Mathematical Programming Computation
https://doi.org/10.1007/s12532-023-00241-9

FULL LENGTH PAPER

Efficient MIP techniques for computing the relaxation complexity

Gennadiy Averkov[1] · Christopher Hojny[2] · Matthias Schymura[1]

Received: 10 March 2022 / Accepted: 20 March 2023

Abstract
The relaxation complexity $\mathrm{rc}(X)$ of the set of integer points X contained in a polyhedron is the minimal number of inequalities needed to formulate a linear optimization problem over X without using auxiliary variables. Besides its relevance in integer programming, this concept has interpretations in aspects of social choice, symmetric cryptanalysis, and machine learning. We employ efficient mixed-integer programming techniques to compute a robust and numerically more practical variant of the relaxation complexity. Our proposed models require row or column generation techniques and can be enhanced by symmetry handling and suitable propagation algorithms. Theoretically, we compare the quality of our models in terms of their LP relaxation values. The performance of those models is investigated on a broad test set and is underlined by their ability to solve challenging instances that could not be solved previously.

Keywords Mixed-integer programming models · Relaxation complexity · Branch-and-price · Branch-and-cut

Mathematics Subject Classification 90C10 · 90C11

✉ Christopher Hojny
c.hojny@tue.nl

Gennadiy Averkov
averkov@b-tu.de

Matthias Schymura
schymura@b-tu.de

1 BTU Cottbus-Senftenberg, Platz der Deutschen Einheit 1, 03046 Cottbus, Germany

2 Combinatorial Optimization Group, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

1 Introduction

Let $X \subseteq \mathbb{Z}^d$ be such that $X = \operatorname{conv}(X) \cap \mathbb{Z}^d$ and let $Y \subseteq \mathbb{Z}^d$. A fundamental problem in various fields is to find a polyhedron P with the minimum number of facets such that $X \subseteq P$ and $(Y \setminus X) \cap P = \emptyset$. We call this quantity the *relaxation complexity of* X *w.r.t.* Y, in formulae, $\operatorname{rc}(X, Y)$, and we write $\operatorname{rc}(X)$ instead of $\operatorname{rc}(X, \mathbb{Z}^d)$ in the case $Y = \mathbb{Z}^d$. Any polyhedron $Q \subseteq \mathbb{R}^d$ with the property that $X \subseteq Q$ and $(Y \setminus X) \cap Q = \emptyset$, is called a *relaxation of* X *w.r.t.* Y. In the theory of social choice, $X \subseteq \{0, 1\}^d$ can be interpreted as the winning strategies of a simple game, see [29, Chap. 8.3]. One is then interested in computing $\operatorname{rc}(X, \{0, 1\}^d)$, i.e., the smallest number of inequalities needed to distinguish winning and loosing strategies. In symmetric cryptanalysis, a subfield of cryptography, $\operatorname{rc}(X, \{0, 1\}^d)$ corresponds to the minimum number of substitutions in symmetric key algorithms [28]. In machine learning, relaxations P correspond to polyhedral classifiers that distinguish two types of data points [2]. The relaxation complexity is then the minimum size of a polyhedral classifier. Finally, of course, $\operatorname{rc}(X)$ is the minimum number of inequalities needed to formulate a linear optimization problem over $X \subseteq \mathbb{Z}^d$ without using auxiliary variables.

Depending on the application, different strategies have been pursued to compute and bound the relaxation complexity. For example, Kaibel and Weltge [22] introduced the notion of hiding sets to derive lower bounds on $\operatorname{rc}(X)$. Using this technique, they could show that several sets X arising from combinatorial optimization problems have superpolynomial relaxation complexity. Moreover, $\operatorname{rc}(X, Y)$ can be found by computing the chromatic number of a suitably defined hypergraph; deriving lower bounds on the chromatic number allowed Kurz and Napel [23] to find a lower bound on $\operatorname{rc}(X, \{0, 1\}^d)$ in the context of social choice. In machine learning, algorithms have been devised to construct polyhedral classifiers and thus providing upper bounds on $\operatorname{rc}(X, Y)$, see [2, 9, 24, 25]. To find the exact value of $\operatorname{rc}(X, \{0, 1\}^d)$ in the context of symmetric cryptanalysis, mixed-integer programming models have been investigated. For higher dimensions, however, many of these models cannot compute $\operatorname{rc}(X, \{0, 1\}^d)$ efficiently in practice.

In this article, we follow the latter line of research. Given the relevance of knowing the exact value of $\operatorname{rc}(X, Y)$, our aim is to develop efficient mixed-integer programming (MIP) techniques that allow to compute $\operatorname{rc}(X, Y)$, if both X and Y are finite. More precisely, we investigate methods to compute $\operatorname{rc}_\varepsilon(X, Y)$, a more robust variant of $\operatorname{rc}(X, Y)$ that is numerically more practical as we discuss below. To this end, we propose in Sect. 2 three different MIP models that allow to compute $\operatorname{rc}_\varepsilon(X, Y)$: a compact model as well as two more sophisticated models that require row or column generation techniques. Section 3 compares the quality of the three models in terms of their LP relaxation value, and we discuss several enhancements of the basic models in Sect. 4. These enhancements include tailored symmetry handling and propagation techniques as well as cutting planes. Finally, we compare the performance of the three different models on a broad test set comprised of instances with different geometric properties and instances arising in symmetric cryptanalysis (Sect. 5). Our novel methods allow to solve many challenging instances efficiently, which was not possible using the basic models.

We remark that the basic versions of two models have already been used by us in [3] to find $\mathrm{rc}_\varepsilon(X, Y)$ for X being the integer points in low-dimensional cubes and crosspolytopes. These experiments helped us to prove general formulae for $\mathrm{rc}(X)$ in these cases. For this reason, we believe that the more sophisticated algorithms described in this article are not only of relevance for practical applications, but also to develop hypotheses for theoretical results. Our code is publicly available at github[1] and zenodo [14].

Regarding the computation of $\mathrm{rc}(X, Y)$ and $\mathrm{rc}_\varepsilon(X, Y)$ for $Y = \mathbb{Z}^d$ and other infinite sets Y the following issues have to be taken into account: In [3, Sect. 3] examples X are given, for which $\mathrm{rc}(X, Y)$ does not approach $\mathrm{rc}(X, \mathbb{Z}^d)$, when Y runs over a sequence of finite sets covering $\mathbb{Z}^d$. Consequently, for general choices of X, there is no algorithm to compute $\mathrm{rc}(X, \mathbb{Z}^d)$ by iterative computations of $\mathrm{rc}(X, Y)$ for growing sets Y. The situation is somewhat better regarding the computation of $\mathrm{rc}_\varepsilon(X, \mathbb{Z}^d)$, for $\varepsilon > 0$. In [3, Lem. 3] it was shown that $\mathrm{rc}_\varepsilon(X, \mathbb{Z}^d) = \mathrm{rc}_\varepsilon(X, Y_\varepsilon)$, for some *finite* set Y_ε that can be algorithmically determined for any given $\varepsilon > 0$ and X. Still, if $\varepsilon > 0$ is small and the underlying MIP model is solved in floating-point numbers, there is no guarantee for the correctness of the value computed due to inexactness of the arithmetic operations in floating-point numbers.

Related Literature One of the earliest references on the relaxation complexity goes back to Jeroslow [18] who showed the tight bound $\mathrm{rc}(X, \{0, 1\}^d) \leq 2^{d-1}$, for any $X \subseteq \{0, 1\}^d$. This result has been complemented by Weltge [32] who showed that most $X \subseteq \{0, 1\}^d$ have $\mathrm{rc}(X, \{0, 1\}^d) \geq \frac{2^d}{c \cdot d^3}$, for some absolute constant $c > 0$. Moreover, hiding sets proposed by Kaibel and Weltge [22] provide a lower bound on $\mathrm{rc}(X)$. The bound given by hiding sets can be improved by computing the chromatic number of a graph derived from hiding sets, see [3]. Regarding the computability of $\mathrm{rc}(X)$, it has been shown in [4] that there exists a proper subset $\mathrm{Obs}(X)$ of $\mathbb{Z}^d \setminus X$ such that $\mathrm{rc}(X) = \mathrm{rc}(X, \mathrm{Obs}(X))$. If $\mathrm{Obs}(X)$ is finite, they show that $\mathrm{rc}(X, \mathrm{Obs}(X))$, and thus $\mathrm{rc}(X)$, can be computed by solving a mixed-integer program. They also provide sufficient conditions on X that guarantee $\mathrm{Obs}(X)$ to be finite. Moreover, they establish that $\mathrm{rc}(X)$ is computable if $d \leq 3$; for $d = 2$, a polynomial time algorithm to compute $\mathrm{rc}(X)$ is discussed in [3]. In general, however, it is an open question whether $\mathrm{rc}(X)$ is computable.

One drawback of relaxations of X as defined above is that they might be sensitive to numerical errors. If $a^\intercal x \leq \beta$ is a facet defining inequality of a relaxation of X that separates $y \in \mathbb{Z}^d \setminus X$, then we only know $a^\intercal y > \beta$. Thus, slightly perturbing a might not separate y anymore. To take care of this, we suggested in [3] to add a safety margin $\varepsilon > 0$ to the separation condition. That is, if $a^\intercal x \leq \beta$ is a facet defining inequality of a relaxation of X with $\|a\|_\infty = 1$ that separates y, then we require $a^\intercal y \geq \beta + \varepsilon$. In this case, we say that y is *ε-separated from* X. Then, $\mathrm{rc}_\varepsilon(X)$ denotes the smallest number of facets of any relaxation of X that satisfies the safety margin condition.[2] We call such a relaxation an ε-relaxation of X. Analogously to $\mathrm{rc}(X, Y)$, we define $\mathrm{rc}_\varepsilon(X, Y)$ to be the smallest number of inequalities

[1] https://github.com/christopherhojny/relaxation_complexity.

[2] Note that the definition in [3] is different, but both concepts coincide if the value of ε is defined appropriately. We follow the definition provided here, because it simplifies the discussion in this article.

needed to ε-separate X and $Y \setminus X$. As ε-relaxations are more restrictive than relaxations, $\mathrm{rc}_\varepsilon(X) \geq \mathrm{rc}(X)$ for each $\varepsilon > 0$. In contrast to $\mathrm{rc}(X)$, however, we show in [3] that for every finite and full-dimensional $X \subseteq \mathbb{Z}^d$ there is a finite set $Y \subseteq \mathbb{Z}^d \setminus X$ such that $\mathrm{rc}_\varepsilon(X) = \mathrm{rc}_\varepsilon(X, Y)$. Thus, $\mathrm{rc}_\varepsilon(X)$ is computable and the aim of this article is to develop MIP techniques that allow to find $\mathrm{rc}_\varepsilon(X, Y)$ efficiently. In particular, if ε approaches 0, then $\mathrm{rc}_\varepsilon(X)$ converges towards $\mathrm{rc}_{\mathbb{Q}}(X)$, a variant of the relaxation complexity which requires the relaxations to be rational. Another reason to passing from $\mathrm{rc}(X)$ to $\mathrm{rc}_\varepsilon(X)$ was given recently in [1]: For dimensions $d \geq 5$, there are choices of X such that there is no rational relaxation attaining $\mathrm{rc}(X)$, that is, any relaxation with the minimal number of facets needs irrational data to be represented. Further variations of $\mathrm{rc}(X)$ in which the size of coefficients in facet defining inequalities are bounded are discussed in [12, 13].

