MILP modeling of Boolean functions by minimum number of inequalities

Aleksei Udovenko

CryptoExperts, Paris, France
aleksei@affine.group

August 25, 2021

Abstract. This work presents techniques for modeling Boolean functions by mixed-integer linear inequalities (MILP) on binary variables in-place (without auxiliary variables), reaching *minimum possible* number of inequalities for small functions and providing meaningful lower bounds on the number of inequalities when reaching the minimum is infeasible. While the minimum number of inequalities does not directly translate to best performance in MILP applications, it nonetheless provides a useful benchmark. We remark that our framework is heuristic and relies on SAT solvers and MILP optimization and so its feasibility is limited.

Keywords: MILP · Modeling · Boolean functions · S-boxes · DDT

1 Introduction

Mixed integer linear programming (MILP/MIP/IP) is a powerful form of expressing optimization and search problems. MILP systems consist of linear inequalities and a linear objective function, subject to minimization or maximization. In contrast to the classic linear programming (LP), MILP allows restricting a subset of variables to *integers*. This greatly expands the set of constraints that can be encoded. In particular, any relation on a given set of *binary* (0/1) variables can be expressed by a set of linear inequalities. Although a general such encoding would have an exponential number of inequalities (in the number of variables), more complex constraints can be constructed by using encodings of smaller parts as building blocks. For example, any Boolean circuit can be implemented by introducing binary variables for each intermediate gate and encoding the relation of its inputs and outputs. In this work, we focus on in-place MILP encoding of relations over a relatively small number of binary variables. Since any such relation defines a Boolean function, this process is often referred to as MILP modeling of a Boolean function. We remark that the linear programming aspect of MILP (i.e., maximization or minimization of an objective function) is irrelevant for the purposes of this work.

Due to its expressiveness, MILP has numerous applications in many fields. In cryptography, MILP is actively used for symmetric cryptanalysis. Typical applications include search of differential/linear/division trails through a given primitive [SHW+14, XZBL16, AST+17, BC20]. The original idea of utilizing such modeling for cryptanalysis is attributed to Mouha, Wang, Gu and Preneel [MWGP12]. A variety of optimization software can be used to solve hard MILP problems: commercial, such as Gurobi, CPLEX, and open source, such as SCIP, GLPK.

In this work, we consider the problem of constructing in-place MILP models with the *minimum possible* number of inequalities.

Problem 0. *Given a set $\mathcal{G} \subseteq \{0,1\}^n$, find a system S of linear inequalities with rational coefficients involving only variables $x_1, \ldots, x_n \in \{0,1\}$, such that the solution set of S is exactly $\mathcal{G}$.*

Related work On the theoretical side, this problem was considered already in 1975 by Jeroslow [Jer75], who gave a tight upper bound of 2^{n-1} inequalities to separate a subset of the hypercube $\{0,1\}^n$ from the rest of the hypercube (with examples of sets for which 2^{n-1} is the minimum possible number of inequalities). Jeroslow called this number the *index* of the set $\mathcal{G} \subseteq \{0,1\}^n$, denoted $ind(\mathcal{G})$. Later, Megiddo [Meg88] proved that deciding a possibility of strict separation of two given sets $\mathcal{G}, \mathcal{B} \subset \mathbb{Z}^n$ by k inequalities[1] is NP-complete even when either k or n is fixed (and is solvable in polynomial time if both are fixed). This more general case (with variables over $\mathbb{Z}$) was considered recently by Kaibel and Weltge [KW14, KW15, Wel15]. They call the minimum number of inequalities to model a set $\mathcal{G} \subset \mathbb{Z}^n$ its *relaxation complexity*, denoted $rc(\mathcal{G})$. Clearly, $ind(\mathcal{G}) \leq rc(\mathcal{G}) \leq ind(\mathcal{G}) + 2n$, where $2n$ stands for the inequalities $0 \leq x_i \leq 1$. However, Weltge also showed that the hypercube $\{0,1\}^n$ has a relaxation in the form of a simplex, reducing the extra $2n$ inequalities to $n+1$. Very recently, Averkov and Schymura [AS21] studied computability of $rc(\mathcal{G})$ and its relation to the rational relaxation complexity $rc_{\mathbb{Q}}(\mathcal{G})$ (i.e., relaxation complexity using rational inequalities). They proposed a direct MILP formulation for the problem of separating two given sets $\mathcal{G}, \mathcal{B} \subset \mathbb{Z}^n$ by minimum number of inequalities (denoted $rc(\mathcal{G}, \mathcal{B})$). Averkov, Hojny and Schymura [AHS21] focused on computational complexity aspects of the relaxational complexity. Besides theoretical results on computability of rc and $rc_{\mathbb{Q}}$, and a weakly polynomial-time algorithm for the two-dimensional case, they provided improved MILP formulations for the separation

[1] The roles of the sets $\mathcal{G}, \mathcal{B}$ are not symmetric: $\mathcal{G}$ must fully satisfy each of the inequalities, whereas for each point of $\mathcal{B}$ there must exist an inequality that it does not satisfy.

problem and performed experiments for up to 4-dimensional simplexes and 5-dimensional crosspolytopes.

On the more practical side, recently, Boura and Coggia [BC20] developed powerful *heuristic* techniques to construct compact MILP models of arbitrary Boolean functions, targeting applications in cryptanalysis. Their methods produce models with significantly smaller numbers of inequalities than by previous methods [SHW+14, AST+17]. A natural question is how far can these numbers be further decreased. In a work concurrent to ours, Yao [Sun21] independently proposed techniques for optimal MILP modeling, based on the so-called "SuperBalls".

An important remark is that the number of inequalities is not a precise criteria relevant for performance of MILP models in applications. For examples, Sasaki and Todo [ST17] described experiments when adding more inequalities improved the performance. Nonetheless, Boura and Coggia [BC20] suggest that the number of inequalities is still a useful benchmark as the smaller number allows more freedom for experiments on improving performance by adding more inequalities. Furthermore, development of new methods to generate inequalities (as we do in this work) further enriches the toolkit for such experiments. Kaibel and Weltge [KW14, KW15] also suggest practical relevance in terms of modeling simplicity. Finally, the problem of finding smallest MILP models can be of independent theoretical interest.

We mention another modeling paradigm which is relevant for our work and which can be seen as a special case of MILP: modeling by CNF (conjunctive normal form) formulas, often referred to simply as the satisfiability problem (SAT). A large number of SAT-solvers is available (Kissat, CaDiCaL, CryptoMiniSat, MiniSat, Glucose, Lingeling, etc.). CNF models are also used naturally for cryptanalysis [AST+17, SWW21]. For CNF models, the well known Quine-McCluskey algorithm [Qui52, Qui55, McC56] allows to construct CNF (or DNF) models with the *minimum possible* number of clauses for arbitrary Boolean functions. It is a standard algorithm for logic minimization of small Boolean functions, whereas heuristic alternatives such as Espresso [BSMH84] are used when the application of the Quine-McCluskey algorithm is not feasible.

Our contribution In this work, we fill the gap for MILP modeling and provide *practical* techniques for finding *optimal* (in the number of inequalities) models on binary variables. In other words, we provide an analogue of the Quine-McCluskey algorithm - which finds smallest CNF models - for finding smallest MILP models on binary variables. Our approach is well feasible for up to 10-bit non-sparse Boolean functions (e.g. the set of valid differential transitions of 5-bit S-boxes) and further provides suboptimal results for larger functions together with meaningful lower bounds. We benchmark the approach on the DDT[2] support of S-boxes from symmetric ciphers to compare with results reported by Boura and Coggia [BC20]. The summary of the results is given in Table 1.

Our algorithms are implemented as a set of Python packages, including automated tools. The implementation will be made publicly available.

Overview of techniques A standard heuristic approach for solving Problem 0 consists of two steps: (1) generating a (large) set of constraints (inequalities) that jointly model the target set and (2) selecting a small subset of the generated set that is sufficient for the model.

For example, the Quine-McCluskey algorithms performs the first step optimally and generates an exhaustive set of CNF/DNF clauses. However, for MILP modeling, only heuristic practical methods were proposed: in [BC20] the inequalities are generated in the

[2]Difference distribution table (DDT) - table characterizing differential transitions through an S-box (a vectorial Boolean function).

form of (distorted) balls; in [AHS21] all subsets of bad points are enumerated which leads to impractical complexity.

The second step is an instance of the NP-complete SetCover problem, and can be solved either heuristically by a greedy method or by using a MILP formulation of the problem, which can yield a guaranteed optimal solution if feasible.

Our main improvements cover the first step. We aim to compute all *maximal* (by inclusion) sets of points that can be removed by a single inequality (similar to the Quine-McCluskey algorithm for CNF/DNF clauses). These sets form an exhaustive set of inequalities sufficient to yield an optimal solution if solving the second step is feasible. To achieve the goal, we utilize techniques for learning *monotone* Boolean functions. While there exists a quasi-polynomial time incremental algorithm for enumerating such maximal sets, its practical efficiency is questionable; we resort to use a SAT-solver instead, which yields a very good performance.

Besides the aforementioned hardness of *enumerating* the maximal sets, another difficulty is that the *number* of these maximal sets can, in theory, be very large. To counter this, we split the problem into 2^n monotone subproblems characterized by the direction in the n-dimensional space, or alternatively by fixed signs of variables in the considered inequalities. This reduction significantly eases the difficulty of the maximal set enumeration. Another advantage is the natural parallelism of this method. Finally, we include a technique that allows to identify maximal such sets without pairwise comparisons.

Table 1: Summary of our new models and comparison to previous results. Bold numbers denote the best known results. Single number in our bounds means optimal result. The results were obtained using the CaDiCaL SAT-solver and the Gurobi optimizer.

function	# zeroes	our bounds	prev. UB [BC20]
	4-bit S-boxes / 8-bit functions		
Present-DDT	159 (62%)	**16**	17
Klein-DDT	150 (59%)	**18**	19
Twine-DDT	150 (59%)	**19**	**19**
Prince-DDT	150 (59%)	**18**	19
Piccolo-DDT	159 (62%)	**14**	16
MIBS-DDT	150 (59%)	**20**	**20**
Midori-S0-DDT	159 (62%)	**16**	**16**
Midori-S1-DDT	150 (59%)	**20**	**20**
Rectangle-DDT	159 (62%)	**15**	17
Skinny64-DDT	159 (62%)	**14**	16
GIFT-DDT	156 (61%)	**16**	17
Pride-DDT	159 (62%)	**16**	**16**
	5-bit S-boxes / 10-bit functions		
Ascon-DDT	707 (69%)	**27**	32
FIDES-5-DDT	527 (51%)	**57**	64
SC2000-5-DDT	527 (51%)	**60**	66
	6-bit S-boxes / 12-bit functions		
APN-6-DDT	2079 (51%)	**145**	167
FIDES-6-DDT	2079 (51%)	**162-166**	180
SC2000-6-DDT	2142 (52%)	**188-205**	218
	8-bit S-boxes / 16-bit functions		
AES-DDT	33150 (51%)	**2008−2699**	2882
Skinny-DDT	54067 (82%)	not feasible	**302**

Relation to the method of Yao [Sun21] Concurrently and independently of our work, Yao proposed techniques for optimal MILP modeling of Boolean functions, based on the so-called "SuperBalls". The high-level approach of Yao is similar to ours: maximal sets of points that can be removed by a single inequality are enumerated per each sign vector (which in Yao's work is defined by the "center of a region"). However, the enumeration procedure is different: we utilize the techniques of monotone set learning, while Yao formulates and solves the problem using MILP optimization.

2 Preliminaries

Boolean operations AND,OR,XOR,NOT denoted respectively by $\wedge, \vee, \oplus, \neg$ can be applied to bits or bitwise to bit-vectors. We use $\underline{1} \in \mathbb{F}_2^n$ (resp. $\underline{0}$) to denote the all-one (resp. all-zero) vector of dimension depending on the context. The unit vectors $e_j \in \mathbb{F}_2^n, 0 \leq j < n$, are vectors with j-th coordinate equal to 1 and all other coordinates equal to 0. We write $\neg x \coloneqq x \oplus \underline{1}$ and $\neg X \coloneqq \{\neg x \mid x \in X\}$, $X \subseteq \mathbb{F}_2^n$, to disambiguate from the set complement $\overline{X} := \{y \in \mathbb{F}_2^n \mid y \notin X\}$. The support of a vector $x \in K^n$ is the set $\mathrm{Supp}\,(x) = \{i \in \{0, \ldots, n-1\} \mid x_i \neq 0\}$ (K may be one of $\mathbb{F}_2^n$, $\mathbb{Z}$ or $\mathbb{R}$).

2.1 Partial order and monotone Boolean functions

We use the product partial order on vectors over $\mathbb{F}_2$ (or, generally, over $\{-d, \ldots, d\}$ for some positive integer d) which is, for $x, y \in \mathbb{F}_2^n$, $x \preceq y$ if and only if $x_i \leq y_i$ for all $i \in \{0, \ldots, n-1\}$. We write $x \prec y$ if $x \preceq y$ and $x \neq y$.

