments

The authors are grateful to two anonymous referees and to the associate editor for their comments, which helped to improve the quality of the paper. The first author acknowledges support from the Université Catholique de Louvain via the “Fonds Spéciaux de Recherche” (FSR) 2017-2021, and the Fondation Louvain via the research grant COALESCENS.

References

1. T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.
2. E. Amaldi and S. Coniglio. A distance-based point-reassignment heuristic for the k-hyperplane clustering problem. *European Journal of Operational Research*, 227(1):22–29, 2013.
3. E. Amaldi, S. Coniglio, and S. Gualandi. Coordinated cutting plane generation via multi-objective separation. *Mathematical Programming*, 143(1-2):87–110, 2014.
4. G. Andreello, A. Caprara, and M. Fischetti. Embedding {0, 1/2}-cuts in a branch-and-cut framework: A computational study. *INFORMS Journal on Computing*, 19(2):229–238, 2007.
5. P. Avella, M. Boccia, and I. Vasilyev. A computational study of exact knapsack separation for the generalized assignment problem. *Computational Optimization and Applications*, 45(3): 543–555, 2010.
6. E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58(1-3):295–324, 1993.

7. E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9):1229–1246, 1996.
8. J.E. Beasley. Or-library: Distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.
9. A. Bettinelli, V. Cacchiani, and E. Malaguti. A branch-and-bound algorithm for the knapsack problem with conflict graph. *INFORMS Journal on Computing*, 29(3):457–473, 2017.
10. P. Bradely and O. Mangasarian. k-plane clustering. *Journal of Global Optimization*, 16:23–32, 2000.
11. S. Chopra and M. R. Rao. The Steiner tree problem II: Properties and classes of facets. *Mathematical Programming*, 64(1-3):231–246, 1994.
12. P.C. Chu and J.E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics*, 4(1):63–86, 1998.
13. S. Coniglio. The impact of the norm on the k-hyperplane clustering problem: Relaxations, restrictions, approximation factors, and exact formulations. In *Proceedings of the 10th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW), Frascati, Italy*, pages 118–121, 2011.
14. S. Coniglio and M. Tieves. On the generation of cutting planes which maximize the bound improvement. In *International Symposium on Experimental Algorithms*, pages 97–109. Springer, 2015.
15. S. Coniglio, F. D'Andreagiovanni, and F. Furini. A lexicographic pricer for the fractional bin packing problem. *Operations Research Letters*, 47:622–628, 2019.
16. S. Coniglio, F. Furini, and P. San Segundo. A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *European Journal of Operational Research*, 289(2): 435–455, 2020.
17. H. Crowder, E. L Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.
18. C. D'Ambrosio, F. Furini, M. Monaci, and E. Traversi. On the product knapsack problem. *Optimization Letters*, 12(4):691–712, 2018.
19. Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255 (1):1–20, 2016.
20. F. Furini, M. Iori, S. Martello, and M. Yagiura. Heuristic and exact algorithms for the interval min–max regret knapsack problem. *INFORMS Journal on Computing*, 27(2):392–405, 2015.
21. F. Furini, I. Ljubić, and M. Sinnl. An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem. *European Journal of Operational Research*, 262(2):438–448, 2017.
22. F. Furini, M. Monaci, and E. Traversi. Exact approaches for the knapsack problem with setups. *Computers & Operations Research*, 90:208–220, 2018.
23. C. Hojny, T. Gally, O. Habeck, H. Lüthen, F. Matter, M.E. Pfetsch, and A. Schmitt. Knapsack polytopes: A survey. *Annals of Operations Research*, pages 1–49, 2019.
24. K. Kaparis and A.N. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical programming*, 124(1-2):69–91, 2010.
25. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, Berlin Heidelberg, 2004.
26. D. Klabjan, C. A. Tovey, and G. L. Nemhauser. The complexity of cover inequality separation. *Operations Research Letters*, 23(1-2):35–40, 1998.
27. Arie MCA Koster, Adrian Zymolka, and Manuel Kutschka. Algorithms to separate $\{0, \frac{1}{2}\}$-chvátal-gomory cuts. *Algorithmica*, 55(2):375–391, 2009.
28. M. Lübbecke. *Engine Scheduling by Column Generation.* PhD thesis, Technische Universität Braunschweig, 2001.
29. O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24(1-2): 15–23, 1999.
30. S. Martello and P. Toth. *Knapsack problems: Algorithms and computer implementations.* John Wiley & Sons, New York, 1990.
31. S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.

32. G. B. Mathews. On the partition of numbers. *Proceedings of the London Mathematical Society*, s1-28(1):486–490, 11 1896.
33. G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization.* Wiley-Interscience, New York, 1999.
34. N. Papadakos. Integrated airline scheduling. *Computers & Operations Research*, 36(1):176–195, 2009.
35. U. Pferschy and J. Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13(2):233–249, 2009.
36. J. Puchinger, G.R. Raidl, and U. Pferschy. The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, 22(2):250–265, 2010.
37. F. Vanderbeck. *Decomposition and Column Generation for Integer Program.* PhD thesis, Faculty of Applied Sciences, Université Catholique de Louvain, 1994.
38. A. Zanette, M. Fischetti, and E. Balas. Lexicography and degeneracy: Can a pure cutting plane algorithm work? *Mathematical Programming*, 130(1):153–176, 2011.