Besides finding relaxations of X, another field of research aims to find outer descriptions of $P = \mathrm{conv}(X)$ to be able to use linear programming techniques to solve optimization problems over X. Since P might have exponentially many facets, the concept of extended formulations has been introduced. Extended formulations are polyhedra $Q \subseteq \mathbb{R}^{d+k}$ whose projection onto $\mathbb{R}^d$ yields P. The smallest number of facets of an extended formulation of P is its extension complexity $\mathrm{xc}(P)$. We refer the reader to the surveys of Conforti et al. [8] and Kaibel [21] as well as the references therein. Extended formulations that allow to use integer variables have been discussed, e.g., by Bader et al. [5], Cevallos et al. [7], and Weltge [32, Chap. 7.1]. A combination of $\mathrm{rc}(X, \{0, 1\}^d)$ and $\mathrm{xc}(\mathrm{conv}(X))$ has been studied by Hrubeš and Talebanfard [17].

Basic Definitions and Notation Throughout this article, we assume that d is a positive integer. The set $\{1, \dots, d\}$ is denoted by $[d]$, and we write $e_1, \dots, e_d$ to denote the d canonical unit vectors in $\mathbb{R}^d$. Moreover, $\Delta_d = \{0, e_1, \dots, e_d\} \subseteq \mathbb{R}^d$ is the vertex set of the standard simplex in $\mathbb{R}^d$, and $\Diamond_d = \{0, \pm e_1, \dots, \pm e_d\} \subseteq \mathbb{R}^d$ denotes the integer points in the d-dimensional standard crosspolytope. The affine hull of a set $X \subseteq \mathbb{R}^d$ is denoted by $\mathrm{aff}(X)$.

A set $X \subseteq \mathbb{Z}^d$ is called *lattice-convex* if $X = \mathrm{conv}(X) \cap \mathbb{Z}^d$. For a lattice-convex set $X \subseteq \mathbb{Z}^d$, we say that $H \subseteq (\mathrm{aff}(X) \cap \mathbb{Z}^d) \setminus X$ is a *hiding set* if, for any distinct $y_1, y_2 \in H$, we have $\mathrm{conv}(\{y_1, y_2\}) \cap \mathrm{conv}(X) \neq \emptyset$. Kaibel and Weltge [22] proved that the cardinality of any hiding set is a lower bound on $\mathrm{rc}(X)$. The maximum size of a hiding set is denoted by $H(X)$. Moreover, if $Y \subseteq \mathbb{Z}^d$, we say that H is a *Y-hiding set* if H is a hiding set that is contained in Y. Analogously to $H(X)$, $H(X, Y)$ denotes the maximum size of a Y-hiding set.

2 Mixed-integer programming models to compute $\mathrm{rc}_\varepsilon(X, Y)$

In this section, we discuss three different mixed-integer programming models to compute $\mathrm{rc}_\varepsilon(X, Y)$. The three different MIP formulations that we discuss differ in the way how they model $\mathrm{rc}_\varepsilon(X, Y)$. The first model uses only polynomially many variables and inequalities, the second model needs exponentially many inequalities while the number of variables is still polynomial, and the third model requires exponentially many variables but only polynomially many inequalities. For this reason, we refer to these three models as the compact, the cutting plane, and the column generation model, respectively. In preliminary experiments with our code, we have already used the compact and column generation model [3]. Nevertheless, we provide the full details of these models to make the article self-contained and to be able to explain the model enhancements. For the sake of convenience, we assume for the remainder of this article that X and Y are disjoint. This is without loss of generality, because we can replace Y by $Y \setminus X$, which does not change the value of $\mathrm{rc}_\varepsilon(X, Y)$. We also refer to X as the set of *feasible* points, whereas the points in Y are called *infeasible*.

2.1 Compact model

Observe that lattice-convex sets are exactly those subsets of $\mathbb{Z}^d$ that admit a relaxation. In [4], a mixed-integer programming formulation has been proposed to check whether a finite lattice-convex set X admits a relaxation with k inequalities, and we have explained in [3] how to adapt the model to be able to compute $\mathrm{rc}_\varepsilon(X, Y)$.

Given an upper bound k on the number of inequalities needed to separate X and Y, the model's idea is to introduce variables a_{ij} and b_i, $(i, j) \in [k] \times [d]$, to model the k potential inequalities needed in a relaxation. Moreover, for each $y \in Y$ and $i \in [k]$, a binary variable s_{yi} is introduced that indicates whether the ith inequality is violated by y; additional binary variables u_i, $i \in [k]$, indicate whether the ith inequality is needed in a relaxation. Using a big-M term with $M \geq d(\rho_X + \rho_Y) + \varepsilon$, with $\rho_X = \max\{\|x\|_\infty : x \in X\}$ and $\rho_Y = \max\{\|y\|_\infty : y \in Y\}$, the mixed-integer programming formulation for $\mathrm{rc}_\varepsilon(X, Y)$ is as follows:

$$\min \sum_{i=1}^{k} u_i \tag{1a}$$

$$\sum_{j=1}^{d} a_{ij} x_j \leq b_i, \qquad x \in X,\ i \in [k], \tag{1b}$$

$$\sum_{i=1}^{k} s_{yi} \geq 1, \qquad y \in Y, \tag{1c}$$

$$\sum_{j=1}^{d} a_{ij} y_j \geq b_i + \varepsilon - M(1 - s_{yi}), \qquad y \in Y,\ i \in [k], \tag{1d}$$

$$s_{yi} \leq u_i, \qquad y \in Y,\ i \in [k], \tag{1e}$$

$$-1 \leq a_{ij} \leq 1, \qquad (i, j) \in [k] \times [d], \tag{1f}$$

$$-d\rho_X \leq b_i \leq d\rho_X, \qquad i \in [k], \tag{1g}$$

$$s_{yi},\ u_i \in \{0, 1\}, \qquad y \in Y,\ i \in [k]. \tag{1h}$$

Inequalities (1b) ensure that the k inequalities are valid for X and Inequalities (1c) guarantee that each $y \in Y$ is cut off by at least one inequality. If an inequality is selected to separate $y \in Y$ and X, Inequalities (1d) ensure that this is consistent with the k inequalities defined by the model. Finally, Inequalities (1e) ensure that u_i is 1 if inequality $i \in [k]$ separates an infeasible point, whereas Inequalities (1f) and (1g) scale the k inequalities without loss of generality. For details on correctness, we refer the reader to [4, Sect. 4.2].

2.2 Cutting plane model

To be able to find $\mathrm{rc}_\varepsilon(X, Y)$, Model (1) introduces two classes of variables: variables u and s model which inequalities are used and subsets of Y that are separated by the selected inequalities, respectively, whereas variables a and b guarantee that the subsets defined by s can be cut by valid inequalities for X. The problem of computing $\mathrm{rc}_\varepsilon(X, Y)$ can thus be interpreted as a two stage problem, where the first stage selects a set of subsets of Y and the second stage checks whether the selected subsets correspond to feasible cut patterns. Since the first stage variables are binary and the second stage problem is a feasibility problem, logic-based Benders decomposition can be used to compute $\mathrm{rc}_\varepsilon(X, Y)$, see [16]. While classical Benders decomposition requires the subproblem to be a linear programming problem, logic-based Benders decomposition allows the subproblem to be an arbitrary optimization problem.

Let $\mathcal{C} = \{C \subseteq Y : C$ and X are not linearly ε -separable$\}$. We refer to $\mathcal{C}$ as the *conflict set*. For all $(C, i) \in \mathcal{C} \times [k]$, the *conflict inequality* $\sum_{y \in C} s_{yi} \leq |C| - 1$ models that not all points in C can be cut by an inequality valid for X. Consequently,

$$\min \sum_{i=1}^{k} u_i \tag{2a}$$

$$\sum_{i=1}^{k} s_{yi} \geq 1, \qquad y \in Y, \tag{2b}$$

$$\sum_{y \in C} s_{yi} \leq |C| - 1, \qquad C \in \mathcal{C},\ i \in [k], \tag{2c}$$

$$s_{yi} \leq u_i, \qquad y \in Y,\ i \in [k], \tag{2d}$$

$$s_{yi},\ u_i \in \{0, 1\}, \qquad y \in Y,\ i \in [k]. \tag{2e}$$

is an alternative model for computing $\mathrm{rc}_\varepsilon(X, Y)$.

2.3 Column generation model

Let $\mathcal{I} = \{I \subseteq Y : I$ and X are linearlyε-separable$\}$. Then, $\mathrm{rc}_\varepsilon(X, Y)$ is the smallest number ℓ of sets $I_1, \dots, I_\ell \in \mathcal{I}$ such that $Y = \bigcup_{i=1}^{\ell} I_i$. Thus, instead of using the matrix $s \in \{0, 1\}^{Y \times [k]}$ to encode which inequality cuts which points from Y, we can introduce for every $I \in \mathcal{I}$ a binary variable $z_I \in \{0, 1\}$ that encodes whether an inequality separates I or not:

$$\min \sum_{I \in \mathcal{I}} z_I \tag{3a}$$

$$\sum_{I \in I_y} z_I \geq 1, \qquad y \in Y, \tag{3b}$$

$$z \in \mathbb{Z}_+^{\mathcal{I}}, \tag{3c}$$

where $I_y = \{I \in \mathcal{I} : y \in I\}$.

Remark 1 In contrast to Model (1), Models (2) and (3) do not directly provide an ε-relaxation of X w.r.t. Y. To find such a relaxation, $\operatorname{rc}_\varepsilon(X, Y)$ many linear programs need to be solved in a post-processing step.

3 Comparison of basic models

While the compact model (1) can be immediately handed to an MIP solver due to the relatively small number of variables and constraints, the cutting plane model (2) and column generation model (3) require to implement separation and pricing routines, respectively. At least for the column generation model, this additional computational effort comes with the benefit of a stronger LP relaxation in comparison with the compact model. To make this precise, we denote by $v^\star_{\text{com}}$, $v^\star_{\text{cut}}$, and $v^\star_{\text{CG}}$ the optimal LP relaxation value of the compact, cutting plane, and column generation model, respectively.

Proposition 2 *Let $X \subseteq \mathbb{Z}^d$ be finite and lattice-convex, let $Y \subseteq \mathbb{Z}^d \setminus X$ be finite, let $\varepsilon > 0$ such that $\operatorname{rc}_\varepsilon(X, Y)$ exists, and suppose both Models (1) and (2) are feasible.*

1. *Then, $v^\star_{\text{com}} \geq v^\star_{\text{cut}} = 1$.*
2. *Moreover, if $\varepsilon \leq (d-1)(\rho_X + \rho_Y)$, then $v^\star_{\text{com}} = 1$.*

Note that 2 is a technical assumption that is almost always satisfied in practice, e.g., to approximate $\operatorname{rc}(X, Y)$ by $\operatorname{rc}_\varepsilon(X, Y)$, one selects $\varepsilon < 1 \leq (d-1)(\rho_X + \rho_Y)$. Thus, $v^\star_{\text{cut}} = v^\star_{\text{com}} = 1$ in all relevant cases.