Definition 1 (Lower/upper sets/closures)**.** The *lower closure* of a set $X \subseteq \mathbb{F}_2^n$, denoted by $\downarrow X$, is the set of all $u \in \mathbb{F}_2^n$ with $u \preceq x$ for some $x \in X$:

$$\downarrow X \coloneqq \{u \in \mathbb{F}_2^n \mid \exists x \in X : u \preceq x\} = \bigcup_{x \in X} \downarrow x \coloneqq \bigcup_{x \in X} \{u \in \mathbb{F}_2^n \mid u \preceq x\}\,.$$

The *upper closure* of a set $X \subseteq \mathbb{F}_2^n$, denoted by $\uparrow X$, is the set of all $u \in \mathbb{F}_2^n$ with $x \preceq u$ for some $x \in X$:

$$\uparrow X \coloneqq \{u \in \mathbb{F}_2^n \mid \exists x \in X : u \succeq x\} = \bigcup_{x \in X} \uparrow x \coloneqq \bigcup_{x \in X} \{u \in \mathbb{F}_2^n \mid u \succeq x\}\,.$$

A set X is an *upper set* (resp. *lower set)*) if it is the upper (resp. lower) closure of itself. A lower (resp. upper) set is called *principal* if it is the closure of a single vector.

Remark 1. A useful interpretation is as follows. For each vector in X, upper closure converts positions with value 0 into a wildcard, whereas lower closure converts positions with value 1 to a wildcard.

Example 1.

$$\downarrow \{(1,1,0),(0,0,1)\} = \{(*,*,0),(0,0,*)\} = \{(0,0,0),(0,1,0),(1,0,0),(1,1,0),(0,0,1)\}\,,$$
$$\uparrow \{(1,1,0),(0,0,1)\} = \{(1,1,*),(*,*,1)\} = \{(0,0,1),(0,1,1),(1,0,1),(1,1,0),(1,1,1)\}\,.$$

Upper/lower sets correspond to supports of monotone Boolean functions.

Definition 2. A Boolean function is *monotone increasing* (resp. *monotone decreasing*) if its support is an upper set (resp. lower set).

By default, we will assume increasing monotone Boolean functions.

Proposition 1. *The complement of a lower set is an upper set and vice versa.*

Upper/lower sets (and thus, supports of monotone Boolean functions) are characterized by their *extreme* (also called *prime*) elements.

Definition 3 (Min-/max-set)**.** The *max-set* of a set $X \subseteq \mathbb{F}_2^n$, denoted by $\mathrm{MaxSet}(X)$, is the set of all maximal elements in X:

$$\mathrm{MaxSet}(X) \coloneqq \{u \in X \mid \nexists x \in X : x \succ u\}.$$

The *min-set* of a set $X \subseteq \mathbb{F}_2^n$, denoted by $\mathrm{MinSet}(X)$, is the set of all minimal elements in X:

$$\mathrm{MinSet}(X) \coloneqq \{u \in X \mid \nexists x \in X : x \prec u\}.$$

Often, min-/max-set is compact compared to the size of the full lower/upper set (often called the *volume*). However, there exist lower/upper sets with exponentially large max-/min-sets. For example, all $2n$-bit vectors with weight n form a max-/min-set of size $\binom{2n}{n} = O(2^n/\sqrt{n})$. Sperner [Spe28] proved that this is in fact an upper bound on the size of an antichain and so on the size of a max-/min-set.

Finally, we remark that a lower set having a small max-set does not guarantee small complementary upper set. Indeed, a standard example [Ang88] is the lower set spanned by the n-element set

$$L \coloneqq \{e_0 \vee e_1, e_2 \vee e_3, \ldots, e_{2n-2} \vee e_{2n-1}\} \subseteq \mathbb{F}_2^{2n},$$

which has the complementary upper set spanned by the 2^n-element set (note the exponential increase)

$$U \coloneqq \left\{u \in \mathbb{F}_2^{2n} \mid u_0 \oplus u_1 = 1, \ldots, u_{2n-2} \oplus u_{2n-1} = 1\right\}.$$

2.2 CNF/DNF models

A *CNF (conjunctive normal form)* formula over variables $x_0, \ldots, x_{n-1} \in \mathbb{F}_2$ has form

$$\bigwedge_{i \in \{0,\ldots,k-1\}} \bigvee_{j \in I_i} (x_j \oplus c_{i,j}),$$

where the size k of the formula is a non-negative integer, $I_i \subseteq \{0, \ldots, n-1\}$ for all i, $c_{i,j} \in \mathbb{F}_2$ for all i, j.

A *DNF (disjunctive normal form)* formula has form

$$\bigvee_{i \in \{0,\ldots,k-1\}} \bigwedge_{j \in I_i} (x_j \oplus c_{i,j}),$$

where the size k of the formula is a non-negative integer, $I_i \subseteq \{0, \ldots, n-1\}$ for all i, $c_{i,j} \in \mathbb{F}_2$ for all i, j.

An expression of the form $(x_j \oplus c_{i,j})$ is called a *positive literal* if $c_{i,j} = 0$ (the literal x_j) and a *negative literal* otherwise (the literal $\neg x_j$).

Monotone Boolean functions can be described by CNF (resp. DNF) formulas with only positive literals, where each clause corresponds to a maximal zero (resp. a minimal one) of the function. Such formulas have minimum possible size.

2.3 MILP models

MILP models are systems of linear *inequalities* $\sum_{i=0}^{n-1} a_i x_i \geq c$ with real coefficients a_i, c over real and/or integer variables x_i. Similarly to clauses in a CNF formula, inequalities in a MILP system can be also seen as connected by a conjunction (AND). Inequalities are more expressive due to the use of arbitrary integer/real coefficients. In particular, any CNF clause $\bigvee_{i \in I} x_i \vee \bigvee_{j \in J} \neg x_j$ is equivalent to the inequality $\sum_{i \in I} x_i + \sum_{j \in J} (1 - x_j) \geq 1$ on the binary variables x_i. Therefore, CNF models form a subset of MILP models.

3 Related work on CNF and MILP modeling

We introduce the following notation. From now on, we assume a set $\mathcal{G} \subseteq \mathbb{F}_2^n$ of "good" points and a set $\mathcal{B} \subseteq \overline{\mathcal{G}}$ of "bad" points are fixed together with a modeling system (CNF or MILP). Note that we allow the so-called "don't care" points from $\overline{\mathcal{G} \cup \mathcal{B}}$ similarly to the Quine-McCluskey algorithm. For these points, it is not required whether produced models must accept or reject them.

Definition 4. A subset $B \subseteq \mathcal{B}$ is said to be *1-removable*, if there exists a constraint C such that all vectors from $\mathcal{G}$ satisfy C and no vectors from B satisfy C.

The standard approach for CNF/MILP modeling [Qui55, McC56, SHW+14], as reformulated and summarized by Boura and Coggia [BC20], requires to solve two sub-problems.

Problem 1. *Given the two sets $\mathcal{G} \subseteq \mathbb{F}_2^n, \mathcal{B} \subseteq \overline{\mathcal{G}}$, find a (large and possibly redundant) set of constraints that jointly remove all points from $\mathcal{B}$ while keeping all points from $\mathcal{G}$. In other words, find a cover of $\mathcal{B}$ by 1-removable sets.*

Problem 2. *Given the two sets $\mathcal{G}, \mathcal{B}$ and a set S of constraints obtained by solving Problem 1, find a subset of S of (close to) minimum size such that it still defines a correct model for $\mathcal{G}, \mathcal{B}$.*

We briefly recall known approaches for the two problems from the literature.

3.1 Problem 1 - generating constraints

There exist two main approaches for solving Problem 1.

P1-hull Using H-representation of the convex hull of $\mathcal{G}$ [SHW+14] (as a subset of $\mathbb{R}^n$). The convex hull of any subset of $\{0,1\}^n$ does not contain any other point from $\{0,1\}^n$. The H-representation of the convex hull consists of linear inequalities and/or linear equations. Conversion from the vertex representation to the H-representation is computationally intensive[3] and is typically feasible only for dimensions $n \leq 16$, depending on the structure of the given set.

P1-logic The second approach is based on logical condition modeling, introduced for cryptographic applications in [AST+17]. In other words, a CNF formula is generated first and then converted to a MILP system. The authors propose to use the standard in hardware synthesis Quine-McCluskey [Qui55, McC56] algorithm to find a CNF representation with the minimum number of clauses. The algorithm consists of two steps which directly map to Problem 1 and Problem 2 that we outlined. The most interesting for us is step 1 of the algorithm, which finds all the so called *prime implicants* - minimal CNF clauses. Boura and Coggia reformulated this step as searching for all $a, u \in \mathbb{F}_2^n, a \wedge u = \underline{0}$, such that $(a \oplus \downarrow u) \subseteq \mathcal{B}$ is maximal (by inclusion).

3.2 Problem 2 - choosing a small subset of constraints

Problem 2 requires to find a small or even the smallest subset of constraints generated from solving Problem 1, while requiring that they still jointly model the target pair $\mathcal{G}, \mathcal{B}$. This is an instance of the NP-hard SetCover/HittingSet problem. Too approaches exist in the literature:

[3] In the cryptanalysis literature [SHW+14], the convex hull is usually computed using the SageMath software [The21] which simply delegates the problem to the backend - Parma Polyhedra Library (PPL) [BHZ08].

P2-greedy Classic greedy approach, used in [SHW+14]. The idea is to iteratively include a constraint that removes the largest number of remaining "bad" points. Even for the inequalities from P1-hull this approach removes a large number of inequalities. This is because facets of the convex hull are often redundant in terms of separating the integer points of the convex-hull from the outer points.

It is well-known [Joh74, Lov75] that the greedy approach produces (in the worst case) an $(1 + \ln n)$-approximate solution to the SetCover problem, where n is the number of points to cover, and this ratio is optimal up to lower-order terms [LY94, Fei95] (unless NP has subexponential time algorithms, which is widely believed to be false). For more fine-grained bounds, see Slavik [Sla97].

P2-milp Sasaki and Todo [ST17] proposed to model the problem by MILP and use modern optimization software (such as commercial Gurobi [GO21] or open-source alternatives SCIP [GAB+20], GLPK [Mak17]) to solve this problem. This approach was also used more recently in [AHS21] for computations of the relaxation complexity.

This is a rather natural approach for an NP-hard problem. The idea is a s follows: a variable is introduced per each inequality and a constraint is added per each bad point requiring that at least one inequality removing this point is chosen.

Notably, the size of a greedy solution to the problem is within the same factor $(1 + \ln n)$ larger than the solution to the LP *relaxation* of the constructed MILP problem (which is called a *fractional set cover*) [Lov75].

The second approach is more preferable as optimization software is expected to perform better than the greedy algorithm; in fact, it is often feasible to obtain an optimal set cover solution (among constraints generated at step 1). However, for large instances with thousands of bad points and millions of constraints, MILP software may be very slow in finding a solution that beats the greedy algorithm. In the next subsection, we suggest a hybrid approach as a compromise.

3.3 On the optimality and lower bounds

A set S of constraints that is heuristically generated when solving Problem 1 may not contain a globally optimal solution. In this case, even solving the set cover (Problem 2) perfectly would not guarantee global optimality. Neither possible it is to derive a meaningful lower bound on the minimum size of a model from heuristic systems. For example, the P1-hull method is heuristic; the P1-logic method is optimal (with respect to CNF/DNF models) as it is exhaustive in maximal 1-removable groups. We introduce a definition to distinguish these two classes of solutions.

Definition 5. A set S of constraints modeling $\mathcal{G}, \mathcal{B}$ is said to be *complete* (in a fixed constraint model), if it contains all maximal 1-removable groups.

Remark 2. A set of constraints that includes an optimally-sized subset does not necessarily have to be complete. However, having a complete system as the input to Problem 2 is the only known way to ensure optimality.

Importantly, a complete set of constraints allows to obtain nontrivial lower bounds on the number of inequalities even when it not feasible to completely solve the corresponding set-cover problem. For example, we can assume that all inequalities remove pairwise disjoint groups of bad points. If the sum of sizes of t largest such groups is less than the number $|\mathcal{B}|$ of bad points, then at least $t + 1$ inequalities are need to cover all bad points. This bound applies to the original system, since without the disjointness assumption the number of required inequalities may only increase. However, such naive approach would provide a rather weak lower bound.

Greedy method (P2-greedy) The greedy method (P2-greedy) reduces the complexity of solving the set cover problem dramatically. Although the optimality of modeling is lost, greedy method still provides a lower bound: it is known that it solves the set cover problem with approximation factor at most $H(|\mathcal{B}|) \leq 1 + \ln |\mathcal{B}|$ (see more tight analysis by Slavik [Sla97]).

(MI)LP-based method (P2-milp) A more generic approach is to use the LP relaxation of the problem (i.e., dropping the integrality constraints) and solve it using e.g. the simplex method. Since constraints are removed, such a solution provides a lower bound on the objective of the original MILP instance. In fact, this technique is at the core of modern MILP solvers and they automatically provide such or an improved lower bound as the first step of dealing with a given problem. It is known that the LP-re

Full-MILP method (as opposed to LP-relaxation-based method) allows to find optimal solution, or, at least, provide lower and upper bounds that are better than given by the approximation factor of the greedy method. This comes however at the cost of increased time complexity which renders large systems infeasible to solve with reasonable resources.

Greedy-hybrid method For large systems, we suggest a hybrid approach between the P2-greedy and the P2-milp methods. The idea is to run the greedy algorithm to find first $t \geq 0$ inequalities greedily and then apply the MILP search method to cover the remaining bad points optimally. This relaxes the pressure on the MILP solver due to significant reduction of the system's size and often allows to quickly obtain good approximate solutions.

Although this method also allows to determine a lower bound (the optimal solution to the subsystem minus at most t extra inequalities), the best lower bounds in our experiments are obtained from running a MILP solver on the full system.

4 General technique for optimal modeling

In this section, we present our general approach for optimal MILP modeling. We aim to solve Problem 1 and generate a complete pool of inequalities, ensuring in particular that it contains a size-optimal subset-solution. While Problem 2 may stay the main obstacle to obtain a globally *optimal* solution, a complete pool would allow to obtain close-to-optimal solution together with a known bound on the optimality.

We start by describing a simple tool, which allows, for a set $B \subset \mathcal{B}$, to determine if it is 1-removable and to construct the respective inequality if it exists. Then, we recall that all 1-removable groups $B \subset \mathcal{B}$ form a structure of a monotone Boolean function. Finally, we focus on the problem of learning this structure using the oracle provided by our first tool. We revisit known approaches to this problem, and propose a way to implement them.

The approach is essentially laid out by series of works on monotone Boolean functions [Zhu78, Val84, Gai84, BI95, FK96, GK99, BEG+02, TT02, TT09], although we are not aware of a relevant formulation of the application to the MILP modeling in the literature. The closest studied problem is the problem of finding all maximal consistent subsystems of a given inconsistent system of inequalities. In Subsection 4.1, we provide the missing link between the two problems and in the rest of the section (Subsection 4.2, Subsection 4.3, Subsection 4.4, Subsection 4.5) we review and assemble the tools to solve the problem.

4.1 (In)consistency oracle

In this subsection, we show how to decide if a set $B \subseteq \mathcal{B}$ is 1-removable by a MILP inequality. This is a well-known technique in threshold logic / linear separability [Der65], see also Gruzling's thesis [Gru07] for a survey.

Recall the defined set $\mathcal{G} \subseteq \mathbb{F}_2^n$ of "good points" and the set $\mathcal{B} \subseteq \overline{\mathcal{G}}$ of "bad points". The problem is, for a given subset $B \subseteq \mathcal{B}$, to determine if B is 1-removable.

Let $a \in \mathbb{R}^n, c \in \mathbb{R}$ be real variables. Consider the inequality $\langle a, y \rangle \geq c$, where $y \in \mathbb{F}_2^n$ is a vector of free variables. We aim to define constraints on a, c which imply that this inequality separates $\mathcal{G}$ from B. For any $x \in \mathcal{G}$, the requirement forms the inequality $\langle a, x \rangle \geq c$; for any $x' \in B$ the requirement forms the inequality $\langle a, x' \rangle < c$. All these inequalities form a system which is consistent if and only if there exists an inequality separating B from $\mathcal{G}$. Moreover, any solution of the inequality system yields such an inequality. Note that typical MILP systems do not allow strict inequalities. This issue can be solved by replacing the inequality $\langle a, x' \rangle < c$ with the inequality $\langle a, x' \rangle \leq c - 1$. The following proposition formalizes the construction.

Proposition 2. *Let $\mathcal{G} \subseteq \mathbb{F}_2^n, B \subseteq \overline{\mathcal{G}}$. Consider the inequality system*

$$\begin{cases} \langle a, v \rangle \geq c, & \text{for each } v \in \mathcal{G}, \\ \langle a, v' \rangle \leq c - 1, & \text{for each } v' \in B, \end{cases} , \tag{1}$$

where $a \in \mathbb{R}^n, c \in \mathbb{R}$ are variables. This system is consistent if and only if there exists $a' \in \mathbb{R}^n, c' \in \mathbb{R}$ such that the inequality $\langle a', x \rangle \geq c'$ with the variable $x \in \{0,1\}^n$ separates $\mathcal{G}$ from B.

Proof. ($\Rightarrow$) Obvious by construction of the system. ($\Leftarrow$) The only subtle points are the generality of the form $\langle a, x \rangle \geq c$ and the gap of length 1 (in $\langle a, x \rangle \leq c - 1$). First, an inequality of the form $\langle a, x \rangle \leq c$ can be always rewritten as $\langle -a, x \rangle \geq -c$. Second, if there exists an inequality $\langle a, x \rangle \geq c$ satisfying all of $\mathcal{G}$ and an inequality $\langle a, x \rangle < c$ satisfying all of B, then, due to finiteness of B, the supremum c' of $\langle a, x \rangle$ as x ranges over B satisfies $c' < c$. Then, letting $k = 1/(c - c')$, the inequality $\langle ka, x \rangle \geq kc$ satisfies all of $\mathcal{G}$ and the inequality $\langle ka, x \rangle \leq kc - 1$ satisfies all of B. □

There exist polynomial-time algorithms for solving a system of linear inequalities / linear programming over rationals, such as Karmarkar's algorithm [Kar84] (within the interior-point methodology due to Dikin [Dik67]), the ellipsoid method by Khachiyan [Kha79]. On practice, the simplex method by Dantzig [Dan90] performs better, although it does not guarantee worst-case polynomial time. An implementation is available as a part of the open-source MILP solver GLPK [Mak17], also shipped as the default MILP solver in the SageMath [The21] computer algebra. In fact, most MILP solvers utilize the linear programming over $\mathbb{Q}^n$ (LP) as a sub-procedure when solving integer optimization problems (MILP).

4.2 Monotonicity of the inconsistency

We are interested to know which subsets of $\mathcal{B}$ are 1-removable. By Proposition 2, such subsets correspond to consistent subsystems of the (possibly inconsistent) full system of inequalities constructed for full $\mathcal{B}$.

A well known observation is that *inconsistency* is *monotonous*: adding a new inequality to an inconsistent subsystem can not make it consistent. It follows that all the consistent subsystems form a *lower set*, and all the inconsistent systems form the complementary *upper set*. In our terminology, all the 1-removable subsets of $\mathcal{B}$ form a lower set and all the non-1-removable subsets of $\mathcal{B}$ form the complementary upper set. This relies on that Definition 4 does not require points from $\mathcal{B} \setminus B$ for B to be 1-removable.

Proposition 3. *The set of 1-removable subsets of $\mathcal{B}$ forms a lower set.*

A lower set (or an upper set) is equivalently described by a monotone Boolean function. In fact, the equivalence between the set of inconsistent subsystems and a monotone Boolean

function was mentioned by Zhuravlev already in 1978 [Zhu78, Thm 19]. More concretely, this connection was used to show an incremental quasi-polynomial time algorithm for identifying maximal consistent subsystems by Gurvich and Khachiyan [GK99], which we will discuss later in Subsection 4.4.

We arrive at the problem of learning a lower set, alternatively called "inference of a monotone Boolean function" in the literature [TT02]. Importantly, such a structure is characterized by its extreme elements: a lower set is characterized by the set of its *maximal* elements, i.e., by its max-set. It is easy to show that the complete set of maximal 1-removable subsets contains an optimal solution.

Proposition 4. *Let U be a set of 1-removable subsets covering $\mathcal{B}$. Then there exists a set U' of maximal 1-removable subsets covering $\mathcal{B}$ with $|U'| = |U|$.*

Proof. For any subset $u \in U, u \subseteq \mathcal{B}$ it is easy to obtain a maximal 1-removable subset $u' \subseteq \mathcal{B}$ by attempting to greedily add each element of $\mathcal{B}$ in an arbitrary order, while ensuring the 1-removable property. Clearly, U' composed of all such u' (one per each $u \in U$) covers $\mathcal{B}$ and has same size as U. □

Generally, by the Sperner's theorem, the size of a max-set can be very large: $\mathcal{O}(2^n/\sqrt{n})$. However, max-sets occurring in various applications (including ours) are much smaller. For example, the max-set of a random Boolean function is expected to be small, because a few vectors with large weight exclude a large combinatorial variety of vectors of smaller weights. A symmetric example with small weight vectors is the max-set of the maximal lower set contained in a random Boolean function. In fact, in our applications we successfully learn lower sets for n up to 700, with the max-set of size about 30,000 vectors and with the complementary min-set of size about 1,000,000 vectors.

4.3 Learning monotone Boolean functions

We now survey the literature on learning monotone Boolean functions. We start by stating the problem.

Problem 3. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an unknown increasing monotone Boolean function given as an oracle. Find a representation of f (such as the max-set of its zeroes) by calling f as few times as possible.*

Note that this classic statement does not cover computations besides oracle calls and this will become a practical issue that we analyze further.

One of the first studies was made by Hansel [Han66], who established the numbers of monotone Boolean functions and introduced a method to partition the hypercube $\{0,1\}^n$ into increasing chains satisfying certain properties; such partition leads to a learning algorithm that performs optimally based on the worst-case hardness. Later, in 1982, Sokolov [Sok82] showed how to construct such partition dynamically, improving memory complexity of the algorithm.

Torvik and Triantaphyllou [TT02] optimized average query complexity over all monotone Boolean functions. Their idea is to query a vector that minimizes the difference between the numbers of learned values of f for any of the two possible oracle answers. For example, for a chosen $v \in \mathbb{F}_2^n$, assume that $f(v) = 0$ implies $f(v') = 0$ for new n_0 vectors, and $f(v) = 1$ implies $f(v') = 1$ for new n_1 vectors. Then, the target difference is $|n_0 - n_1|$ and v should be chosen among all unknown vectors that minimizes this difference. The authors prove that such approach leads to optimal *average* query complexity and also experimentally show that it concretely improves over previous approaches. However, they admit that their approach is not feasible on practice for $n > 20$, due to the complexity of evaluating the criterion.

The work of most interest for us is that by Gainanov [Gai84]. It proposes algorithms optimizing the number of queries *with respect to the size of the border set*, that is, the max-set of zeroes and the min-set of ones of the target Boolean function. This measure crucially allows to separate instances occurring in various applications and worst-case instances having large optimal complexity. We describe the algorithm in more details using the lower/upper set terminology.

4.3.1 Gainanov's algorithm

Let $L \subseteq \mathbb{F}_2^n$ be the target lower set being learned, and let $U \coloneqq \overline{L}$ be the complementary upper set. We aim to learn elements from $\operatorname{MaxSet}(L)$ and from $\operatorname{MinSet}(U)$ by querying $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that $f(x) = 1$ if and only if $x \in U$. Assume that we have already learned $\widehat{L} \subseteq \operatorname{MaxSet}(L)$ and $\widehat{U} \subseteq \operatorname{MinSet}(U)$. We will need the following procedure.

Procedure LearnUp/LearnDown Assume that we are given $x \in \mathbb{F}_2^n$ such that $f(x)$ is not known (from $\widehat{L}$ and $\widehat{U}$), that is,

$$x \in \overline{\downarrow \widehat{L} \cup \uparrow \widehat{U}}.$$

Let us query $f(x)$ and without loss of generality assume that $f(x) = 0$. Then, in at most n queries, we can obtain a new element of $\operatorname{MaxSet}(L)$. Indeed, let us iterate over all positions $i \in \{0, \dots, n-1\}$. If $x_i = 0$, query $f(x \vee e_i)$. If $f(x \vee e_i) = 0$, update $x \leftarrow x \vee e_i$, otherwise continue without modifying x. It is not hard to see that after the n iterations the resulting x will be such that:

1. $f(x) = 0$, i.e., $x \in L$.
2. $x \in \operatorname{MaxSet}(L)$;
3. $x \notin \widehat{L}$.

The first property follows by construction. The second property follows from the fact that for any e_i with $x_i = 0$, $f(x_i \vee e_i) = 1$ (by monotonicity of f and the construction of x). The third property follows from the fact that an element of $\downarrow \widehat{L}$ could not be obtained by starting with an element from outside and flipping some coordinates from 0 to 1.

Remark 3. An analogous procedure similarly recovers a new element of $\operatorname{MinSet}(U)$ in case of $f(x) = 1$.

Remark 4. By iterating again over $\{0, \dots, n-1\}$ in a random order, we could possibly obtain more new elements of $\operatorname{MaxSet}(L)$. In fact, LearnUp($\underline{0}$) would return random elements of $\operatorname{MaxSet}(L)$ with higher probabilities given to elements of higher weight. This is a useful alternative to full provable exhaustion of $\operatorname{MaxSet}(L)$, as it does not require sampling an unknown vector, which turns out to be a hard problem, as we will see later.

The algorithm is simple: while there are vectors outside of $\downarrow \widehat{L}$ and $\uparrow \widehat{U}$, find any such vector x and query $f(x)$. Depending on the answer, apply the procedure LearnUp or LearnDown. This yields a new element of $\operatorname{MaxSet}(L)$ or $\operatorname{MinSet}(U)$ to be added to $\widehat{L}$ or $\widehat{U}$. The query complexity of such algorithm is $|\operatorname{MaxSet}(L)| + |\operatorname{MinSet}(U)|$ calls to LearnUp or LearnDown, each of which incurs at most n extra queries, leading to the total query complexity $n(|\operatorname{MaxSet}(L)| + |\operatorname{MinSet}(U)|)$.