Proof First we show $v^\star_{\text{cut}} \geq 1$ and $v^\star_{\text{com}} \geq 1$. Observe that we get for every (partial) feasible solution (s, u) and every $\bar{y} \in Y$ the estimation

$$\sum_{i=1}^{k} u_i \geq \sum_{i=1}^{k} \max\{s_{yi} : y \in Y\} \geq \sum_{i=1}^{k} s_{\bar{y}i} \geq 1,$$

where k is the upper bound used in Model (1) of (2). Hence, $v^\star_{\text{cut}} \geq 1$ and $v^\star_{\text{com}} \geq 1$. If the upper bound $k = 1$, we thus have necessarily $v^\star_{\text{cut}} = 1$. If $k \geq 2$, we construct a feasible solution for (2) with objective value 1 by assigning all variables value 0 except for s_{yi}, $(y, i) \in Y \times [2]$, u_1, and u_2, which get value $\frac{1}{2}$. Indeed, the left-hand side of each conflict inequality evaluates to $\frac{|C|}{2}$, while the right-hand side is $|C| - 1$. Thus, because $|C| \geq 2$ for any conflict as X is lattice-convex, we find $\frac{|C|}{2} \leq |C| - 1$, i.e., all conflict inequalities are satisfied. Since the remaining inequalities hold trivially, $1 \geq v^\star_{\text{cut}}$ follows. Consequently, $v^\star_{\text{com}} \geq 1 \geq v^\star_{\text{cut}} \geq 1$.

For the second statement, we assume $k \geq 2$, because otherwise $v^\star_{\text{com}} = 1$ follows as above. We define a feasible solution with objective value 1 of Model (1) by assigning all variables value 0 except for

- $u_1 = s_{y1} = \frac{\varepsilon}{M}$ for all $y \in Y$;
- $u_2 = s_{y2} = 1 - \frac{\varepsilon}{M}$ for all $y \in Y$;
- $a_{11} = 1$ and $b_1 = \rho_X$.

The inequalities $a_{i\cdot}^\intercal x \leq b_i$ defined this way are either $0 \leq 0$ or $x_1 \leq \rho_X$, which are valid for X. Moreover, the Inequalities (1d) are satisfied, because for $i = 1$ and every $y \in Y$, we have

$$\sum_{j=1}^{d} a_{1j} y_j - b_1 = y_1 - \rho_X \geq -\rho_Y - \rho_X \geq -d(\rho_X + \rho_Y) + \varepsilon \geq \varepsilon - M(1 - s_{y1}),$$

and for the remaining $i \geq 2$, we get $\varepsilon - M(1 - s_{y2}) = 0$. Since one can easily check that the remaining inequalities of (1) are also satisfied, $v^\star_{\text{com}} \leq 1$ follows, concluding the proof using the first part of the assertion. □

The value of the LP relaxations thus does not indicate whether the compact or cutting plane model performs better in practice. An advantage of the latter is that the conflict inequalities encode a hypergraph coloring problem, which is a structure appearing frequently in practice. Hence, there might be a chance that a solver can exploit this structure if sufficiently many inequalities have been separated. The compact model, however, might have the advantage that the a- and b-variables guide the solver in the right direction when branching on s- or u-variables, because feasibility is already encoded in the model and does not need to be added to the model by separating cutting planes.

Proposition 3 *Let $X \subseteq \mathbb{Z}^d$ be finite and lattice-convex, let $Y \subseteq \mathbb{Z}^d \setminus X$ be finite, let $\varepsilon > 0$ be such that $\operatorname{rc}_\varepsilon(X, Y)$ exists, and suppose both Models (1) and (2) are feasible. Let k be the number of inequalities encoded in Model (1).*

1. *If there exists an optimal solution of the LP relaxation of (3) that assigns at most k variables a positive value, then $v^\star_{CG} \geq v^\star_{\text{com}} \geq v^\star_{\text{cut}} = 1$.*
2. *We have $v^\star_{CG} \geq H(X, Y)$, and this can be strict.*

Proof To show $v^\star_{\text{CG}} \geq v^\star_{\text{com}}$, recall that for each $I \in \mathcal{I}$ there exists an inequality $a(I)^\intercal x \leq b(I) + \varepsilon$ separating I and X. Due to rescaling, we may assume that $a(I) \in [-1, 1]^d$ and $b(I) \in [-d\rho_X, d\rho_X]$.

If we are given a solution $z \in \mathbb{R}^{\mathcal{I}}_+$ of (3b) with at most k non-zero entries, we define a solution of the LP relaxation of (1) with the same objective value as follows. Let $I_1, \ldots, I_\ell \in \mathcal{I}$ be the indices of non-zero entries in z. For each $i \in [\ell]$ and $y \in Y$, define

$$s_{yi} = \begin{cases} z_{I_i}, & \text{if } y \in I_i, \\ 0, & \text{otherwise,} \end{cases} \qquad \text{and} \qquad u_i = z_{I_i}.$$

For $i \in \{\ell + 1, \dots, k\}$ and $y \in Y$, we define $s_{yi} = 0$ and $u_i = 0$. Finally, let $a_{ij} = a(I_i)_j$ and $b_i = b(I_i)$ for $(i, j) \in [\ell] \times [d]$. For $i \in \{\ell + 1, \dots, k\}$, define $a_{ij} = 0$ and $b_i = 1$. Indeed, this solution adheres to (1b) since (a, b) defines valid inequalities, and also (1e)–(1g) hold trivially. By definition, s and u also satisfy the box constraints corresponding to (1h). To see that (1c) holds, note that for each $y \in Y$,

$$\sum_{i=1}^{k} s_{yi} = \sum_{i \in [\ell]\colon\, y \in I_i} z_{I_i} \overset{(3b)}{\geq} 1,$$

since z is feasible for the LP relaxation of (3). For the last constraint (1d), note that the constraint is trivially satisfied if $s_{yi} = 0$. If $s_{yi} > 0$, then $a_{i\cdot}{}^{\top} x \leq b_i$ corresponds to an inequality separating X and y, which finally shows that the newly defined solution is feasible for the LP relaxation of (1). To conclude, note that $\sum_{i=1}^{k} u_i = \sum_{i=1}^{\ell} z_{I_i}$. Hence, $v^{\star}_{\mathrm{CG}} \geq v^{\star}_{\mathrm{com}}$ and the remaining estimations hold by Proposition 2.

For the second part, let $H \subseteq Y$ be a hiding set for X and let $z \in \mathbb{R}^{\mathcal{I}}_{+}$ be an optimal solution of the LP relaxation of (3). Then, for distinct $y_1, y_2 \in H$, we have $I_{y_1} \cap I_{y_2} = \emptyset$. Consequently, we can estimate

$$v^{\star}_{\mathrm{CG}} = \sum_{I \in \mathcal{I}} z_I \geq \sum_{y \in H} \sum_{I \in \mathcal{I}_y} z_I \overset{(3b)}{\geq} |H|,$$

which shows $v^{\star}_{\mathrm{CG}} \geq H(X, Y)$.

To see that the inequality can be strict, consider $X = \{0, 1\}^2$ and let Y be the set of infeasible points in $\mathbb{Z}^2$ with ℓ_∞-distance 1 from X. One can readily verify that a maximum hiding set for X has size 2, while the LP relaxation of (3) has value $\frac{8}{3}$. □

If Y contains a hiding set of size at least 2, the column generation model is thus strictly stronger than the compact and cutting plane model. In particular, the gap between $v^{\star}_{\mathrm{CG}}$ and $v^{\star}_{\mathrm{cut}}$ (and $v^{\star}_{\mathrm{com}}$) can be arbitrarily large: if $d = 2$ and $Y = \mathrm{Obs}(X)$, there is always a hiding set of size $\mathrm{rc}(X, Y) - 1$, see [3, Thm. 23]. On the contrary, at least in dimension $d = 2$, for a suitably small $\varepsilon > 0$ the gap between $v^{\star}_{\mathrm{CG}}$ and $\mathrm{rc}_{\varepsilon}(X, Y)$ is at most 1. This follows once more from [3, Thm. 23] together with the identities $\lim_{\varepsilon \to 0} \mathrm{rc}_{\varepsilon}(X, Y) = \mathrm{rc}_{\mathbb{Q}}(X, Y)$ (see [3, Thm. 4]) and $\mathrm{rc}_{\mathbb{Q}}(X, Y) = \mathrm{rc}(X, Y)$, for $X, Y \subseteq \mathbb{Z}^2$ (see [32, Sect. 7.5] or the discussion in [3, Sect. 4]). The situation in dimensions $d \geq 3$ is less clear, because a lower bound on the maximal size hiding set in terms of $\mathrm{rc}(X, Y)$, analogous to [3, Thm. 23], is not known.

4 Enhancements of basic models and algorithmic aspects

In their basic versions, the compact and cutting plane model are rather difficult to solve for a standard MIP solver, e.g., because not enough structural properties of $\mathrm{rc}_{\varepsilon}(X, Y)$ are encoded in the models that are helpful for a solver. Moreover, the cutting plane and column generation model require to solve a separation and pricing problem, respectively, to be used in practice. In this section, we discuss these aspects and suggest model improvements.

4.1 Incorporation of structural properties

In the following, we describe cutting planes, propagation algorithms, and how to handle symmetries and redundancies in the compact and cutting plane model. Moreover, we discuss heuristic approaches to the problem of computing $\mathrm{rc}_\varepsilon(X, Y)$.

Cutting Planes In both the compact and cutting plane model, variable s_{yi} encodes whether a point $y \in Y$ is separated by inequality $i \in [k]$. To strengthen the compact model and the initial LP relaxation of (2) without inequalities (2c), we can add inequalities that rule out combinations of points from Y that cannot be separated simultaneously.

For any hiding set $H \subseteq Y$, the *hiding set cut*

$$\sum_{y \in H} s_{yi} \leq 1, \qquad\qquad i \in [k]$$

encodes that each inequality $i \in [k]$ can separate at most one element from a hiding set. Although these cuts are the stronger the bigger the underlying hiding set, we add these inequalities just for hiding sets of size 2. The reason for this is that such hiding sets can be found easily by iterating over all pairs (y_1, y_2) of distinct points in Y and checking whether the line segment $\mathrm{conv}(\{y_1, y_2\})$ intersects $\mathrm{conv}(X)$ non-trivially. In our implementation, we insert the expression $\lambda y_1 + (1 - \lambda) y_2$ in each facet defining inequality of $\mathrm{conv}(X)$ to derive bounds on the parameter λ. Then, the final bounds on λ are within $[0, 1]$ if and only if $\{y_1, y_2\}$ is a hiding set.

For hiding sets of arbitrary cardinality, the task is more difficult, because there might exist exponentially many hiding sets. Thus, we are relying on a separation routine for hiding set cuts. The separation problem for hiding set cuts, however, is at least as difficult as finding a maximum hiding set for X, and the complexity of the latter is open.

Propagation Suppose we are solving the compact and cutting plane model using branch-and-bound. At each node of the branch-and-bound tree, there might exist some binary variables that are fixed to 0 or 1, e.g., by branching decisions. The aim of propagation is to find further variable fixings based on the already existing ones.

Our first propagation algorithm is based on the following observation.

Observation 4 *Suppose some s-variables have been fixed and let $i \in [k]$. Then, $F_i := \{y \in Y : s_{yi} = 1\}$ can be separated from X if and only if $F'_i := Y \cap \mathrm{conv}(F_i)$ can be separated from X.*

The *convexity propagation algorithm* computes the sets F'_i, $i \in [k]$, and fixes s_{yi} to 1 for all $y \in F'_i$. If there is $y' \in F'_i$ such that $s_{y'i}$ is already fixed to 0, then the algorithm prunes the node of the branch-and-bound tree. This is indeed a valid operation, because Inequalities (1c) and (2b) allow each point $y \in Y$ to be separated by several inequalities.