The algorithm can be further improved by reducing cost for recovering elements either of $\operatorname{MaxSet}(L)$ or of $\operatorname{MinSet}(U)$. Indeed, if x is sampled from the *minimal* "unknown" vectors (e.g., from unknown vectors of minimum weight), it is guaranteed that $x \in \operatorname{MinSet}(U)$ in case $f(x) = 1$ and so the call to LearnDown is not needed. Alternatively, if x is sampled from the *maximal* "unknown" vectors (e.g., from unknown vectors of maximum weight), it

is guaranteed that $x \in \text{MaxSet}(L)$ in case $f(x) = 0$. This means that cost per element of $\text{MaxSet}(L)$ (or $\text{MinSet}(U)$) can be reduced to 1, while the other cost stays at most n.

In our applications, the lower set L would be typically much smaller than the complementary upper set U, and we expect there borders (extreme elements) follow the same imbalance. Therefore, we aim to sample x from "unknown" vectors of minimum weight. This would allow to obtain final cost of at most $n\,|\text{MaxSet}(L)| + |\text{MinSet}(U)|$ operations, plus an extra query for the edge cases like $L = \emptyset$, where $n+1$ queries need be done per one element (only possible for elements $\underline{0}$ and $\underline{1}$). This formulation is sketched in Algorithm 1.

Algorithm 1 Learn an unknown lower set $L \subseteq \mathbb{F}_2^n$ [Gai84]

Input: $f : \mathbb{F}_2^n \to \mathbb{F}_2$ such that $f(x) = 0$ if and only if $x \in L$
Output: $\text{MaxSet}(L), \text{MinSet}(\overline{L})$
Complexity: $n \cdot |\text{MaxSet}(L)| + \left|\text{MinSet}(\overline{L})\right| + 1$ queries to f

1: $\widehat{L} \leftarrow \emptyset$, $\widehat{U} \leftarrow \emptyset$
2: **while** $\downarrow \widehat{L} \cup \uparrow \widehat{U} \neq \mathbb{F}_2^n$ **do**
3: $\quad x \xleftarrow{\$} \mathbb{F}_2^n \setminus \left(\downarrow \widehat{L} \cup \uparrow \widehat{U}\right)$ with minimum weight
4: $\quad$ **if** $f(x) = 0$ **then**
5: $\quad\quad \widehat{L} \leftarrow \widehat{L} \cup \{\text{LearnUp}(x)\}$
6: $\quad$ **else**
7: $\quad\quad \widehat{U} \leftarrow \widehat{U} \cup \{x\}$
8: **return** $\widehat{L}, \widehat{U}$

4.4 Sampling unknown vectors

Unfortunately, methods of sampling vectors x with an unknown value of $f(x)$ were not proposed by Gainanov. Indeed, this was out of scope of the paper as the goal was only to minimize the number of oracle calls. While for small n, similarly to the methods of Hansel, Sokolov, Torvik and Triantaphyllou, this could be done in an exhaustive manner, for larger n this approach quickly becomes infeasible. We briefly survey known results about this problem.

Bioch and Ibaraki [BI95] studied the problem of interactive learning of a monotone Boolean function from the complexity perspective. They prove that this problem (which they call IDENTIFY) is polynomial-time equivalent to the problem (which they call EQ) of determining whether given sets $\widehat{L} \subseteq \mathbb{F}_2^n, \widehat{U} \subseteq \mathbb{F}_2^n$ are such that their respective lower and upper closures partition $\mathbb{F}_2^n$ (i.e., condition on line 2 in Algorithm 1). This problem is often formulated as the (monotone) DUAL problem.

Problem 4 (DUAL)**.** *Let $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ be monotone Boolean functions given by their irredundant monotone DNF formulas. Test whether f and g are dual, that is, whether $f(x) = \neg g(\neg x)$ for all $x \in \mathbb{F}_2^n$.*

This problem belongs to co-NP (the negative result is easily certified) and there exist no known polynomial-time algorithms. However, Fredman and Khachiyan [FK96] proposed a quasi-polynomial algorithm for DUAL running in time $\mathcal{O}(n^{4l(n)+\mathcal{O}(1)}) = \mathcal{O}(n^{o(\log n)})$, where $l = l(n)$ is such that $l^l = n$. Due to this, it is widely believed that the problem is not co-NP-complete. Importantly, the Fredman-Khachiyan algorithm also returns the proof in case the functions are not dual. When mapped to our setting, such a proof constitutes an unknown vector needed for learning (here we use the fact that the input lower/upper sets are non-intersecting).

The algorithm of Gurvich and Khachiyan [GK99] mentioned above consists in applying the Fredman-Khachiyan algorithm to obtain a complete quasi-polynomial solution for

learning maximal consistent subsystems and minimal inconsistent subsystems (which they call *convex programming*).

Several works [KBEG06, EMG08, Elb08, BM09, Sed18] studied implementations of the Fredman-Khachiyan algorithms. The implementations of [KBEG06][4] and [Sed18][5] are available online. We attempted to use the implementation of [KBEG06] and observed that it quickly becomes very slow or infeasible for values of $n \geq 100$ and the max-set of size ≥ 1000. The main source of complexity is that the current lower/upper sets are reduced per each guessed bit, which requires time quadratic in the size of the respective max-set/min-set. While this is not critical for the theoretical quasi-polynomial complexity, it becomes the bottleneck in practice.

To combat this issue, we resort to using SAT-solvers to solve this problem on practice. As we will show, for our purposes, modern SAT-solvers such as CaDiCaL [BFFH20] perform extremely well and in most cases the set-cover optimization stage remains the main difficulty even when relaxing the search to close-to-optimal results.

4.5 CNF model of sampling unknown vectors

We now briefly describe the natural CNF encoding of the problem. Let $\widehat{L}$ and $\widehat{U}$ be defined as above. Let $x \in \mathbb{F}_2^n$ be the vector of variables. For each $v \in \widehat{L}$ we add the clause

$$\bigvee_{i \in \{0,\dots,n-1\} | v_i = 0} x_i;$$

for each $u \in \widehat{U}$ we add the clause

$$\bigvee_{i \in \{0,\dots,n-1\} | u_i = 1} \neg x_i.$$

Clearly, each such clause excludes the principal lower set $\downarrow v$ and the principal upper set $\uparrow u$ respectively. The conjunction (AND) of all described clauses excludes $\downarrow \widehat{L}$ and $\uparrow \widehat{U}$ as required. In other words, x is a solution of the CNF formula if and only if x is outside of $\downarrow \widehat{L}$ and $\uparrow \widehat{U}$.

SAT-based minimization/maximization of unknown vectors As mentioned before and suggested in Algorithm 1, it is often beneficial for performance to sample minimal (or maximal) unknown vectors. Indeed, if the min-set of the learned upper set is expected to be larger, sampling minimal unknown vectors allows to avoid all calls to LearnDown. This requirement can be embedded in the CNF encoding using techniques for expressing Boolean cardinality constraints, for example, the unary sequential counter [Sin05]. It allows to constraint new variables v_i such that $v_i = 1$ if and only if $\sum_{x \in X} x \geq i$ for a chosen set X of variables. This in turn allows to sample unknown vectors in an order with non-decreasing (or non-increasing) weights, i.e., exhausting all unknown vectors of weight $0, 1, \dots, n$, implying the minimality (resp. maximality) of sampled vectors.

5 Modeling monotone sets

In this section, we explore the problem of modeling *monotone* sets / Boolean functions. This setting may be interesting for a range of applications and will be in particular useful for our improved general algorithm in Section 6.

[4] Available at https://github.com/VeraLiconaResearchGroup/MHSGenerationAlgorithms/tree/master/containers/fka-begk (written in C/C++)

[5] Available at https://github.com/WGS-TB/FK (written in MATLAB)

Notation Without loss of generality, we restrict to modeling an upper set. Similarly to the general setting, we allow "don't care" points. We denote the modeled upper set by $U \subseteq \mathbb{F}_2^n$ ("good" points) and the subset of the complementary lower set by $L \subseteq \overline{U}$ ("bad" points). Their extreme subsets are denoted by

$$\widehat{L} \coloneqq \mathrm{MaxSet}\left(L\right), \quad \widehat{U} \coloneqq \mathrm{MinSet}\left(U\right).$$

On the general non-compactness In 1983, Zuev and Trishin [ZT83] proved that there exist monotone Boolean functions that require at least $\binom{n}{\lfloor n/2 \rfloor}/n = \mathcal{O}(2^n/n^{1.5})$ linear inequalities to be specified. In other words, there exist upper sets that do not allow compact models. However, similarly to the lower set learning problem, instances arising in applications are often much "simpler" and in fact allow significant reduction in the number of inequalities compared to general subsets of $\{0,1\}^n$.

5.1 Exploiting monotonicity

Despite the negative result by Zuev and Trishin, the problem of modeling an upper set is arguably simpler and more structured than modeling an arbitrary Boolean function, i.e., of an arbitrary subset of the unit hypercube $\{0,1\}^n$. We start by defining monotone inequalities and showing that they can only model upper sets (monotone decreasing inequalities model lower sets, but we restrict to the increasing ones for simplicity).

Definition 6. An inequality $\langle a, x\rangle \geq c$, $a \in \mathbb{R}^n, c \in \mathbb{R}$ is said to be *monotone* if $a \in \mathbb{R}^n_{\geq 0}$.

Remark 5. The constant c has also to be nonnegative, otherwise the inequality trivially satisfies all points.

Proposition 5. *A monotone inequality models an upper set (removes a lower set).*

Proof. Trivially, switching x_i from zero to one can not decrease the value of $\langle a, x\rangle$ and so can not change a satisfying x to an unsatisfying x. □

The following lemma shows that *any* inequality removing some set $B \subseteq \mathbb{F}_2^n$ can be transformed into an inequality removing $\downarrow B$.

Lemma 1. *Let $\langle a, x\rangle \geq c, a \in \mathbb{R}^n, c \in \mathbb{R}$, be an inequality removing $B \subseteq \mathbb{F}_2^n$, i.e., $B = \{u \in \mathbb{F}_2^n \mid \langle a, u\rangle < c\}$. Let*

$$a' = (\max(0, a_i) \mid i \in \{0, \ldots, n-1\}) \in \mathbb{R}^n_{\geq 0}, \quad c' = c - \sum_{i:a_i<0} a_i.$$

Then, $\langle a', x\rangle \geq c'$ precisely removes $\downarrow B \subseteq \mathbb{F}_2^n$.

Proof. If $a \in \mathbb{R}^n_{\geq}$ then B must be a lower set by Proposition 5 and the lemma follows. Otherwise, assume without loss of generality that $a_0 < 0$ and other coordinates are nonnegative. Let $u \in \mathbb{F}_2^{n-1}$. The new inequality $\langle a', x\rangle \geq c'$ is constructed in such a way that the vector $(0||u)$ is rejected by the new inequality if and only if the vector $(1||u)$ is rejected by the original inequality; the acceptance of $(1||u)$ is unchanged. Indeed,

$$\begin{aligned}\langle a', (1||u)\rangle + a_0 = \langle a', (0||u)\rangle + a_0 = \langle (0, a_1, \ldots, a_{n-1}), (0||u)\rangle + a_0 = \\ = \langle (a_0, a_1, \ldots, a_{n-1}), (1||u)\rangle = \langle a, (1||u)\rangle .\end{aligned}$$

Consider 3 cases:

1. Both $\langle a, 0||u\rangle \geq c$ and $\langle a, 1||u\rangle \geq c$ hold or both do not gold. In this case, the inequality holds independently of x_0 and replacing a_0 by 0 does not change the satisfying set.

2. $\langle a, 0||u\rangle < c$ and $\langle a, 1||u\rangle \geq c$. This is impossible because $a_0 < 0$.
3. $\langle a, 0||u\rangle \geq c$ and $\langle a, 1||u\rangle < c$. By construction of a', c', the new inequality rejects a vector from $\downarrow B$ that was accepted by the original inequality.

By applying this change iteratively to each coordinate, we obtain an inequality that removes a subset of $\downarrow B$. Furthermore, this inequality must be monotone by construction and by Proposition 5 it must precisely remove full $\downarrow B$. □

Remark 6. Keeping original c would correspond to taking $(0||u)$ as the source vector. This would in turn correspond to taking *upper closure of the accepted vectors* (as opposed to the lower closure of the removed vectors), or, equivalently, reducing the removed set to the maximal lower set included in B.

Importantly, it follows that is sufficient to consider only monotone inequalities for optimal modeling.

Corollary 1. *There exist complete MILP models of any upper set (i.e., removing a lower set) containing only monotone inequalities.*

Another useful effect of monotonicity is that it is sufficient to separate extreme elements - the min-set of the modeled upper set and the max-set of the complementary lower set.

Proposition 6. *Let $Ax \geq C, A \in \mathbb{R}^{m\times n}_{\geq 0}, C \in \mathbb{R}^m_{\geq 0}$ be a system of inequalities. Then, $Ax \geq C$ accepts an upper set $U \in \mathbb{F}_2^n$ and rejects a lower set $L \subseteq \overline{U}$ if and only if both following conditions are true:*

1. *for all $x \in \mathrm{MinSet}(U)$ the system $Ax \geq C$ holds,*
2. *for all $x \in \mathrm{MaxSet}(L)$ the system $Ax \geq C$ does not hold.*

Proof. ($\Rightarrow$) is trivial. For ($\Leftarrow$), observe that $\langle a, x\rangle \leq \langle a, x'\rangle$ when $x \preceq x'$. This implies that if $\langle a, x\rangle \geq c$ for all $x \in \mathrm{MinSet}(U)$, then $\langle a, x'\rangle \geq c$ for all $x' \in \uparrow U$. Similarly, if $\langle a, x'\rangle < c$ for some $x' \in \mathrm{MaxSet}(L)$, then $\langle a, x\rangle < c$ for all $x \in \downarrow x'$. Since the lower set consists of the union of lower sets of single elements from its max-set, the proposition follows. □

5.2 Adapting the inconsistency oracle

The general inconsistency oracle from Subsection 4.1 can be easily adapted to the modeling based on the extreme elements. Recall that the oracle allowed to find an inequality (if it exists) that removes a given group of bad points. While it can be directly applied to the sets L and U ignoring their monotonicity, it is much more fruitful to apply it to the reduced sets $\widehat{L}$ and $\widehat{U}$. The only modification needed is to constraint the coefficients c and $a_i, 0 \leq i < n$ to be non-negative (whereas in the general version of the oracle they were allowed to be negative). Indeed, Proposition 6 and Corollary 1 guarantee that such an inequality must exist if any general inequality removing the given group exists.

Note that 1-removable subsets of $\widehat{L}$ form a lower set by themselves (by the monotonicity of inconsistency, exactly as in Subsection 4.2). Therefore, the general learning technique developed in Section 4 applies directly to $\widehat{L}$ and $\widehat{U}$. We conclude that the outlined approach allows to find optimal solutions, as shown by the following proposition.

Proposition 7. *Let S be a subset of 1-removable subsets covering L. There exists a subset S' of 1-removable subsets of $\widehat{L}$ by monotone inequalities with $|S'| \leq |S|$. Furthermore, these inequalities remove the whole L.*

Proof. Without loss of generality, by Lemma 1, we assume that subsets included in S are 1-removable by monotone inequalities. Consider those subsets that include at least one element from $\widehat{L}$. Since $\widehat{L} \subseteq L$, full $\widehat{L}$ must be covered. Since inequalities are monotone, it follows that full $\downarrow \widehat{L} \coloneqq L$ is removed. □

6 Improved technique for optimal modeling

In this section, we present an improved approach for learning all maximal 1-removable subsets of $\mathcal{B} \subseteq \overline{\mathcal{G}} \subseteq \mathbb{F}_2^n$ in the general case.

The key idea is to understand better what kinds of subsets of $\mathcal{B}$ can be 1-removable. The understanding is inspired by the reformulation of the Quine-McCluskey algorithm by Boura and Coggia, in which sets of the form $s \oplus \downarrow u$ are searched inside $\mathcal{B}$. More precisely, we reuse the idea to perform such a search separately per each candidate of s. In the case of inequalities, each such candidate corresponds to a sign vector of the inequality's coefficients. This proves useful as for each candidate of s the problem can be restricted to monotone set modeling, because maximal 1-removable subsets of $\mathcal{B}$ are lower sets (up to a change of coordinates). Although, to ensure a globally complete system of inequalities we have to apply a more general (and harder) learning method than the one from Section 5. More precisely, it is necessary to consider all general 1-removable lower sets, as opposed to considering only lower sets spanned by the extreme elements of the main lower set.

6.1 Classification of inequalities by their sign vectors

Consider an arbitrary linear inequality $\langle a, x\rangle \geq c$, $a \in \mathbb{R}^n, c \in \mathbb{R}$. Let $s = s(a) \in \mathbb{F}_2^n$ be given by

$$s_i = \begin{cases} 0, & \text{if } a_i \geq 0, \\ 1, & \text{if } a_i < 0. \end{cases} \tag{2}$$

Let $x' = x \oplus s$. Then, the inequality $\langle a, x\rangle \geq c$ can be expressed as

$$\sum_{i \in \overline{\mathrm{Supp}(s)}} a_i x'_i + \sum_{i \in \mathrm{Supp}(s)} a_i (1 - x'_i) \geq c.$$

Let $\langle a', x'\rangle \geq c'$ be its canonical form. It is easy to see that a' has only non-negative coefficients (the sign is switched only when $s_i = 1$ which happens only when $a_i < 0$.

We already studied monotone inequalities in Section 5. In particular, we have shown that such inequalities can only remove a lower set from $\{0,1\}^n$. It follows that *any* inequality removes a lower set "shifted" by $\oplus s$. It is not surprising since inequalities are monotone by nature. The described variable replacement only shows a concrete way to view each inequality as a monotone increasing one, i.e., as a constraint for an upper set.

Proposition 8. *Let $\langle a, x\rangle \geq c$ be an inequality, $a \in \mathbb{R}^n, c \in \mathbb{R}$. Then, the subset of $\mathbb{F}_2^n$ satisfying the inequality has form $s \oplus U$, where $U \subseteq \mathbb{F}_2^n$ is an upper set and $s = s(a) \in \mathbb{F}_2^n$ is defined as in (2).*

Remark 7. A *set* of inequalities with the same sign vector s still models an upper set, as upper/lower sets are closed under intersection and union.

6.2 Complete monotone learning

The sign-based classification allows us to build all the maximal 1-removable sets *separately* per each shift $s \in \mathbb{F}_2^n$, where each such set corresponds is a lower set included in $s \oplus \overline{B}$. Furthermore, all such sets are subsets of the single maximal lower set included in $s \oplus \overline{B}$ (which we will call the *main* lower set).

Proposition 9. *Let $L \subseteq \mathcal{B}$ be a lower set. If B is a maximal 1-removable subset of L, then it is a lower set.*

Proof. This is true in general, even if we allow non-monotone inequalities. Similarly to the proof of Corollary 1, we can show that an inequality removing a set $B \subseteq L$ can be transformed into a monotone inequality removing $\downarrow B$ (and nothing else) by replacing negative coefficients with zeros. Since $B \subseteq L$ implies $\downarrow B \subseteq L$, from maximality of B inside L it follows that B must be a lower set. □

This is very similar to the situation in the Quine-McCluskey algorithm. However, here we can not filter such lower sets by requiring the disjointness with the corresponding shift s. In the Quine-McCluskey algorithm this is done to avoid duplicate prime implicants; this idea exploits the fact that maximal elements are precisely maximal 1-removable sets by CNF clauses and have a clear structure of alternative representations. In our case, a maximal 1-removable set, while being a lower set, is not necessarily spanned by some maximal elements of the main lower set and it is not easy to directly decide if it is a duplicate or not.

In order to ensure a complete system of inequalities containing an optimal solution, we have to explore all subsets of the main lower set that are lower sets themselves. This task quickly becomes infeasible for a main lower set of large volume. However, for many dense Boolean functions the maximum size of the main lower set over all shifts is not very large.

A simple solution to implement such complete monotone learning is to apply the general method from Section 4 with the following optimizations:

1. The inconsistency oracle should use only monotone inequalities as in the adaptation from Subsection 5.2.
2. When the inconsistency oracle is called on a set $B \subseteq L$, it is sufficient to check one of $\downarrow B$ or $\mathrm{MaxSet}(B)$. The preference can be given to the best performing check, which is typically $\mathrm{MaxSet}(B)$ as it includes less inequalities.
3. In the case when the oracle replies that the given unknown set B is 1-removable, replace B with $\downarrow B$ to shorten the process of LearnUp (additionally, the lower closure replacement can be done on each step of LearnUp).
4. In the case when the oracle replies that the given unknown set B is not 1-removable, replace B with $\mathrm{MaxSet}(B)$ to shorten the process of LearnDown (if the sampling method does not ensure minimality).

These optimizations force the exploitation of the known lower set structure of each queried unit to shortcut the feasible maximization / infeasible minimization processes.

6.3 Eliminating redundant sets

In their reformulation of the Quine-McCluskey algorithm, Boura and Coggia utilize the following idea to avoid duplicate prime implicants (i.e., same subsets of $\mathcal{B}$ with different representation of the form $s \oplus \downarrow v$). Since lower closure converts bits with value 1 into wildcards, the bits s_i with $v_i = 1$ can be set assigned arbitrarily. Any choice would result in a different representation of the same set $s \oplus \downarrow v$. It is also easy to see that these representations are exhaustive. Therefore, it is sufficient to consider representations with constraint $s \wedge v = 0$ to ensure non-redundant resulting set of prime implicants.

In the complete monotone learning setting, we have sets of the form $s \oplus \downarrow V$, where $V \subseteq \mathbb{F}_2^n$ is a set instead of a single vector v. The following lemma and corollary characterize equivalent descriptions of maximal such sets.

Lemma 2. *Let $V \subseteq \mathbb{F}_2^n$ be maximal lower set such that $V \subseteq \mathcal{B} \subseteq \mathbb{F}_2^n$. Then, for any $\delta \in \mathbb{F}_2^n$, there exists a lower set $V' \subseteq \mathbb{F}_2^n$ such that*

$$V \subseteq \delta \oplus V' \subseteq \mathcal{B}$$

if and only if $\delta \preceq \bigwedge_{v \in \mathrm{MaxSet}(V)} v$*, in which case* $\delta \oplus V$ *is also a lower set.*

Proof. ($\Leftarrow$) If $\delta \preceq v$ for all $v \in \mathrm{MaxSet}(V)$, then $\delta \oplus \downarrow v = \downarrow v$ for all $v \in \mathrm{MaxSet}(V)$ and then

$$V \coloneqq \bigcup_{v \in V} \downarrow v = \bigcup_{v \in V} (\delta \oplus \delta \oplus \downarrow v) = \bigcup_{v \in V} (\delta \oplus \downarrow v) = \delta \oplus V.$$

By letting $V' = V$ this direction is proved. Same argument also implies that $\delta \oplus V$ is a lower set.
($\Rightarrow$) Assume that there exists $v \in \mathrm{MaxSet}(V)$ such that $\delta \not\preceq v$. We have $\downarrow v \subseteq V \subseteq \delta \oplus V'$ and thus, $\delta \oplus \downarrow v \subseteq V'$. Furthermore, $v' \coloneqq v \vee \delta$ is such that $v' \in \delta \oplus \downarrow v \subseteq V'$. Since $\delta \not\preceq v$, we know that $v' \succ v$, and since V' is a lower set, we obtain $\downarrow v' \subseteq V' \subseteq \mathcal{B}$. It follows that $\downarrow v$ is not a maximal lower set included in $\mathcal{B}$ and so V is not maximal too. This contradiction shows that there exist no $v \in \mathrm{MaxSet}(V)$ such that $\delta \not\preceq v$. □

Corollary 2. *Let* $V \subseteq \mathbb{F}_2^n$ *be maximal lower set such that* $s \oplus V \subseteq \mathcal{B} \subseteq \mathbb{F}_2^n$*. Then, for any* $s' \in \mathbb{F}_2^n$*, there exists a lower set* V' *such that*

$$s \oplus V \subseteq s' \oplus V' \subseteq \mathcal{B}$$

if and only if $(s' \oplus s) \preceq \bigwedge_{v \in \mathrm{MaxSet}(V)} v$*, in which case* $s' \oplus V = s \oplus V$*.*

Proof. Follows from Lemma 2 by setting $\mathcal{B} \leftarrow s \oplus \mathcal{B}$ and $\delta \leftarrow s \oplus s'$. □

We now show how Corollary 2 can be embedded in the algorithm to eliminate redundant maximal sets efficiently.

Assume that we have learned all maximal lower sets V for all s such that $s \oplus V \subseteq \mathcal{B}$. Our goal is to eliminate (mark) pairs (s, V) where V is a lower set such that there exist a vector s' and a lower set V' such that $s \oplus V \subsetneq s' \oplus V' \subseteq \mathcal{B}$. In case such strict inclusions do not exist for a given pair (s, V), we aim to eliminate all equivalent representations, i.e., pairs (s', V') such that $s \oplus V = s' \oplus V'$. This can be done in the following way.

For each identified 1-removable subset $B \subseteq \mathcal{B}$, we keep track of representations $s \oplus V$, V is a lower set, learned by the algorithm. More precisely, we consider all representations $B = s \oplus V$ where the lower set V is maximal such that $s \oplus V \subseteq \mathcal{B}$. Let $s \oplus V$ be any such representation. By Corollary 2, the other respective shifts must have the form $s \oplus \delta$ where $\delta \preceq \mathbf{v} \coloneqq \bigwedge_{v \in \mathrm{MaxSet}(V)} v$ (note that V is independent of the choice of the representation). Let $l \coloneqq \mathbf{wt}(\mathbf{v})$. We consider the two cases:

1. If for all $\delta \preceq \mathbf{v}$ the lower set V is maximal such that $s \oplus \delta \oplus V \subseteq \mathcal{B}$, then the set B is a maximal 1-removable set. Indeed, otherwise, we could consider the shift correspond to the inequality removing a superset of B and obtain a contradiction. This case is distinguished by observing all 2^l shifts corresponding to representations of B returned by the learning algorithm (recall that the algorithm returns maximal lower sets per each shift). Lastly, a single representation can be arbitrarily selected for B and returned as non-redundant.