The second propagation algorithm exploits that $F_i \cap \operatorname{conv}(X)$ needs to be empty in each feasible solution. The *intersection propagation algorithm* thus iterates over all $y \in Y \setminus F_i$ and checks whether $\operatorname{conv}(F_i \cup \{y\}) \cap \operatorname{conv}(X) \neq \emptyset$. If the check evaluates positively, s_{yi} is fixed to 0.

Comparing both propagation algorithms, the convexity propagator requires to compute only a single convex hull per set F_i, whereas the intersection propagator needs to compute $O(|Y|)$ convex hulls per set F_i, which can be rather expensive. In our experiments, we will investigate whether the additional effort pays off in reducing the running time drastically. To avoid computing unnecessary convex hulls, we call both propagation algorithms in our implementation only if the branching decision at the parent node is based on a variable s_{yi}, and in this case only for this particular inequality index i and no further $i' \in [k] \setminus \{i\}$.

Symmetry Handling It is well-known that the presence of symmetries slows down MIP solvers, because symmetric solutions are found repeatedly during the solving process leading to an exploration of unnecessary parts of the search space. In a solution of the compact and cutting plane model, e.g., we can permute the inequality labels $i \in [k]$ without changing the structure of the solution. For this reason, one can enforce that only one representative solution per set of equivalent solutions is computed without losing optimal solutions.

One way of handling symmetric relabelings of inequalities is to require that the columns of the matrix $s \in \{0, 1\}^{Y \times [k]}$ are sorted lexicographically non-increasingly. To enforce sorted columns, we use a separation routine for orbisack minimal cover inequalities as suggested in [15] and the propagation algorithm orbitopal fixing by Bendotti et al. [6]. Both algorithms' running time is in $O(|Y| \cdot k)$. Moreover, sorting the columns of s implies that we can also require the u-variables to be sorted, i.e., the first $\operatorname{rc}_\varepsilon(X, Y)$ inequalities are the inequalities defining an ε-relaxation, which can be enforced by adding

$$u_i \geq u_{i+1}, \qquad\qquad i \in [k-1], \tag{4}$$

to the problem.

Besides the symmetries of relabeling inequalities, we might also be able to relabel points in Y without changing the structure of the problem. This is the case if we find a permutation π of $[d]$ such that $\pi(X) = X$ and $\pi(Y) = Y$, where for a set $T \subseteq \mathbb{R}^d$ we define $\pi(T) = \{\pi(t) : t \in T\}$ and $\pi(t) = (t_{\pi^{-1}(1)}, \ldots, t_{\pi^{-1}(d)})$. The permutation π gives rise to a permutation ϕ of Y and ψ of X, where $\phi(y) := \pi(y)$ and $\psi(x) := \pi(x)$.

Lemma 5 *Let (s, u) be a (partial) solution of Model (1) or (2) for $\operatorname{rc}_\varepsilon(X, Y)$. If there exists a permutation π of $[d]$ such that $\pi(X) = X$ and $\pi(Y) = Y$, then also (s', u) is a (partial) solution, where s' arises from s by reordering the rows of s according to ϕ.*

Proof Suppose (s, u) is a solution of Model (2). Then, (s, u) can be extended to a solution of Model (1), i.e., there exist k inequalities $\sum_{j=1}^{d} a_{ij} x_j \leq b_i$, $i \in [k]$, such that the ith inequality separates the points in $F_i = \{y \in Y : s_{yi} = 1\}$ from X. If we apply permutation π to X and Y, we do not change the structure of the problem, that is, $\sum_{j=1}^{d} a_{ij} \pi(x)_j \leq b_i$, $i \in [k]$, defines also a relaxation of X w.r.t. Y. Thus, if the original ith inequality separated point $y \in Y$, the permuted inequality separates $\phi(y)$. Consequently, if we define s' by relabeling the rows of s according to ϕ, $(\pi(a), b, s', u)$ is a solution of Model (1) and thus (s', u) is a solution of Model (2). □

If $\Pi = \{\pi \in S_d : \pi(X) = X,\ \pi(Y) = Y\}$ and Φ is the group containing all ϕ associated with the permutations $\pi \in \Pi$, Lemma 5 tells us that we can also force the rows of $s \in \{0, 1\}^{Y \times [k]}$ to be sorted lexicographically non-increasingly w.r.t. permutations from Φ. In our implementation, we compute a set Γ of generators of the group Φ and enforce for each $\gamma \in \Gamma$ that the matrix s is lexicographically not smaller than the reordering of s w.r.t. γ. We enforce this property by separating minimal cover inequalities for symresacks and a propagation algorithm, see [15]. Both run in $O(k)$ time per $\gamma \in \Gamma$.

To detect the symmetries Φ, we construct a colored bipartite graph $G = (V, E)$. The left side of the bipartition is given by $X \cup Y$ and the right side is defined as $R = \{(v, j) \in \mathbb{Z} \times [d] :$ there is $z \in X \cup Y$ with $z_j = v\}$. There is an edge between $z \in X \cup Y$ and $(v, j) \in R$ if and only if $z_j = v$. Moreover, each node gets a color uniquely determining its type: all nodes in X are colored equally with color "X", all nodes in Y are colored equally by color "Y", and node $(v, j) \in R$ is colored by color "v". Then, the restriction of every automorphism σ of G to R corresponds to a permutation in Π, and thus, restricting σ to Y is a permutation in Φ.

Note that the graph G defined above might not allow to detect symmetries if a symmetric arrangement of X and Y is translated asymmetrically. For example, if $X = t + \Delta_2$, $Y = t + ((\Delta_2 + \Diamond_2) \setminus \Delta_2)$, and $t = \binom{1}{2}$, then there is no permutation keeping X invariant. For this reason, we use in the construction of G relative coordinates. That is, for each coordinate $j \in [d]$, we compute $\mu_j = \min_{z \in X \cup Y} z_j$ and translate $X \cup Y$ by $-\mu$ before building G.

Another way of handling symmetries for the compact model (1) is to handle symmetries of the inequalities $\sum_{j=1}^{d} a_{ij} x_j \leq b_i$ defined in the model. We can reorder the inequalities $\sum_{j=1}^{d} a_{ij} x_j \leq b_i$, $i \in [k]$ that are (not) used in the relaxation, to obtain another solution with the same objective value. To handle these symmetries, we can add the inequalities

$$a_{i1} \geq a_{(i+1)1} - 2(u_i - u_{i+1}), \qquad i \in [k-1]. \tag{5}$$

Inequalities (5) sort the inequalities (not) present in a relaxation by their first coefficient. The inequalities are compatible with Inequalities (4), but not necessarily with the lexicographic ordering constraints. The latter is the case because cutting the point $y \in Y$ associated with the first row of matrix s might require a very small first coefficient in any separating inequality, whereas other points might require a very large first coefficient. In our experiments, we will investigate which symmetry handling method works best for the compact and cutting plane model.

Finally, additional redundancies in Model (1) can be handled by enforcing that $\sum_{j=1}^{d} a_{ij} x_j \leq b_i$ becomes the trivial inequality $0^\intercal x \leq d\rho_X$ if it is not used in a relaxation of X (i.e., $u_i = 0$). This removes infinitely many equivalent solutions from the search space, and can be modeled by replacing (1f) by

$$-u_i \leq a_{ij} \leq u_i, \qquad (i, j) \in [k] \times [d],$$

and the lower bound constraint in (1g) by

$$d\rho_X \leq b_i + 2d\rho_X u_i, \qquad i \in [k].$$

This method is compatible with both the lexicographic symmetry handling approach and Inequalities (5).

Heuristics Recall that both the compact and the cutting plane model require to know an upper bound k on $\mathrm{rc}_\varepsilon(X, Y)$. To find such an upper bound, we use two heuristics in our experiments, which we describe next.

The first heuristic is based on the fact that $\mathrm{conv}(X)$ is an ε-relaxation of X w.r.t. $\mathbb{Z}^d$. For this reason, for any $Y \subseteq \mathbb{Z}^d$, an upper bound on $\mathrm{rc}_\varepsilon(X, Y)$ is given by the number of facet defining inequalities of $\mathrm{conv}(X)$.

The second heuristic is a greedy algorithm that iteratively constructs an ε-relaxation as follows. The algorithm maintains a set $\mathcal{Y} \subseteq Y \setminus X$ of points that are not yet separated from X; initially, $\mathcal{Y}$ is thus $Y \setminus X$. In each iteration of the algorithm, a mixed-integer program is solved that finds an inequality separating as many points in $\mathcal{Y}$ from X as possible. Then, the newly separated points are removed from $\mathcal{Y}$ and the algorithm repeats the previous steps until $\mathcal{Y}$ becomes empty. The mixed-integer program that we used to find a separating inequality is a variant of Model (1) with $k = 1$, where the objective is to maximize $\sum_{y \in \mathcal{Y}} s_{y1}$.

4.2 Algorithmic aspects of the cutting plane model

To be able to deal with the exponentially many conflict inequalities (2c) in the cutting plane Model (2), we are relying on a separation routine. We start by discussing the case that the point $s^\star$ to be separated is contained in $\{0, 1\}^{Y \times [k]}$, i.e., for each of the k inequalities we already know which points it is supposed to separate.

To check whether $s^\star \in \{0, 1\}^{Y \times [k]}$ satisfies all conflict inequalities, we can compute for each $i \in [k]$ the set $F_i = \{y \in Y : s^\star_{yi} = 1\}$, and build a linear program similar to Model (1) that decides whether X and F_i are ε-separable. If the answer is yes, we know $s^\star$ is feasible. Otherwise, we have found a violated conflict inequality, namely $\sum_{y \in F_i} s_{yi} \leq |F_i| - 1$. Of course, this inequality will be rather weak in practice, because it excludes only the single assignment F_i.

One way to strengthen the inequality is to search for a minimum cardinality subset $F_{\min}$ of F_i, which cannot be separated from X. The corresponding inequality $\sum_{y \in F_{\min}} s_{yi} \leq |F_{\min}| - 1$ then does not only cut off $s^\star$, but every solution that assigns inequality i all points from $F_{\min}$. However, we do not expect that $F_{\min}$ can be computed efficiently, because detecting a minimum cardinality set of inequalities whose removal leads to a feasible LP is NP-hard, see Sankaran [27]. Instead, we compute a minimal cardinality subset $F \subseteq F_i$ by initializing $F = \emptyset$, adding points $y \in F_i$ to F until F and X are no longer separable, and then iterating over all points y' in F

and checking whether their removal leads to a separable set. In the latter case, we keep y' in F; otherwise, we remove it. Although this procedure is costly as it requires to solve $\Theta(|F_i|)$ LPs to find F, preliminary experiments revealed that the running time of the cutting plane model can be reduced drastically when using the sparsified inequalities.

To summarize, we use the previously mentioned scheme to separate integral points. As this scheme is already rather involved, we expect the generation of (2c) to be even more difficult for non-integral points. Thus, we only heuristically separate non-integral points $s^\star \in [0, 1]^{Y\times[k]}$ in our implementation. To this end, for each $i \in [k]$, we again initialize an empty set F and iteratively add $y \in Y$ in non-increasing order w.r.t. $s^\star_{yi}$ until $F \in \mathcal{C}$ and $s^\star$ violates the inequality (or we know that such an inequality cannot be violated).

4.2.1 Algorithmic aspects of the column generation model

In contrast to the compact model (1), the number of variables in (3) grows exponentially in $|Y|$, which makes it already challenging to solve the LP relaxation of (3). In our implementation, we thus use a branch-and-price procedure for solving (3), i.e., we use a branch-and-bound procedure in which each LP relaxation is solved by column generation. In the following, we discuss the different components of the branch-and-price procedure.