2. If for some $\delta \preceq \mathbf{v}$ there exists a lower set V' such that $V \subsetneq V'$ and $s \oplus \delta \oplus V' \subseteq \mathcal{B}$, then the set B is redundant, i.e., it is not a maximal 1-removable set. This case is distinguished by observing strictly less than 2^l shifts in representations of B returned by the learning algorithm. Such sets B can be marked as redundant as there exist shifts for which B can be extended.

6.4 Full algorithm

The full algorithm is summarized in Algorithm 2.

Algorithm 2 Constructing complete inequalities system for a Boolean function

Input: $\mathcal{G} \subseteq \mathbb{F}_2^n, \mathcal{B} \subseteq \overline{\mathcal{G}}$
Output: complete set S of inequalities for modeling $\mathcal{G}, \mathcal{B}$

assume $C[B] = 0$ for all $B \subseteq \mathcal{B}$
for all $s \in \mathbb{F}_2^n$ **do**
    $L \leftarrow$ the maximal lower set included in $s \oplus \mathcal{B}$
    $S_s \leftarrow$ maximal 1-removable subsets of L by monotone inequalities (see Algorithm 1, Subsection 5.2, Subsection 6.2); record the associated inequalities
    for all $V \in S_s$ **do**
        $C[s \oplus V] \leftarrow C[s \oplus V] + 1$
$S \leftarrow \emptyset$
for all $V : C[V] > 0$ **do**
    $\mathbf{v} \coloneqq \bigwedge_{v \in \mathrm{MaxSet}(V)} v$
    if $C[V] = 2^{|\mathbf{v}|}$ **then**
        $S \leftarrow S \cup \{V\}$
return S

Acknowledgements

We thank Gurobi for providing a license for this work.

References

[AHS21] Gennadiy Averkov, Christopher Hojny, and Matthias Schymura. Computational Aspects of Relaxation Complexity. In Mohit Singh and David P. Williamson, editors, *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, pages 368–382, Cham, 2021. Springer International Publishing. 2, 4, 8

[Ang88] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, April 1988. 6

[AS21] Gennadiy Averkov and Matthias Schymura. Complexity of linear relaxations in integer programming. *Mathematical Programming*, February 2021. 2

[AST+17] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP modeling for (large) s-boxes to optimize probability of differential characteristics. *IACR Trans. Symm. Cryptol.*, 2017(4):99–129, 2017. 2, 3, 7

[BBK+] Begül Bilgin, Andrey Bogdanov, Miroslav Knežević, Florian Mendel, and Qingju Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, Lecture Notes in Computer Science, pages 142–158. Springer. 26

[BC20] Christina Boura and Daniel Coggia. Efficient milp modelings for sboxes and linear layers of spn ciphers. *IACR Transactions on Symmetric Cryptology*, 2020(3):327–361, 2020. 2, 3, 4, 7

[BEG+02] E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan, and K. Makino. Dual-Bounded Generating Problems: All Minimal Integer Solutions for a Monotone

System of Linear Inequalities. *SIAM Journal on Computing*, 31(5):1624–1643, May 2002. 9

[BFFH20] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020. 14

[BHZ08] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1):3–21, 2008. 7

[BI95] J. C. Bioch and T. Ibaraki. Complexity of Identification and Dualization of Positive Boolean Functions. *Information and Computation*, 123(1):50–63, November 1995. 9, 13

[BM09] Endre Boros and Kazuhisa Makino. A Fast and Simple Parallel Algorithm for the Monotone Duality Problem. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 183–194, Berlin, Heidelberg, 2009. Springer. 14

[BSMH84] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, USA, 1984. 3

[Dan90] George B. Dantzig. *Origins of the simplex method*, page 141–151. Association for Computing Machinery, Jun 1990. 10

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *Journal of Cryptology*, 34(3):33, Jun 2021. 25

[Der65] Dertouzos. *Threshold Logic: A Synthesis Approach*. 1965. 9

[Dik67] I. I. Dikin. Iterative solution of problems of linear and quadratic programming. *Soviet Mathematics. Doklady*, 8:674–675, 1967. 10

[Elb08] Khaled M. Elbassioni. On the complexity of monotone dualization and generating minimal hypergraph transversals. *Discrete Applied Mathematics*, 156(11):2109–2123, June 2008. 14

[EMG08] Thomas Eiter, Kazuhisa Makino, and Georg Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, June 2008. 14

[Fei95] Uriel Feige. *A Threshold of Ln n for Approximating Set Cover*. 1995. 8

[FK96] Michael L. Fredman and Leonid Khachiyan. On the Complexity of Dualization of Monotone Disjunctive Normal Forms. *Journal of Algorithms*, 21(3):618–628, November 1996. 9, 13

[GAB+20] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. Technical report, Optimization Online, 2020. 8

[Gai84] D. N. Gainanov. On one criterion of the optimality of an algorithm for evaluating monotonic boolean functions. *USSR Computational Mathematics and Mathematical Physics*, 24(4):176–181, January 1984. 9, 12, 13

[GK99] V. Gurvich and L. Khachiyan. On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions. *Discrete Applied Mathematics*, 96-97:363–373, October 1999. 9, 11, 13

[GO21] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021. 8

[Gru07] Nicolle Gruzling. *Linear Separability of the Vertices of an N-Dimensional Hypercube.* Master of Science, University of Northern British Columbia, 2007. 9

[Han66] Georges Hansel. Sur le nombre des fonctions booléennes monotones de n variables. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Série B*, 262:1088–1090, 1966. 11

[Jer75] R. G. Jeroslow. On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11(2):119–124, January 1975. 2

[Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, December 1974. 8

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, STOC '84, page 302–311. Association for Computing Machinery, Dec 1984. 10

[KBEG06] Leonid Khachiyan, Endre Boros, Khaled Elbassioni, and Vladimir Gurvich. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discrete Applied Mathematics*, 154(16):2350–2372, November 2006. 14

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979. MSC2010: 90C31 = Sensitivity, stability, parametric optimization MSC2010: 90C05 = Linear programming MSC2010: 68Q25 = Analysis of algorithms and problem complexity MSC2010: 65K05 = Numerical mathematical programming methods Zbl: 0414.90086. 10

[KW14] Volker Kaibel and Stefan Weltge. Lower Bounds on the Sizes of Integer Programs without Additional Variables. In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, pages 321–332, Cham, 2014. Springer International Publishing. 2, 3

[KW15] Volker Kaibel and Stefan Weltge. Lower bounds on the sizes of integer programs without additional variables. *Mathematical Programming*, 154(1):407–425, December 2015. 2, 3

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, January 1975. 8

[LY94] Carsten Lund and Mihalis Yannakakis. On the Hardness of Approximating Minimization Problems. *J. ACM*, 41:960–981, September 1994. 8

[Mak17] Andrew Makhorin. GNU Linear Programming Kit, version 4.65. http://www.gnu.org/software/glpk/glpk.html, 2017. 8, 10

[McC56] E. J. McCluskey. Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444, 1956. 3, 7

[Meg88] Nimrod Megiddo. On the complexity of polyhedral separability. *Discrete & Computational Geometry*, 3(4):325–337, December 1988. 2

[MWGP12] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuan-Kun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology*, Lecture Notes in Computer Science, pages 57–76, Berlin, Heidelberg, 2012. Springer. 2

[Qui52] W. V. Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531, 1952. 3

[Qui55] W. V. Quine. A way to simplify truth functions. *The American Mathematical Monthly*, 62(9):627–631, 1955. 3, 7

[Sed18] Nafiseh Sedaghat. Speeding up Dualization in the Fredman-Khachiyan Algorithm B. pages 6:1–6:13, 2018. 14

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 158–178. Springer, Heidelberg, 2014. 2, 3, 7, 8

[Sin05] Carsten Sinz. *Towards an Optimal CNF Encoding of Boolean Cardinality Constraints*, volume 3709 of *Lecture Notes in Computer Science*, page 827–831. Springer Berlin Heidelberg, 2005. 14

[Sla97] Petr Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, December 1997. 8, 9

[Sok82] N. A. Sokolov. On the optimal evaluation of monotonic Boolean functions. *USSR Computational Mathematics and Mathematical Physics*, 22(2):207–220, January 1982. 11

[Spe28] Emanuel Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928. 6

[ST17] Yu Sasaki and Yosuke Todo. New algorithm for modeling s-box in milp based differential and division trail search. In Pooya Farshim and Emil Simion, editors, *Innovative Security Solutions for Information Technology and Communications*, pages 150–165, Cham, 2017. Springer International Publishing. 3, 8

[Sun21] Yao Sun. Towards the least inequalities for describing a subset in z_2^n. Cryptology ePrint Archive, Report 2021/1084, 2021. https://ia.cr/2021/1084. 3, 5

[SWW21] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the Search of Differential and Linear Characteristics with the SAT Method. *IACR Transactions on Symmetric Cryptology*, pages 269–315, March 2021. 3

[SYY+] Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, and Hidema Tanaka. The Block Cipher SC2000. In Mitsuru Matsui, editor, *Fast Software Encryption*, Lecture Notes in Computer Science, pages 312–327. Springer. 27

[The21] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.3)*, 2021. `https://www.sagemath.org`. 7, 10

[TT02] Vetle I. Torvik and Evangelos Triantaphyllou. Minimizing the Average Query Complexity of Learning Monotone Boolean Functions. *INFORMS Journal on Computing*, 14(2):144–174, May 2002. 9, 11

[TT09] Vetle I. Torvik and Evangelos Triantaphyllou. Inference of monotone boolean functions. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1591–1598. Springer US, Boston, MA, 2009. 9

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984. 9

[Wel15] Stefan Weltge. *Dipl.-Math. Stefan Weltge geb. am 14. August 1986 in Lichtenstein*. PhD thesis, 2015. 2

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 648–678. Springer, Heidelberg, 2016. 2

[Zhu78] Yuri Ivanovich Zhuravlyov. Algebraic approach to the solution of recognition or classification problems. *Problemy kibernetiki*, (33):5–68, 1978. 9, 11

[ZT83] Yu.A. Zuev and V.N. Trishin. The lower bound of the number of inequalities which represent a monotonic boolean function of n variables. *USSR Computational Mathematics and Mathematical Physics*, 23(3):155–156, 1983. 15

A Inequalities

We provide inequalities for several functions obtained using our approach. Complete digital archive will be made publicly available.

ASCON-DDT Optimal 27 inequalities for the DDT support of the 5-bit S-box of ASCON [DEMS21]:

$$
\begin{aligned}
-2x_0 - x_1 + 5x_2 + 4x_3 - 6x_4 + 5y_0 + 4y_1 - 6y_2 - 2y_3 - 6y_4 &\geq -17 && (\#1)\\
-3x_0 - 5x_1 + x_2 - 4x_3 + 6x_4 - y_0 + 6y_1 + 3y_2 - 6y_3 - 6y_4 &\geq -19 && (\#2)\\
5x_0 - x_1 + 2x_2 - 7x_3 + 7x_4 + y_0 - 2y_1 - 7y_2 - 7y_3 + 7y_4 &\geq -17 && (\#3)\\
9x_0 - 11x_1 + 4x_2 - 3x_3 + 7x_4 - y_0 - 2y_1 - 11y_2 + 11y_3 - 11y_4 &\geq -28 && (\#4)\\
2x_0 + 4x_1 + 10x_2 + 5x_3 - 5x_4 - 10y_0 - 8y_1 - 6y_2 + y_3 - 10y_4 &\geq -29 && (\#5)\\
-4x_0 + 3x_1 + 6x_2 + 7x_3 - 7x_4 - 5y_0 + 6y_1 - 7y_2 + 2y_3 + 10y_4 &\geq -13 && (\#6)\\
-11x_0 - 10x_1 - 8x_2 - 10x_3 + 11x_4 + 2y_0 + 12y_1 + y_2 + 4y_3 + 4y_4 &\geq -27 && (\#7)\\
12x_0 + 12x_1 - 12x_2 - 10x_3 - 6x_4 - 11y_0 - 2y_1 - 2y_2 + y_3 + 6y_4 &\geq -31 && (\#8)\\
-2x_0 - 12x_1 - 6x_2 + 10x_3 - 2x_4 + 11y_0 - 11y_1 + 6y_2 + y_3 - 12y_4 &\geq -33 && (\#9)\\
6x_0 - 12x_1 + 12x_2 - 11x_3 - 9x_4 - y_0 - y_1 + 6y_2 + 12y_3 + 3y_4 &\geq -22 && (\#10)\\
-3x_0 - 6x_1 - 12x_2 + 12x_3 + x_4 - 8y_0 - 8y_1 - 4y_2 + 3y_3 + 11y_4 &\geq -29 && (\#11)\\
13x_0 + 9x_1 - 13x_2 + 12x_3 - 5x_4 + 11y_0 - 4y_1 + 2y_2 + y_3 - 5y_4 &\geq -14 && (\#12)\\
8x_0 - 4x_1 - 11x_2 - 15x_3 - 14x_4 + y_0 + 16y_1 + 8y_2 + 12y_3 + 2y_4 &\geq -28 && (\#13)\\
12x_0 - 12x_1 - 10x_2 - 5x_3 + 6x_4 + y_0 - 10y_1 + 2y_2 - 6y_3 - 2y_4 &\geq -33 && (\#14)\\
-3x_0 - 14x_1 + 13x_2 + 12x_3 + 10x_4 - y_0 - 2y_1 + 14y_2 - 3y_3 + 4y_4 &\geq -9 && (\#15)\\
x_0 - 11x_1 - 11x_2 + 12x_3 + 9x_4 + 3y_0 + 3y_1 - 9y_2 + y_3 + 3y_4 &\geq -19 && (\#16)\\
-23x_0 + 20x_1 - 8x_2 - 22x_3 - 16x_4 + 8y_0 - 4y_1 - 2y_2 + y_3 - 24y_4 &\geq -75 && (\#17)\\
-15x_0 - 4x_1 + 2x_2 - 10x_3 - 15x_4 - y_0 - 3y_1 + 17y_2 + 17y_3 - 6y_4 &\geq -37 && (\#18)\\
-2x_0 - 4x_1 - 19x_2 + 17x_3 + 7x_4 - 15y_0 + 11y_1 - 8y_2 - y_3 - 13y_4 &\geq -43 && (\#19)\\
-3x_0 + 26x_1 - 21x_2 + 26x_3 + 19x_4 - 9y_0 - 7y_1 + 17y_2 + y_3 - 5y_4 &\geq -19 && (\#20)\\
-14x_0 - 6x_1 - 8x_2 - 19x_3 - 22x_4 - 2y_0 - 4y_1 - 18y_2 - 18y_3 - y_4 &\geq -90 && (\#21)\\
9x_0 + 17x_1 + 16x_2 + 23x_3 + 21x_4 + 2y_0 + 4y_1 - 24y_2 + 8y_3 - y_4 &\geq 0 && (\#22)\\
-2x_0 + 17x_1 + 10x_2 + 20x_3 - 11x_4 + 19y_0 - 4y_1 - 3y_2 - y_3 + 19y_4 &\geq 0 && (\#23)\\
3x_0 - 4x_1 + 14x_2 + 22x_3 + 17x_4 - 9y_0 + 8y_1 + 24y_2 - 2y_3 - 9y_4 &\geq 0 && (\#24)\\
14x_0 + 32x_1 + 10x_2 - 33x_3 + 18x_4 + 20y_0 - y_1 + 2y_2 + 6y_3 + 24y_4 &\geq 0 && (\#25)\\
-16x_0 + 45x_1 + 20x_2 - 44x_3 - 31x_4 - 21y_0 - 3y_1 - 2y_2 - 10y_3 - 25y_4 &\geq -105 && (\#26)\\
-8x_0 + 18x_1 + x_2 + 33x_3 - 22x_4 + 13y_0 + 12y_1 + 15y_2 - 3y_3 + 20y_4 &\geq 0 && (\#27)
\end{aligned}
$$

FIDES-5-DDT Optimal 57 inequalities for the DDT support of the 5-bit S-box of FIDES [BBK+]:

$$
\begin{aligned}
-2x_0 - 5x_1 + 2x_2 + 5x_3 - 4x_4 - 5y_0 + 5y_1 - 5y_2 - y_3 - 3y_4 &\geq -20 && (\#1)\\
4x_0 + 4x_1 - 4x_2 - 3x_3 + 3x_4 - y_0 + 4y_1 + 4y_2 - y_3 + y_4 &\geq -5 && (\#2)\\
8x_0 - 7x_1 - 4x_2 - 6x_3 - 8x_4 - 8y_0 - 2y_1 + 4y_2 - 8y_3 - y_4 &\geq -36 && (\#3)\\
5x_0 + 2x_1 - 2x_2 - 4x_3 + 4x_4 - 3y_0 + 5y_1 + 5y_2 + y_3 - 3y_4 &\geq -7 && (\#4)\\
-7x_0 + 6x_1 + 7x_2 - 7x_3 - 4x_4 - 4y_0 + y_1 + 2y_2 - 8y_3 - 8y_4 &\geq -30 && (\#5)\\
12x_0 + x_1 + 9x_2 + 11x_3 + 12x_4 - 5y_0 - 5y_1 - 2y_2 + 3y_3 + 11y_4 &\geq 0 && (\#6)\\
10x_0 - 11x_1 + 10x_2 - 6x_3 - 10x_4 - 2y_0 - y_1 + 12y_2 - 2y_3 + 6y_4 &\geq -20 && (\#7)\\
8x_0 + 8x_1 + 2x_2 + 5x_3 - 3x_4 + 3y_0 - 8y_1 + y_2 + 6y_3 - 8y_4 &\geq -11 && (\#8)\\
6x_0 - x_1 + 4x_2 + 8x_3 - 8x_4 + 2y_0 - 7y_1 - 8y_2 + 4y_3 - 8y_4 &\geq -24 && (\#9)\\
4x_0 - 2x_1 + 3x_2 - 4x_3 + 4x_4 + 4y_0 + y_1 - 2y_2 - 4y_3 - 4y_4 &\geq -12 && (\#10)\\
-6x_0 + 6x_1 + 4x_2 - 5x_3 - 5x_4 + y_0 - 2y_1 - 2y_2 + 6y_3 + 6y_4 &\geq -14 && (\#11)\\
-2x_0 - 2x_1 - 7x_2 + 3x_3 + 3x_4 + 10y_0 + 8y_1 + 11y_2 + 11y_3 + 11y_4 &\geq 0 && (\#12)\\
-2x_0 - 4x_1 - 3x_2 + 4x_3 - 4x_4 + y_0 - 4y_1 - 4y_2 + 2y_3 + 4y_4 &\geq -17 && (\#13)\\
-3x_0 - 4x_1 + 3x_2 + 4x_3 + 2x_4 + 4y_0 - 4y_1 - 2y_2 - 4y_3 + y_4 &\geq -13 && (\#14)\\
-3x_0 + 4x_1 + x_2 + 2x_3 - 4x_4 - 4y_0 - 4y_1 - 4y_2 - 4y_3 - 2y_4 &\geq -21 && (\#15)\\
-3x_0 + 3x_1 - x_2 + 3x_3 + x_4 - 3y_0 + 2y_1 - 3y_2 + 3y_3 + 3y_4 &\geq -7 && (\#16)\\
-6x_0 - 3x_1 + 6x_2 + 3x_3 - 2x_4 + 7y_0 + 7y_1 + 7y_2 + 5y_3 + y_4 &\geq -4 && (\#17)\\
-4x_0 - 4x_1 + 7x_2 - 7x_3 + 9x_4 - 5y_0 - 2y_1 + y_2 + 9y_3 - 5y_4 &\geq -18 && (\#18)\\
6x_0 + 3x_1 + 9x_2 + 10x_3 + 7x_4 + y_0 + 4y_1 - 10y_2 + y_3 + 7y_4 &\geq 0 && (\#19)\\
7x_0 + 6x_1 + 4x_2 - 3x_3 + 2x_4 + 7y_0 - 5y_1 - 7y_2 - y_3 - 7y_4 &\geq -16 && (\#20)\\
6x_0 - 6x_1 - 2x_2 + 4x_3 + 6x_4 + 7y_0 + 3y_1 + 7y_2 + y_3 + 5y_4 &\geq -1 && (\#21)\\
2x_0 - 3x_1 - 2x_2 + 4x_3 - 4x_4 - y_0 + 5y_1 - 5y_2 - 5y_3 - 3y_4 &\geq -18 && (\#22)\\
-4x_0 - 3x_1 - 9x_2 + 4x_3 - 5x_4 + 2y_0 + 10y_1 + 10y_2 + 7y_3 + 10y_4 &\geq -11 && (\#23)\\
5x_0 - 5x_1 - x_2 + 4x_3 + 6x_4 - 7y_0 + 3y_1 - 7y_2 - 2y_3 + 6y_4 &\geq -15 && (\#24)\\
4x_0 - 3x_1 - 4x_2 - 4x_3 - 3x_4 + 4y_0 + y_1 - y_2 + 4y_3 - y_4 &\geq -12 && (\#25)\\
-4x_0 - 4x_1 + 9x_2 - 9x_3 + 8x_4 + 6y_0 + y_1 + 2y_2 + 10y_3 + 6y_4 &\geq -7 && (\#26)\\
5x_0 + 6x_1 + 6x_2 + 3x_3 - 4x_4 + 3y_0 + 6y_1 - y_2 - y_3 + 6y_4 &\geq 0 && (\#27)\\
-3x_0 + x_1 - 4x_2 - 4x_3 - 2x_4 + 2y_0 + 4y_1 - 4y_2 - 4y_3 + 4y_4 &\geq -17 && (\#28)\\
-6x_0 + 4x_1 - 5x_2 + 4x_3 - 6x_4 - y_0 + 6y_1 - 2y_2 + 6y_3 - 2y_4 &\geq -16 && (\#29)\\
-4x_0 + 8x_1 - 8x_2 + 6x_3 - 7x_4 - y_0 - 8y_1 + 2y_2 - 8y_3 + 4y_4 &\geq -28 && (\#30)\\
-4x_0 - x_1 - 3x_2 + 4x_3 + 2x_4 + 3y_0 - 2y_1 - 4y_2 - 4y_3 + 4y_4 &\geq -14 && (\#31)\\
6x_0 + 4x_1 - 6x_2 - 6x_3 + 5x_4 - 2y_0 - 6y_1 - 6y_2 + y_3 - 2y_4 &\geq -22 && (\#32)\\
-3x_0 - 3x_1 - 3x_2 + 4x_3 + 2x_4 + y_0 - 2y_1 + 4y_2 - 4y_3 - 4y_4 &\geq -15 && (\#33)\\
5x_0 - 6x_1 - 6x_2 + 4x_3 + 5x_4 + 6y_0 - 2y_1 + 6y_2 - y_3 - 2y_4 &\geq -11 && (\#34)\\
3x_0 - 2x_1 - 3x_2 + 6x_3 - 6x_4 + y_0 + 7y_1 + 7y_2 + 7y_3 + 5y_4 &\geq -4 && (\#35)\\
-11x_0 - 11x_1 + 10x_2 - 10x_3 + 8x_4 - 4y_0 - 2y_1 - 4y_2 + 12y_3 - y_4 &\geq -31 && (\#36)\\
16x_0 + 16x_1 + 14x_2 + 15x_3 + 10x_4 - 2y_0 - 6y_1 + y_2 - 2y_3 - 6y_4 &\geq 0 && (\#37)\\
2x_0 + 5x_1 - 6x_2 + 5x_3 - 4x_4 + 2y_0 - 6y_1 + y_2 - 6y_3 + 6y_4 &\geq -16 && (\#38)\\
-x_0 + 3x_1 - x_2 + 3x_3 + 2x_4 - 3y_0 + 2y_1 + 3y_2 + 3y_3 - 3y_4 &\geq -5 && (\#39)\\
-2x_0 + 2x_1 - x_2 + 2x_3 + 2x_4 - 2y_0 + y_1 - 2y_2 - 2y_3 - 2y_4 &\geq -9 && (\#40)\\
-6x_0 - 5x_1 - 3x_2 - 6x_3 - 6x_4 + 3y_0 - y_1 - 6y_2 - 6y_3 - y_4 &\geq -34 && (\#41)\\
-6x_0 + x_1 - 4x_2 + 2x_3 + 3x_4 + 6y_0 + 3y_1 - 6y_2 + 6y_3 - 6y_4 &\geq -16 && (\#42)\\
-x_0 + 3x_1 + 2x_2 + x_3 - 3x_4 + 3y_0 - 3y_1 + 3y_2 - 3y_3 - 2y_4 &\geq -9 && (\#43)\\
-3x_0 + 12x_1 + 16x_2 + 17x_3 + 13x_4 - 7y_0 + 17y_1 + 4y_2 - 7y_3 + 2y_4 &\geq 0 && (\#44)\\
-6x_0 + 4x_1 + 6x_2 - 5x_3 - 6x_4 + 2y_0 + y_1 - 2y_2 - 6y_3 - 6y_4 &\geq -25 && (\#45)\\
-4x_0 - 6x_1 - 5x_2 - 5x_3 + 5x_4 - 6y_0 - 6y_1 + 2y_2 + y_3 + 2y_4 &\geq -26 && (\#46)\\
-3x_0 - 2x_1 + 4x_2 + 4x_3 - 2x_4 - 4y_0 - 4y_1 + 4y_2 + 2y_3 + y_4 &\geq -11 && (\#47)\\
-7x_0 + 11x_1 + 2x_2 - 9x_3 + 11x_4 + 12y_0 + 3y_1 + 5y_2 + y_3 + 12y_4 &\geq -4 && (\#48)\\
2x_0 + 3x_1 + 3x_2 - 4x_3 - 4x_4 - 4y_0 - 4y_1 + y_2 + 2y_3 + 4y_4 &\geq -12 && (\#49)\\
3x_0 + x_1 - 4x_2 - 4x_3 + 2x_4 + 3y_0 - 4y_1 - 4y_2 + 2y_3 + 3y_4 &\geq -12 && (\#50)\\
-9x_0 + 6x_1 - 12x_2 - 11x_3 + 11x_4 + 12y_0 + 6y_1 + 3y_2 - y_3 - y_4 &\geq -22 && (\#51)\\
5x_0 + 4x_1 + 2x_2 - 3x_3 + 2x_4 - 5y_0 - 3y_1 - 5y_2 - y_3 + 5y_4 &\geq -12 && (\#52)\\
4x_0 - 5x_1 - 6x_2 + 6x_3 + 6x_4 - 6y_0 + y_1 - 6y_2 + 2y_3 - 2y_4 &\geq -19 && (\#53)\\
-5x_0 - 4x_1 - 5x_2 - 4x_3 - 6x_4 + y_0 - 2y_1 + 6y_2 + 6y_3 - 2y_4 &\geq -22 && (\#54)\\
4x_0 + 4x_1 - 4x_2 + 2x_3 - x_4 + 3y_0 + 4y_1 + 2y_2 - 4y_3 - 4y_4 &\geq -9 && (\#55)\\
2x_0 + 4x_1 + 3x_2 - 4x_3 - 4x_4 - 4y_0 + 4y_1 - y_2 - 2y_3 - 4y_4 &\geq -15 && (\#56)\\
2x_0 + 5x_1 - 2x_2 + 4x_3 - 5x_4 - y_0 + 5y_1 - y_2 + 5y_3 + 5y_4 &\geq -4 && (\#57)
\end{aligned}
$$