Solving the Root Relaxation At the root node of the branch-and-bound tree, we are given a subset $\mathcal{I}'$ of all possible variables in $\mathcal{I}$ and solve the LP relaxation of (3) restricted to the variables in $\mathcal{I}'$. To check whether the solution obtained for the variables in $\mathcal{I}'$ is indeed an optimal solution of the LP relaxation, we need to solve the pricing problem, i.e., to check whether all variables in $\mathcal{I}$ have non-negative reduced costs. Since the pricing problem is equivalent to the separation problem for the dual, we determine the dual of the root node LP relaxation of (3), which is given by

$$\max \sum_{y\in Y} \alpha_y \tag{6a}$$

$$\sum_{y\in I} \alpha_y \le 1, \qquad I \in \mathcal{I}, \tag{6b}$$

$$\alpha_y \ge 0, \qquad y \in Y. \tag{6c}$$

The pricing problem at the root node is thus to decide, for given dual weights α_y, $y \in Y$, whether there exists a set $I \in \mathcal{I}$ with $\sum_{y\in I} \alpha_y > 1$. Unfortunately, we cannot expect to solve this problem efficiently in general.

Proposition 6 *Let $X \subseteq \mathbb{Z}^d$ be finite and lattice-convex, let $Y \subseteq \mathbb{Z}^d \setminus X$ be finite, and let $\alpha_y \ge 0$ be a rational weight for $y \in Y$. Then, the pricing problem with input X, Y, and $\alpha \in \mathbb{Q}^Y$, for the LP relaxation of (3), i.e., deciding whether there exists $I \in \mathcal{I}(X, Y)$ with $\sum_{y\in Y} \alpha_y > 1$, is NP-hard.*

Proof Note that the pricing problem is equivalent to finding a set $I \in \mathcal{I}(X, Y)$ that maximizes the value $\sum_{y \in I} \alpha_y$. If all weights α_y, $y \in Y$, have the same value $\alpha > 0$, the problem reduces to find a set $I \in \mathcal{I}$ of maximum cardinality. The latter problem is NP-hard even if X consists of a single point, in which case it reduces to the open hemisphere problem, see Johnson and Preparata [19]. $\square$

To solve the pricing problem, we use a mixed-integer program that is a variant of (1) with $k = 1$. The only difference is that instead of minimizing the number of needed inequalities, we maximize the expression $\sum_{y \in Y} \alpha_y s_{y1}$. If this value is at most 1, we have found an optimal solution of the LP relaxation. Otherwise, we have found a variable z_I with negative reduced cost, add I to $\mathcal{I}'$, and iterate this procedure until all reduced costs are non-negative. In our implementation, we initialize the set $\mathcal{I}'$ by

$$\mathcal{I}' = \big\{\{y \in Y : a^\intercal y > b\} : a^\intercal x \leq b \text{ is facet defining for } \mathrm{conv}(X)\big\} \cup \big\{\{y\} : y \in Y\big\}.$$

Branching Strategy Let u be a node of the branch-and-bound tree and denote by z^u an optimal solution of the LP relaxation at node u. A classical branching strategy is to select a variable z_I with $z_I^u \notin \mathbb{Z}$ and to create two child nodes u^0 and u^1 by enforcing $z_I = 0$ in u^0 and $z_I = 1$ in u^1. While the branching decision $z_I = 1$ has strong implications for computing $\mathrm{rc}_\varepsilon(X, Y)$ (we basically fix an inequality used in the relaxation), branching $z_I = 0$ only rules out one of the exponentially many choices in $\mathcal{I}$ for a separated set.

To obtain a more balanced branching rule, we use the branching rule suggested by Ryan and Foster [26]. We are looking for two distinct variables z_I and z_J with z_I^u, $z_J^u \notin \mathbb{Z}$ such that both the intersection $I \cap J$ and symmetric difference $I \Delta J$ of I and J are non-empty. Let $y_1 \in I \cap J$ and $y_2 \in I \Delta J$. Then, two child nodes u^0 and u^1 of u are created as follows. In u^0, variables $z_{I'}$ are fixed to 0 if I' contains both y_1 and y_2. In u^1, we fix $z_{I'}$ to 0 if I' contains either y_1 or y_2. That is, u^0 enforces y_1 and y_2 to be contained in different sets, and u^1 forces them to be contained in the same set I'. This branching rule obviously partitions the integer solutions feasible at node u. To show its validity it is thus sufficient to show that for every non-integral solution z^u the sets I and J exist.

Lemma 7 *Let z^u be a non-integral optimal solution of the LP relaxation of* (3) *at node u of the branch-and-bound tree. Then, there exist two distinct sets $I, J \in \mathcal{I}'$ with $z_I^u, z_J^u \notin \mathbb{Z}$ such that $I \cap J \neq \emptyset$ and $I \Delta J \neq \emptyset$.*

Proof Let $I \in \mathcal{I}'$ be such that $z_I^u \notin \mathbb{Z}$. Then, $z_I^u \in (0, 1)$, since z^u is an *optimal* solution of the LP relaxation. Due to (3b), for every $y \in I$, there exists $J^y \in \mathcal{I}' \setminus \{I\}$ with $y \in J^y$ such that $z_{J^y}^u > 0$. For at least one J^y we have $z_{J^y}^u \in (0, 1)$, because otherwise, we could improve the objective value of z^u by setting z_I^u to 0 and still satisfying all constraints. Such a set J^y together with I satisfy the properties in the statement of the lemma: Since y is contained in both I and J^y, we have $I \cap J^y \neq \emptyset$. Moreover, as $I \neq J^y$, $I \Delta J^y \neq \emptyset$. $\square$

In our implementation, we compute for each variable z_I^u its fractionality $\theta(I) = \frac{1}{2} - \min\{z_I^u, 1 - z_I^u\}$. Then, we select I and J such that $\theta(I) + \theta(J)$ is maximized; the branching candidates $y_1 \in I \cap J$ and $y_2 \in I \Delta J$ are selected arbitrarily. Note that other scores than the fractionality can be used to define branching rules and that this might have a big impact on the performance of branch-and-price codes [31]. In our experiments, however, we observed that the main computational bottleneck seems to be solving the pricing problem. We have therefore not conducted experiments with other branching rules.

Solving LP Relaxations in the Tree To not re-generate variables that have been fixed to 0 by the branching rule, we need to incorporate the branching decisions active at a node of the branch-and-bound tree into the pricing problem. This can easily be done by adding linear constraints to the root node formulation of the pricing problem. If a branching decision was that y_1 and y_2 shall be contained in different sets, we add $s_{y_1 1} + s_{y_2 1} \leq 1$ to the pricing problem. The branching decision that y_1 and y_2 have to be contained in the same set can be enforced by the constraint $s_{y_1 1} = s_{y_2 1}$.

5 Numerical experiments

The aim of this section is to compare the practical performance of the three models for computing $\mathrm{rc}_\varepsilon(X, Y)$ as well as their enhancements. To this end, we have implemented all three models in C/C++ using `SCIP 7.0.3` [11] as modeling and branch-and-bound framework and `SoPlex 5.0.2` to solve all LP relaxations. All branching, propagation, separation, and pricing methods are implemented using the corresponding plug-in types of `SCIP`. Since we are not aware of an alternative separation routine for hiding set cuts, we compute all hiding sets of size two in a straightforward fashion before starting the branch-and-bound process. During the solving process, we separate these inequalities if the corresponding cuts are violated. To handle symmetries via lexicographic orderings, we use `SCIP`'s internal plug-ins `cons_orbitope`, `cons_orbisack`, and `cons_symresack` that implement the methods discussed in Sect. 4; the branching and pricing plug-ins for the column generation model strongly build up on the corresponding plug-ins of the binpacking example provided in the SCIP Optimization Suite. All convex hull computations have been carried out using `cdd 0.94m` [10] and graph symmetries are detected using `bliss 0.73` [20].

Our implementation is available online at github[3] and zenodo [14].

Implementation Details All models admit some degrees of freedom that we detail in the following. Both the compact model and the cut model require an upper bound on the relaxation complexity. In both models, we run the heuristics discussed in Sect. 4 to find an upper bound k on $\mathrm{rc}_\varepsilon(X, Y)$. We also use the heuristic solution with the least number of inequalities as an initial solution. A comparison of the quality of solutions found by the facet and greedy heuristics is provided by Table 1 of this article's online supplement. In the column generation model, we need to select a subset of $\mathcal{I}$ to define initial variables. In our experiments, we used two different variants to initialize $\mathcal{I}$. On the one hand, we use the sets $I \in \mathcal{I}$ that are defined by the facet defining inequalities

[3] https://github.com/christopherhojny/relaxation_complexity (githash c2edaaae was used for our experiments).

of conv(X), i.e., the sets of points in Y that are separated from X by the facet defining inequalities. On the other hand, if the greedy heuristic provides a solution with fewer inequalities, we use the corresponding separated sets instead. Moreover, we include the singleton sets $\{y\}$, for $y \in Y$, to make sure that the LP relaxation remains feasible after branching. The reason why we experimented with two different initializations is that the initial variables have a strong impact on the performance of the branch-and-price algorithm, as we will discuss below.

Since we observed that the mixed-integer programs solved as a subroutine of our greedy heuristic require a lot of time, we do not solve the subproblems to global optimality. Instead, we impose a limit of 1000 nodes in the branch-and-bound trees used to solve each subproblem. We imposed a node limit rather than a time limit to obtain a deterministic algorithm, which is not impacted by performance variability when solving the subproblems. Moreover, we have also tried higher node limits to find better solutions to the subproblems, see "Appendix A". Interestingly, the quality of heuristic solutions degraded with a higher node limit, which seems counterintuitive at first glance. However, by not separating the maximum number of points in the first iterations, there are more degrees of freedom in later iterations of the heuristic that allow to save some inequalities.

Settings To encode the different settings that we have tested, we make use of the following abbreviations:

hiding	Whether hiding set cuts are added (1) or not (0).
sym.	Which symmetry method is used: none (0), simple (s), or advanced (a), where simple is (4) and (5), and advanced uses (4) and additionally enforcing lexicographically maximal solutions based on symmetries of X and Y.
prop.	Whether the convexity propagator is used (1) or not (0).

Note that we do not report on results for the intersection propagation algorithm. This is because, in preliminary experiments, we have seen that its running time is very high, in particular, because it needs to compute in each iteration $\mathcal{O}(|Y|)$ convex hulls. As a result, we could hardly solve any instance, not even small ones.

Test Sets In our experiments, we have used three different test sets:

basic	The sets X are the integer points in the 0/1 cube, the crosspolytope, or the standard simplex in dimensions $d \in \{3, 4, 5\}$. Thus, the sizes of these test sets are 2^d, $2d + 1$, and $d + 1$, respectively. For any such $X \subseteq \mathbb{Z}^d$, the sets Y consist of all points in $\mathbb{Z}^d \setminus X$ whose ℓ_1-distance to X is at most ℓ, where $1 \leq \ell \leq 10 - d$. The reason for smaller distance in higher dimension is that the problems get considerably more difficult to solve with increasing ℓ.
downcld	This test set consists of 99 full-dimensional subsets X of $\{0, 1\}^5$ that correspond to *down-closed subsets* (or abstract simplicial complexes) of the Boolean lattice on 5 elements. This means that, whenever a point $x \in X$, then every vector $x' \in \{0, 1\}^5$ that differs from x only in the coordinates i with $x_i = 1$, also belongs to X. The corresponding sets Y are the points in $\mathbb{Z}^5 \setminus X$ whose ℓ_1-distance to X is at most $\ell \in \{1, 2, 3\}$. The sets X have been generated by the natural one-to-one correspondence between

inclusion-wise maximal sets in a down-closed family and antichains in the Boolean lattice.

sboxes The test set comprises 18 instances modeling 4-bit (12 instances) and 5-bit (6 instances) S-boxes, which are certain non-sparse Boolean functions arising in symmetric-key cryptography. The derived sets X are contained in $\{0, 1\}^8$ and $\{0, 1\}^{10}$, respectively, and Y are the complementary binary points. These instances have also been used by Udovenko [30] who solved the full model (3), i.e., without column generation.