SC2000-5-DDT Optimal 60 inequalities for the DDT support of the 5-bit S-box of SC2000 [SYY+]:

$$
\begin{aligned}
-x_0 + 2x_1 - 3x_2 - 3x_3 - 3x_4 + 10y_0 + 10y_1 + 9y_2 + 8y_3 + 10y_4 &\geq 0 && (\#1)\\
-3x_0 + x_1 - 3x_2 + 3x_3 + 3x_4 - 3y_0 - 3y_1 + 2y_2 - y_3 - 3y_4 &\geq -13 && (\#2)\\
2x_0 + x_1 + 2x_2 - 2x_3 - 2x_4 - 2y_0 - 2y_1 + y_2 - 2y_3 - 2y_4 &\geq -10 && (\#3)\\
-6x_0 + 2x_1 - 6x_2 + x_3 - 2x_4 - 5y_0 + 6y_1 - 4y_2 + 6y_3 - 6y_4 &\geq -23 && (\#4)\\
6x_0 - 2x_1 + x_2 + 6x_3 - 2x_4 - 5y_0 + 6y_1 + 6y_2 - 4y_3 - 6y_4 &\geq -13 && (\#5)\\
8x_0 + 8x_1 + x_2 + 2x_3 + 4x_4 + 4y_0 + 7y_1 + 6y_2 + 7y_3 - 8y_4 &\geq 0 && (\#6)\\
x_0 + 5x_1 - 9x_2 - 2x_3 - 9x_4 - 4y_0 + 7y_1 + 4y_2 - 9y_3 + 9y_4 &\geq -24 && (\#7)\\
4x_0 + 3x_1 - 4x_2 + 2x_3 - 4x_4 - 2y_0 - y_1 - 4y_2 + 4y_3 + 4y_4 &\geq -11 && (\#8)\\
3x_0 - 4x_1 - 4x_2 + 2x_3 + 4x_4 - 2y_0 + 3y_1 - 4y_2 - y_3 + 3y_4 &\geq -11 && (\#9)\\
-x_0 - 2x_2 - 2x_3 + 6x_4 - 5y_0 - 4y_1 + 5y_2 + 4y_3 + 6y_4 &\geq -8 && (\#10)\\
10x_0 + 10x_1 + 10x_2 + 7x_3 + 10x_4 - 5y_0 + 2y_1 - 5y_2 + y_3 + y_4 &\geq 0 && (\#11)\\
-3x_0 - 3x_1 - 3x_2 + x_3 - 3x_4 - 2y_0 + 3y_1 - 2y_2 + 3y_3 + y_4 &\geq -13 && (\#12)\\
2x_0 - 4x_1 + 4x_2 - x_3 - 4x_4 - 3y_0 + 3y_1 - 4y_2 - 2y_3 + 4y_4 &\geq -14 && (\#13)\\
-2x_0 - 6x_1 - x_2 - 6x_3 + 2x_4 + 5y_0 - 5y_1 - 4y_2 + 6y_3 + 6y_4 &\geq -18 && (\#14)\\
3x_0 - 3x_1 + 2x_2 - 3x_3 + 3x_4 + 2y_0 + 3y_1 - 3y_2 + y_3 - y_4 &\geq -7 && (\#15)\\
-x_0 + 7x_1 - x_2 + 4x_3 + 7x_4 + 3y_0 - 6y_1 - 7y_2 - 3y_3 + 7y_4 &\geq -11 && (\#16)\\
2x_0 - 2x_1 - 2x_2 + 2x_3 - x_4 + 2y_0 + y_1 - 2y_2 - 2y_3 + 2y_4 &\geq -7 && (\#17)\\
-6x_0 + 2x_1 + x_2 - 2x_3 - 6x_4 + 6y_0 - 5y_1 + 6y_2 - 6y_3 - 4y_4 &\geq -23 && (\#18)\\
-10x_0 - 10x_1 + 4x_2 + 2x_3 + 7x_4 + 3y_0 + 6y_1 + 9y_2 + 10y_3 - 5y_4 &\geq -15 && (\#19)\\
8x_0 - 2x_1 + x_2 - 2x_3 + 8x_4 + 8y_0 - 7y_1 + 2y_2 - 8y_3 - 6y_4 &\geq -17 && (\#20)\\
-x_0 + 3x_1 - x_2 + 6x_3 + 6x_4 - 3y_0 - 6y_1 - 6y_2 - 5y_3 + 3y_4 &\geq -16 && (\#21)\\
-2x_0 + 6x_1 - 2x_2 + 6x_3 + x_4 + 5y_0 - 5y_1 - 6y_2 + 5y_3 + 4y_4 &\geq -9 && (\#22)\\
8x_0 + 4x_1 + 8x_2 + x_3 + 2x_4 - 7y_0 + 7y_1 - 6y_2 + 4y_3 - 6y_4 &\geq -11 && (\#23)\\
5x_0 + 2x_1 - 5x_2 - 5x_3 - x_4 + 3y_0 - 4y_1 - 5y_2 + y_3 - 5y_4 &\geq -20 && (\#24)\\
-7x_0 - 7x_1 + x_2 - 2x_3 - 2x_4 + 7y_0 + 6y_1 + 5y_2 + 7y_3 - y_4 &\geq -12 && (\#25)\\
-x_0 + 14x_1 - 5x_2 + 2x_3 - 5x_4 + 9y_0 + 14y_1 + 13y_2 - 3y_3 + 12y_4 &\geq 0 && (\#26)\\
-4x_0 + 2x_1 + 4x_2 - x_3 - 4x_4 - 4y_0 - 2y_1 - 4y_2 + 3y_3 + 4y_4 &\geq -15 && (\#27)\\
-x_0 - 3x_1 + x_2 + 3x_3 - 3x_4 - 3y_0 - 3y_1 - 2y_2 - 3y_3 - 3y_4 &\geq -18 && (\#28)\\
-3x_0 + x_1 + 3x_2 - 3x_3 + x_4 + 3y_0 - 2y_1 - 3y_2 + 3y_3 - 3y_4 &\geq -11 && (\#29)\\
3x_0 - 3x_1 + 2x_2 + 3x_3 - 3x_4 + y_0 + 2y_1 - 3y_2 + 2y_3 - y_4 &\geq -7 && (\#30)\\
x_0 - 4x_1 + 4x_2 - 4x_3 + 2x_4 - 4y_0 - 3y_1 - 4y_2 + 2y_3 - 3y_4 &\geq -18 && (\#31)\\
x_0 - 4x_1 - 2x_2 - 8x_3 - 8x_4 - 4y_0 - 8y_1 - 6y_2 - 7y_3 + 8y_4 &\geq -39 && (\#32)\\
2x_0 + x_1 + 13x_2 + 13x_3 + 9x_4 - 12y_0 - 5y_1 + 7y_2 - 4y_3 + 11y_4 &\geq -8 && (\#33)\\
-4x_0 + 4x_1 + 4x_2 + 4x_3 - 4x_4 + 2y_0 + y_1 - 2y_2 + 4y_3 + 3y_4 &\geq -6 && (\#34)\\
-2x_0 - 3x_1 - 3x_2 - 3x_3 - 3x_4 + y_0 + y_1 + 3y_2 - 2y_3 - 3y_4 &\geq -16 && (\#35)\\
2x_0 + 13x_1 - 6x_2 - 6x_3 + 3x_4 + 13y_0 + 13y_1 + 11y_2 - y_3 + 10y_4 &\geq 0 && (\#36)\\
3x_0 - 2x_1 - 3x_2 + 4x_3 - 3x_4 - 3y_0 - 2y_1 + 2y_2 + y_3 - 4y_4 &\geq -13 && (\#37)\\
4x_0 + 12x_1 + 2x_2 + 4x_3 + x_4 + 11y_0 + 8y_1 + 10y_2 - 12y_3 + 11y_4 &\geq 0 && (\#38)\\
-2x_0 + 6x_1 - 2x_2 - 6x_3 - 6x_4 - 6y_0 - 6y_1 - 4y_2 - 5y_3 - y_4 &\geq -32 && (\#39)\\
-4x_0 - 6x_1 - 6x_2 + 6x_3 + x_4 - 3y_0 - 2y_1 - 3y_2 + 6y_3 - 5y_4 &\geq -23 && (\#40)\\
12x_0 - x_1 + 6x_2 + 3x_3 - x_4 + 11y_0 - 6y_1 - 12y_2 - 9y_3 - 11y_4 &\geq -28 && (\#41)\\
-5x_0 - 5x_1 + x_2 + 5x_3 - 3x_4 + 4y_0 - 5y_1 + 3y_2 - 2y_3 + 2y_4 &\geq -15 && (\#42)\\
x_0 + 6x_1 + 12x_2 + 2x_3 + 2x_4 + 6y_0 + 11y_1 - 10y_2 - 11y_3 - 11y_4 &\geq -20 && (\#43)\\
-7x_0 + 17x_1 + 4x_2 - 7x_3 + 2x_4 + 15y_0 + 12y_1 + 13y_2 - 3y_3 + 17y_4 &\geq 0 && (\#44)\\
-x_0 + 4x_1 + 7x_2 - x_3 + 7x_4 - 3y_0 + 7y_1 + 3y_2 - 7y_3 + 6y_4 &\geq -5 && (\#45)\\
-2x_0 - 2x_1 + 8x_2 + 2x_3 - x_4 + 8y_0 + 8y_1 - 7y_2 - 8y_3 - 6y_4 &\geq -18 && (\#46)\\
5x_0 - x_1 - x_2 + 5x_3 + 5x_4 + 2y_0 - 3y_1 + 5y_2 + 5y_3 + 4y_4 &\geq 0 && (\#47)\\
-6x_0 + 2x_1 - 6x_2 - x_3 + 2x_4 - 5y_0 + 5y_1 - 6y_2 + 6y_3 - 4y_4 &\geq -22 && (\#48)\\
7x_0 - 6x_1 - 2x_2 - 7x_3 - 7x_4 + 7y_0 - 4y_1 + y_2 + 3y_3 + 5y_4 &\geq -19 && (\#49)\\
-2x_0 - 2x_1 - 2x_2 - 2x_3 + x_4 + y_0 + 2y_1 - 2y_2 - y_3 + 2y_4 &\geq -9 && (\#50)\\
x_0 - 2x_1 - 2x_2 + 2x_3 + 2x_4 + y_0 + y_1 + 2y_2 - 2y_3 - 2y_4 &\geq -6 && (\#51)\\
5x_0 - 5x_1 - x_2 + 3x_3 - 5x_4 - 5y_0 + 2y_1 + 5y_2 + 2y_3 - 4y_4 &\geq -15 && (\#52)\\
8x_0 + 8x_1 + 4x_2 - 3x_3 - 3x_4 + 8y_0 + 4y_1 + 5y_2 + 5y_3 - 2y_4 &\geq 0 && (\#53)\\
-x_0 + x_1 + 4x_2 - 4x_3 - 4x_4 - y_0 + 4y_1 + 3y_2 + 4y_3 + 3y_4 &\geq -6 && (\#54)\\
10x_0 - 2x_1 - 3x_2 - 3x_3 + 10x_4 + 7y_0 - 10y_1 + y_2 - 8y_3 - 9y_4 &\geq -25 && (\#55)\\
-x_0 - 8x_1 - 8x_2 - 2x_3 + 4x_4 - 8y_0 + 6y_1 - 7y_2 - 7y_3 - 4y_4 &\geq -37 && (\#56)\\
-4x_0 + 4x_1 - 4x_2 + x_3 - 2x_4 + 3y_0 - 4y_1 + 3y_2 + 2y_3 - 3y_4 &\geq -13 && (\#57)\\
-x_0 + 3x_1 + 3x_2 + 9x_3 - x_4 - 9y_0 - 8y_1 + 6y_2 + 8y_3 - 9y_4 &\geq -19 && (\#58)\\
-3x_0 - 3x_1 + 3x_2 + 2x_3 + 3x_4 - y_0 + 3y_1 - y_2 + 3y_3 + 3y_4 &\geq -5 && (\#59)\\
-6x_0 - x_1 + 2x_2 - 6x_3 + 2x_4 - 5y_0 + 4y_1 + 6y_2 - 5y_3 - 4y_4 &\geq -21 && (\#60)
\end{aligned}
$$