The basic instances feature various aspects that might be relevant for computing $\mathrm{rc}(X)$ via computing a series of values $\mathrm{rc}_\varepsilon(X, Y)$ for different Y and ε according to [4]: The cube is parity complete, thus there exists a small set Y such that $\mathrm{rc}(X) = \mathrm{rc}_\varepsilon(X, Y)$ (in fact, this set is $\{-1, 0, 1, 2\}^d \setminus X$); the crosspolytope has an interior integer point and thus there exists a (potentially large) finite set Y with $\mathrm{rc}(X) = \mathrm{rc}_\varepsilon(X, Y)$; for the simplex Δ_4 in $\mathbb{R}^4$, no finite set Y exists with $\mathrm{rc}(\Delta_4) = \mathrm{rc}(\Delta_4, Y)$; see [3]. That is, $\mathrm{rc}(\Delta_4, Y) < \mathrm{rc}(\Delta_4) \leq \mathrm{rc}_{\mathbb{Q}}(X, Y)$ for all finite sets $Y \subseteq \mathbb{Z}^4$.

Since the standard simplex Δ_d is a down-closed subset of $\{0, 1\}^d$, the small-sized downcld instances might be good candidates for further examples X such that $\mathrm{rc}_\varepsilon(X, Y) < \mathrm{rc}_{\mathbb{Q}}(X)$, for every finite set $Y \subseteq \mathbb{Z}^d$ and for $\varepsilon > 0$ small enough. Our aim for selecting these instances is thus to identify whether there are potentially further candidates for sets X whose relaxation complexity cannot be computed via finite sets Y.

Finally, the sboxes instances are used to investigate whether our techniques are suited to compute $\mathrm{rc}_\varepsilon(X, Y)$ also in higher dimensions. This is relevant, among others, in the field of social choice or symmetric cryptanalysis, where the aim is to find $\mathrm{rc}(X, \{0, 1\}^d)$ for sets $X \subseteq \{0, 1\}^d$. As described in more detail in [30], the dimensions of the sets $X \subseteq \{0, 1\}^d$ in our sboxes test set are twice the input length of the used S-boxes in the analyzed cryptographic scheme. This input length is typically in the range of 4–6 bits, so that our instances of dimensions $d \in \{8, 10\}$ are representative for the application.

We provide an overview of the exact sizes of X and Y for the different instances as well as the best bounds we could obtain using the different settings in Table 1 of this article's online supplement. This supplement also contains detailed information about the bounds on $\mathrm{rc}_\varepsilon(X, Y)$ found by the different methods.

Computational Setup All experiments have been run on a Linux cluster with Intel Xeon E5 3.5 GHz quad core processors and 32 GB memory. The code was executed using a single thread and the time limit for all computations was 4 h per instance.

All mean numbers are reported in shifted geometric mean $\prod_{i=1}^{n}(t_i + s)^{\frac{1}{n}} - s$ to reduce the impact of outliers. For mean running times, a shift of $s = 10$ is used; for nodes of the branch-and-bound tree, we use $s = 100$. The value of ε in computing $\mathrm{rc}_\varepsilon(\cdot, \cdot)$ is set to 0.001. The upper bound on the number of inequalities needed in the compact and cutting plane model is given by the number of facets of $\mathrm{conv}(X)$. We also provide an initial primal solution corresponding to a facet description of $\mathrm{conv}(X)$ or a solution found by the greedy heuristic.

5.1 Results for test set basic

Due to our choice of the sets X and Y, the basic test set comprises 18 cube, crosspolytope, and simplex instances, respectively. Table 1 shows the results for the compact model. For the plain compact model, we observe that SCIP can already solve quite some instances, but, in comparison to the enhanced variants, the running times are rather high. Checking each of the enhancements separately, handling symmetries is most important to reduce running time and to increase the number of instances solvable within the time limit (except for the simplex test set). Interestingly, handling symmetries on the a-variables modeling the inequalities in a relaxation performs better than handling the symmetries of the points to separate. Adding hiding set cuts to the problem formulation is also beneficial, whereas the convexity propagator only slightly improves the running time for crosspolytope and simplex instances, and harms the solution process for cube instances. For the latter, the worse performance for enabled propagation cannot be explained by the running time of the propagator: For cube instances, e.g., the shifted geometric mean running time per instance of the propagator was 3.1 s, which is much smaller than the increase of mean running time. Thus, it seems that the found reductions guide the branch-and-bound search into the wrong direction or make it more difficult for SCIP to find other reductions.

Regarding symmetry handling, the simple symmetry handling method performs better than the advanced method if not other enhancements are enabled. Additionally enabling hiding set cuts, however, neither the simple nor the advanced method dominates the other method. For cube and simplex instances, the advanced method requires roughly two thirds of the running time of the simple method; for crosspolytopes, the advanced method is, however, slower by roughly 33%. Comparing simple symmetry handling and hiding set cuts with the basic model, we achieve significant performance improvements of 86% for cubes, 85% for crosspolytopes, and 72% for simplices, and similar trends can be found for the advanced symmetry handling method with hiding set cuts. We conclude that hiding set cuts and symmetry handling are important enhancements for solving the compact model efficiently.

Next, we discuss two variants of the column generation model for which we only compare two variants each: we either disable or enable hiding set cuts in the pricing problem. Since the convexity propagator does not seem to be helpful for the compact model, we do not enable it when solving the pricing problem. Moreover, symmetry handling is not important, because there is only one inequality to be identified by the pricing model. The results are summarized in Tables 2 and 3, where we consider the column generation model with and without the iterative greedy heuristic, respectively. Our motivation for these two variants is that the initial variables $\mathcal{I}$ have a big impact on the performance of the column generation procedure. It is not clear whether the potentially larger set $\mathcal{I}$ provided by facet defining inequalities performs better than $\mathcal{I}$ defined via the iterative greedy solution.

Indeed, comparing the two variants of the column generation model, we see that, in general, disabling the iterative greedy heuristic performs much better. When hiding set cuts are enabled, the version without the heuristic is 39%, 10%, and 26% faster

Table 1 Run times for different settings using the compact model

Setting			cube (18 instances)		cross (18 instances)		simplex (18 instances)	
Hiding	Sym	Prop	Time	#solved	Time	#solved	Time	#solved
0	0	0	283.3	13	583.5	12	209.4	15
0	0	1	322.9	13	519.0	12	204.9	15
0	a	0	264.0	14	377.3	12	144.8	14
0	s	0	179.7	15	164.2	15	113.7	15
1	0	0	187.5	14	224.8	15	73.0	15
1	a	0	40.4	18	115.8	16	40.1	17
1	a	1	40.1	18	115.9	16	50.1	17
1	s	0	62.4	18	87.2	16	58.3	16
1	s	1	64.4	18	90.2	17	60.9	16

Table 2 Run times for different settings using the column generation model

Setting			cube (18 instances)		cross (18 instances)		simplex (18 instances)	
Hiding	Sym	Prop	Time	#solved	Time	#solved	Time	#solved
0			919.5	7	887.0	7	916.2	8
1			123.6	13	648.4	10	521.6	11

Table 3 Run times for different settings using the column generation model without greedy heuristics

Setting			cube (18 instances)		cross (18 instances)		simplex (18 instances)	
Hiding	Sym	Prop	Time	#solved	Time	#solved	Time	#solved
0			878.4	10	3258.3	4	697.8	9
1			75.9	14	584.2	9	387.0	12

for cube, crosspolytope, and simplex instances, respectively, than the version with the heuristic.

Comparing the column generation model without the iterative greedy heuristic and enabled hiding set cuts with the plain compact model, we see that the column generation model performs much better for cube instances, performs comparably for crosspolytope instances, and is much worse for simplex instances. For cubes, all solvable instances are solved within the root node which is, on the one hand, because of the strong dual bound as described in Proposition 3. On the other hand, the generated sets $I \in \mathcal{I}$ allow heuristics to find high quality solutions yielding a matching primal bound. Looking onto results on a per-instance basis reveals that the pricing problems become considerably harder if d and ℓ increases. For example, the solvable cube instances (greedy heuristic and hiding set cuts enabled) require a shifted geometric mean running time of 9.4 s. For $d = 5$ and $\ell \geq 2$, however, only 4–10 nodes of the branch-and-bound tree can be processed. For the simplex instances, the column generation model needs approximately twice as much time as the plain compact model, which is again explained by the very high running time of the pricing problem.

Finally, we consider the cutting plane model. In preliminary experiments we observed that the plain version without iterative greedy heuristic can hardly solve any instance efficiently. Enabling the heuristic, however, leads to a much smaller model as the strength of the model highly depends on the quality of the best known upper bound. Comparing the different enhancements with each other, we can see, analogously to the compact model, that adding hiding set cuts and handling symmetries is beneficial as it reduces the running time by 92–99% (Table 4). Interestingly, the advanced symmetry handling methods work consistently better than the simple symmetry handling methods, whereas we observed no clear trend for the compact model. A possible explanation is that the simple methods just sort the used inequalities in the cut model, whereas, in the compact model, they also impact the a-variables. Since the latter is not possible in the cut model, only a small portion of symmetries is handled. The advanced methods compensate the weaker effect of the simple methods by also sorting the patterns of separated points.

In summary, the column generation model provides very good primal and dual bounds. If these bounds match, $\mathrm{rc}_{\varepsilon}(X, Y)$ can be computed rather efficiently if not too many pricing problems need to be solved. However, if the bounds do not match, the NP-hardness of the pricing problem strikes back and solving many further pricing problems is too expensive. In this case, the compact and cut models are rather effective alternatives that also allow to compute $\mathrm{rc}_{\varepsilon}(X, Y)$ for $d = 5$ in many cases. Comparing the latter two models, the compact model is the more competitive alternative as, for crosspolytope and simplex instances, it performs 67% and 82%, respectively, better than the cut model in the respective best settings. For cube instances, the compact model performs 50% better.

5.2 Results for test set downcld

In this section, we turn the focus on 5-dimensional 0/1 down-closed sets. On the one hand, our aim is to investigate whether the findings of the previous section carry over

Table 4 Run times for different settings usign the cut model

Setting			cube (18 instances)		cross (18 instances)		simplex (18 instances)	
Hiding	Sym	Prop	Time	#solved	Time	#solved	Time	#solved
0	0	0	7312.0	7	7200.9	5	2759.3	7
0	0	1	7359.9	7	6979.7	5	2725.3	7
0	a	0	3124.2	8	3774.1	10	1809.0	10
0	s	0	4351.5	7	5358.9	8	2016.5	8
1	0	0	985.4	10	513.4	10	372.5	12
1	a	0	87.1	15	267.5	11	243.9	12
1	a	1	80.9	16	273.0	11	223.9	12
1	s	0	175.5	14	12.3	11	294.1	11
1	s	1	176.5	14	393.9	11	294.8	12

to a much broader test set in dimension 5. On the other hand, we are interested in identifying further sets $X \subseteq \{0,1\}^5$ with $\mathrm{rc}_\varepsilon(X,Y) < \mathrm{rc}_\mathbb{Q}(X)$ for every choice of a finite set $Y \subseteq \mathbb{Z}^d$ and $\varepsilon > 0$ small enough. Because of our results on the basic test set, we did not run any experiments using the cutting plane model as we expect that it performs worse than the compact model. Instead, we consider a hybrid version of the compact model and the column generation model: We only solve the column generation model's LP relaxation to derive a strong lower bound on $\mathrm{rc}_\varepsilon(X,Y)$ and to find good primal solutions. Both are transferred to the compact model with the hope to drastically reduce solving time. The running times and number of nodes reported for the hybrid model are means of the total running time and total number of nodes for solving the LP relaxation in the column generation model and the resulting compact model.

Table 5 shows aggregated results for the 99 instances of the downcld test set for different ℓ_1-neighborhoods Y of X (radius 1–3). While the plain compact model is able to solve almost all instances for radius 1, computing $\mathrm{rc}_\varepsilon(X,Y)$ for larger radii becomes much harder. As the plain model can only solve roughly a quarter of all instances with radius 2 and hardly any instance for radius 3, there is definitively a need for model enhancements. In general, the same observations as in the previous section can be made: symmetry handling and adding hiding set cuts improve the solution process a lot. The biggest impact is achieved by hiding set cuts. Interestingly, without hiding set cuts, the simple symmetry handling methods dominate the advanced techniques. When hiding set cuts are enabled, however, the advanced methods perform significantly better than the simple symmetry handling methods. The consistently best setting is thus to enable hiding set cuts and advanced symmetry handling methods. It allows us to solve all instances with radius at most 2; for radius 3, we can solve 68 out of the 99 instances. Regarding the number of solved instances, however, we observe that simple symmetry handling allows to solve 6 more instances for radius 3, while the mean running time increases by 37%.

A possible explanation is based on the nature of the advanced setting: Each inequality defining a relaxation of X w.r.t. Y defines a pattern on the points from Y that are cut by this inequality. The advanced method enforces that the cut patterns of the inequalities are sorted lexicographically based on a sorting of the elements of Y. Since the results of the lexicographic comparison is determined by the first position in which two vectors differ, it is unlikely that points having a late position in the ordering of Y are very relevant for the lexicographic constraint. Thus, the symmetries are in a certain sense mostly handled for the early points in this ordering. In contrast to this, the simple method takes the geometry of the inequalities in a relaxation into account by sorting inequalities based on their first coefficients. If also other components are enabled, however, the geometric information of the simple symmetry handling methods can also be exploited by the advanced symmetry handling techniques, which yields a better running time in particular for instances with large radius.

For the column generation model, we again have conducted experiments with enabled and disabled iterative greedy heuristic. As the results do not differ qualitatively, we only elaborate on the version without the heuristic in the following. In comparison to the enhanced compact model, the column generation model is again inferior. For radius at least 2, it can hardly solve any instance and, as already discussed

Table 5 Comparison of running times for different settings for downcld instances

Setting			Radius 1 (99 instances)			Radius 2 (99 instances)			Radius 3 (99 instances)		
Hiding	Sym	Prop	#solved	#nodes	Time	#solved	#nodes	Time	#solved	#nodes	Time
Compact model											
0	0	0	95	75,389.8	333.0	27	472,466.3	10,199.4	2	87,019.8	14,302.6
0	0	1	94	75,154.5	351.3	30	494,155.6	10,525.3	2	89,688.5	14,336.8
0	a	0	99	26,828.6	131.9	50	412,021.5	8053.6	3	114,261.0	13,995.1
0	s	0	99	22,739.3	89.0	87	374,855.2	3695.0	7	262,680.2	13,464.6
1	0	0	96	3437.5	74.2	64	86,274.9	3006.7	16	72,317.5	11,174.3
1	a	0	99	172.7	11.8	99	1784.6	190.1	68	12,368.7	3208.1
1	a	1	99	172.8	11.8	99	1845.5	201.2	70	12,643.2	3275.4
1	s	0	99	544.9	18.6	99	6429.9	420.5	74	29,777.4	4407.5
1	s	1	99	544.9	18.7	99	6261.1	410.3	75	28,816.4	4134.3
Column generation model											
0			1	5.4	13,798.5	0	2.7	14,400.0	0	1.3	14,400.0
1			55	45.5	957.0	5	33.8	12,597.5	2	11.9	13,524.7
Column generation model without greedy heuristic											
0			5	2.6	11,278.8	0	1.4	14,400.0	0	2.0	14,400.0
1			60	67.4	1188.9	4	34.9	12,135.6	1	12.3	13,965.1
Hybrid model											
0	s	0	8	16.6	8797.4	7	64.7	11,388.8	3	98.5	13,561.4
1	s	0	98	360.4	21.8	95	3458.9	181.8	84	13,553.2	1421.4
0	a	0	8	19.8	8846.6	7	69.6	11,645.0	3	94.9	13,566.4
1	a	0	98	201.9	19.5	95	1552.5	133.0	86	9677.0	1195.4
Hybrid model without greedy heuristic											
0	s	0	3	5.8	11,722.9	1	8.1	13,993.8	0	1.0	14,400.0
1	s	0	97	364.4	20.3	95	3797.4	173.4	83	13,921.7	1352.5
0	a	0	3	6.3	11,718.5	1	8.9	14,093.7	0	1.0	14,400.0
1	a	0	97	203.2	17.7	95	1710.8	125.9	85	8799.4	1043.0

in the previous section, the reason for this is the long running time of the pricing models that need to be solved often at each node of the tree. This is reflected by the number of processed nodes during the branch-and-price procedure that drops drastically (as the number of solved instances) if the radius is getting larger. However, we can again observe that the root node can be solved relatively efficiently and that the obtained primal and dual bounds are rather strong. This is reflected in the hybrid model, which solves most instances with radius 3 and reduces the running time (in comparison to the best compact model) by 34% for radius 2 and 67% for radius 3. For radius 1, the running time of the hybrid model is roughly 50% larger, but the absolute difference is rather small.

In summary, based on our experiments, the hybrid model is the best choice for computing $\mathrm{rc}_\varepsilon(X, Y)$ as it combines the strong bounds from the column generation model with the ability of the compact model to quickly solve LP relaxations within the branch-and-bound tree. In particular, it benefits from hiding set cuts since their implications are very difficult to be found by SCIP.

Finally, concerning our goal to identify candidates for sets $X \subseteq \{0, 1\}^5$ such that $\mathrm{rc}_\varepsilon(X, Y) < \mathrm{rc}_\mathbb{Q}(X)$ for all finite $Y \subseteq \mathbb{Z}^5$ and $\varepsilon > 0$ small enough, our experiments for radius 3 revealed the following: If $Y(X)$ are the integer points in the ℓ_1-neighborhood of X with radius 3, then there are three sets X such that $\mathrm{rc}_\varepsilon(X, Y(X)) = 4$. These sets are Δ_5, $\Delta_5 \cup \{e_1 + e_2\}$ and $\Delta_4 \times \{0, 1\}$. Moreover, there are 16 sets X with $\mathrm{rc}_\varepsilon(X, Y(X)) = 5$. It is left open for future research to identify which other sets than Δ_5 satisfy $\mathrm{rc}_\varepsilon(X, Y(X)) < \mathrm{rc}_\mathbb{Q}(X)$. Note that $\mathrm{rc}_\mathbb{Q}(X) \geq 6$, whenever $X \subseteq \{0, 1\}^5$ is full-dimensional, because rational relaxations must be bounded.

5.3 Results for test set sboxes

The results for the sboxes test set are summarized in Table 6. In preliminary experiments, we observed that the iterative greedy heuristic is very important for solving the compact model. If we do not use this heuristic, none of the 10-dimensional instances could be solved with enabled hiding set cuts, because all these experiments hit a memory limit of 20 GB. The reason is that these models grow very large even without any enhancements as we use the number of facets of $\mathrm{conv}(X)$ to upper bound $\mathrm{rc}_\varepsilon(X, Y)$; the number of facets for these instances ranges between 888 and 2395. For this reason, we only report on the numbers with enabled heuristic in Table 6.

But even when the heuristic is enabled, the compact model turns out to be very challenging to solve. In dimension 8, both symmetry handling methods and hiding set cuts are needed to solve instances within the time limit: the advanced method allows to solve three instances, whereas the simple symmetry handling method only allows to solve a single instance. In dimension 10, no instance can be solved.

In contrast to this, we see that the column generation model performs extremely well for the problems in dimension 8. It can solve all twelve 8-dimensional instances within the time limit, on average in 69.9 s if hiding set cuts and the greedy heuristic are enabled. When hiding set cuts are disabled, the running time roughly doubles and only 8 instances can be solved. Disabling the greedy heuristic has an interesting effect. On the one hand, it drastically reduces the running time when hiding set cuts are

Table 6 Comparison of running times for different settings for sboxes instances

Setting			Dimension 8 (12 instances)			Dimension 10 (6 instances)		
Hiding	Sym	Prop	#solved	#nodes	Time	#solved	#nodes	Time
Compact model								
0	0	0	0	367,460.1	14,400.0	0	7877.6	14,400.0
0	0	1	0	369,738.1	14,400.0	0	7705.2	14,400.0
0	a	0	0	361,190.9	14,400.0	0	5318.0	14,400.0
0	s	0	0	739,363.8	14,400.0	0	3727.3	14,400.0
1	0	0	0	108,496.5	14,400.0	0	460.6	14,400.0
1	a	0	3	207,232.0	11,220.9	0	243.4	14,400.0
1	s	0	1	31,090.4	13,792.2	0	67.4	14,400.0
Column generation model								
0			8	35.4	981.8	1	1.2	9320.9
1			12	69.9	443.4	0	1.7	14,400.0
Column generation model without greedy heuristic								
0			9	21.4	245.0	2	8.6	6845.6
1			10	64.9	461.3	1	3.2	14,273.9
Hybrid model								
0	s	0	4	7193.8	2023.4	1	5093.9	9476.6
1	s	0	6	31,439.0	1937.1	0	7.8	14,400.0
0	a	0	4	7888.8	2040.4	1	8365.8	9479.1
1	a	0	6	52,965.3	2191.1	0	20.1	14,400.0
Hybrid model without greedy heuristic								
0	s	0	6	5992.6	635.8	2	622.1	4627.5
1	s	0	6	21,585.5	1390.0	1	36.0	14,090.7
0	a	0	6	7752.4	635.7	2	983.4	4619.2
1	a	0	4	59,210.9	2770.9	1	87.2	14,088.8

disabled; with enabled hiding set cuts, however, it cannot solve all instances anymore. For dimension 10, the column generation model is also able to solve 2 out of 6 instances within the time limit when both the greedy heuristic and hiding set cuts are disabled.

Finally, the hybrid model performs worse than the column generation model in dimension 8. Although the derived bounds from solving the column generation model's LP relaxation yield again very good bounds on the relaxation complexity, the value of $\mathrm{rc}(X, \{0, 1\}^d)$ can still be large if $d \in \{8, 10\}$. Thus, also the compact model embedded in the hybrid model is struggling with the number of variables and constraints. In dimension 10, the hybrid model with disabled heuristic and hiding set cuts performs much better than the corresponding column generation model. However, as only two instances can be solved to optimality, it is difficult to draw reliable conclusions. For this reason, computing $\mathrm{rc}(X, \{0, 1\}^d)$ via the column generation model is most competitive.

5.4 Conclusions

Being able to compute the exact value of the quantity $\mathrm{rc}_\varepsilon(X, Y)$ is highly relevant in many areas, such as, social choice, symmetric cryptanalysis, or machine learning. For this reason, we have proposed three different models that allow to compute $\mathrm{rc}_\varepsilon(X, Y)$ using mixed-integer programming techniques. As our experiments reveal, each of these models comes with advantages and disadvantages. The compact model, for example, works well in small dimensions as the number of variables and inequalities is small and it encapsulates all essential information about $\mathrm{rc}_\varepsilon(X, Y)$. In higher dimensions, however, the dual bounds of the compact model become weaker. In this case, the column generation model provides very good bounds that can be transferred to the compact model to still compute $\mathrm{rc}_\varepsilon(X, Y)$ rather efficiently if $d = 5$. But if the dimension d grows even larger, only the column generation model seems to be competitive as it does not scale as badly as the compact model when $\mathrm{rc}_\varepsilon(X, Y)$ increases. The main reason is that the compact model is relying on a good upper bound on $\mathrm{rc}_\varepsilon(X, Y)$ to be indeed compact. A possible drawback of the column generation model is, however, that it is very sensitive to the initial set of variables $\mathcal{I}$.

These findings thus open the following directions for future research. Since the compact model requires a good upper bound on $\mathrm{rc}_\varepsilon(X, Y)$, it is natural to investigate heuristic approaches for finding ε-relaxations of X or to develop approximation algorithms. Moreover, since the column generation model becomes more relevant if d is large, it is essential that the pricing problem can be solved efficiently. Since the pricing problem is NP-hard, also here a possible future direction could be to develop heuristics or approximation algorithms for solving it. For both the compact and column generation model, hiding set cuts turned out to be useful. However, we are not aware of an efficient routine for generating these cutting planes. Thus, it is natural to devise an efficient scheme for generating hiding set cuts on the fly. As additional inequalities such as hiding set cuts and symmetry handling inequalities drastically improved the performance of the compact model, the development of further inequalities modeling structural properties of relaxation complexity might allow to solve the compact model even more efficiently.

Finally, a natural approach to compute $\mathrm{rc}_\varepsilon(X, Y)$ for finite sets Y is to consider a sequence of sets $(Y_i)_{i \geq 0}$ such that $Y_i \subseteq Y_{i+1} \subseteq Y$ and to compute $\mathrm{rc}_\varepsilon(X, Y_i)$ until the found relaxation w.r.t. Y_i is already a relaxation w.r.t. Y. Preliminary experiments with this approach using our code reveal that it might even take longer to compute $\mathrm{rc}_\varepsilon(X, Y_i)$ than $\mathrm{rc}_\varepsilon(X, Y)$ and that the sets Y_i need to be selected carefully to yield a bound close to $\mathrm{rc}_\varepsilon(X, Y)$. We thus believe that an analysis of the geometry of X and Y is needed to find meaningful sets Y_i for which $\mathrm{rc}_\varepsilon(X, Y_i)$ can be found efficiently. This is out of scope of this article though.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12532-023-00241-9.

Acknowledgements We thank Aleksei Udovenko for providing the sboxes instances used by him in 31. Moreover, we thank the anonymous referees for their very valuable comments, in particular regarding the heuristics, that helped to substantially improve our results.
All authors contributed to the study conception and design. Material preparation, data collection and analysis

were performed by Gennadiy Averkov, Christopher Hojny and Matthias Schymura. The first draft of the manuscript was written by Christopher Hojny and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availibility The datasets generated during and/or analysed during the current study are available in the github repository https://github.com/christopherhojny/relaxation_complexity (githash c2edaaae). A persistent link to the dataset used can be found in [14].

Declaration

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Code availability The code generated during the current study is available in the github repository https://github.com/christopherhojny/relaxation_complexity (githash c2edaaae). A persistent link to the code can be found in [14].

A Comparison of different node limits for the greedy heuristic

In this section, we provide some statistics on the impact of the node limit per subproblem of the greedy heuristic on the quality of the solution provided by the greedy heuristic. Table 7 compares three parameterizations of the greedy heuristic by imposing a node limit of 1000, 10,000, and 100,000, respectively. The left part of the table compares the average running times of the three parameterizations. One can clearly see that the running time increases drastically when increasing the node limit. To see whether the increased running time is compensated by finding relaxations with less inequalities, we computed for each instance the number of inequalities used by the three greedy solutions and checked how often a parameterization found a solution with the least number of inequalities. These numbers are reported in the right part of the table. For many test sets, the parameterization using 1000 performs best, i.e., finds in most cases a relaxation with the least number of inequalities. Thus, the increase in running time when using a higher node limit is not compensated by finding better solutions than parameterizations with a lower node limit in general. For this reason, the default node limit imposed in our experiments is 1000.

Table 7 Comparison of different parameterizations of the greedy heuristic

Test set	Node limit/running times			Node limit/best solution		
	1000	10,000	100,000	1000	10,000	100,000
cube	11.54	61.06	297.82	17	15	13
cross	16.23	82.72	421.07	16	16	14
simplex	16.42	102.99	541.39	15	14	17
downcld radius 1	3.34	17.95	58.74	74	68	74
downcld radius 2	10.84	59.06	382.92	69	78	76
downcld radius 3	23.19	137.96	773.11	61	67	63
sboxes dimension 8	11.78	56.98	114.16	7	3	4
sboxes dimension 10	242.97	680.57	2838.90	1	3	4

References

1. Aprile, M., Averkov, G., Di Summa, M., Hojny, C.: The role of rationality in integer-programming relaxations. https://arxiv.org/abs/2206.12253 (2022)
2. Astorino, A., Gaudioso, M.: Polyhedral separability through successive LP. J. Optim. Theory Appl. **112**, 265–293 (2002)
3. Averkov, G., Hojny, C., Schymura, M.: Computational aspects of relaxation complexity: possibilities and limitations. Math. Program. (2021). https://doi.org/10.1007/s10107-021-01754-8
4. Averkov, G., Schymura, M.: Complexity of linear relaxations in integer programming. Math. Program. (2021). https://doi.org/10.1007/s10107-021-01623-4
5. Bader, J., Hildebrand, R., Weismantel, R., Zenklusen, R.: Mixed integer reformulations of integer programs and the affine TU-dimension of a matrix. Math. Program. **169**(2), 565–584 (2018). https://doi.org/10.1007/s10107-017-1147-2
6. Bendotti, P., Fouilhoux, P., Rottner, C.: Orbitopal fixing for the full (sub-)orbitope and application to the unit commitment problem. Math. Program. **186**, 337–372 (2021). https://doi.org/10.1007/s10107-019-01457-1
7. Cevallos, A., Weltge, S., Zenklusen, R.: Lifting linear extension complexity bounds to the mixed-integer setting. In: Czumaj, A. (ed.) Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7–10, 2018, pp. 788–807. SIAM (2018). https://doi.org/10.1137/1.9781611975031.51
8. Conforti, M., Cornuéjols, G., Zambelli, G.: Extended formulations in combinatorial optimization. Ann. Oper. Res. **204**(1), 97–143 (2013). https://doi.org/10.1007/s10479-012-1269-0
9. Dundar, M.M., Wolf, M., Lakare, S., Salganicoff, M., Raykar, V.C.: Polyhedral classifier for target detection: a case study: colorectal cancer. In: ICML '08: Proceedings of the 25th International Conference on Machine Learning, pp. 288–295 (2008)
10. Fukuda, K.: cdd/cdd+ Reference Manual. Institute for Operations Research, ETH-Zentrum, pp. 91–111 (1997)
11. Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Le Bodic, P., Maher, S.J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M.E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D., Witzig, J.: The SCIP Optimization Suite 7.0. Technical report, Optimization Online (2020). http://www.optimization-online.org/DB_HTML/2020/03/7705.html
12. Hojny, C.: Polynomial size IP formulations of knapsack may require exponentially large coefficients. Oper. Res. Lett. **48**(5), 612–618 (2020)
13. Hojny, C.: Strong IP formulations need large coefficients. Discrete Optim. **39**, 100624 (2021)
14. Hojny, C.: christopherhojny/relaxation_complexity: Version 1.0.0 (2023). https://doi.org/10.5281/zenodo.7755327

15. Hojny, C., Pfetsch, M.E.: Polytopes associated with symmetry handling. Math. Program. **175**, 197–240 (2019). https://doi.org/10.1007/s10107-018-1239-7
16. Hooker, J.N.: Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. Wiley, New York (2000)
17. Hrubeš, P., Talebanfard, N.: On the extension complexity of polytopes separating subsets of the Boolean cube. arXiv: 2105.11996 (2021)
18. Jeroslow, R.G.: On defining sets of vertices of the hypercube by linear inequalities. Discrete Math. **11**, 119–124 (1975)
19. Johnson, D., Preparata, F.: The densest hemisphere problem. Theor. Comput. Sci. **6**, 93–107 (1978)
20. Junttila, T., Kaski, P.: bliss: A tool for computing automorphism groups and canonical labelings of graphs. http://www.tcs.hut.fi/Software/bliss/ (2012)
21. Kaibel, V.: Extended formulations in combinatorial optimization. Optima **85**, 2–7 (2011). (**Newsletter of the Mathematical Optimization Society**)
22. Kaibel, V., Weltge, S.: Lower bounds on the sizes of integer programs without additional variables. Math. Program. **154**(1–2, Ser. B), 407–425 (2015)
23. Kurz, S., Napel, S.: Dimension of the Lisbon voting rules in the EU council: a challenge and new world record. Optim. Lett. **10**, 1245–1256 (2016)
24. Manwani, N., Sastry, P.S.: Learning polyhedral classifiers using logistic function. In: Sugiyama, M., Yang, Q. (eds.) Proceedings of 2nd Asian Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 13, pp. 17–30. PMLR, Tokyo, Japan (2010). https://proceedings.mlr.press/v13/manwani10a.html
25. Orsenigo, C., Vercellis, C.: Accurately learning from few examples with a polyhedral classifier. Comput. Optim. Appl. **38**, 235–247 (2007)
26. Ryan, D., Foster, B.: An integer programming approach to scheduling. In: Wren, A. (ed.) Computer scheduling of public transport: Urban passenger vehicle and crew scheduling. North-Holland, pp. 269–280 (1981)
27. Sankaran, J.K.: A note on resolving infeasibility in linear programs by constraint relaxation. Oper. Res. Lett. **13**(1), 19–20 (1993). https://doi.org/10.1016/0167-6377(93)90079-V
28. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to simon, present, lblock, des(l) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014, pp. 158–178. Springer, Berlin Heidelberg (2014)
29. Taylor, A.D., Pacelli, A.M.: Mathematics and Politics: Strategy, Voting, Power and Proof, 2nd edn. Springer, New York (2008)
30. Udovenko, A.: Milp modeling of boolean functions by minimum number of inequalities. Cryptology ePrint Archive, Report 2021/1099. https://ia.cr/2021/1099 (2021)
31. van der Hulst, R.: A Branch-Price-and-Cut Algorithm for Graph Coloring. https://essay.utwente.nl/88263/ (2021)
32. Weltge, S.: Sizes of Linear Descriptions in Combinatorial Optimization. Ph.D. thesis, Otto-von-Guericke-Universität Magdeburg. https://doi.org/10.25673/4350 (2015)