INSTITUTE OF MATHEMATICS

THE CZECH ACADEMY OF SCIENCES

Representations of monotone Boolean functions by linear programs

Mateus de Oliveira Oliveira
Pavel Pudlák

Preprint No. 37-2016

PRAHA 2016

Representations of monotone Boolean functions by linear programs

(preliminary version)

Mateus de Oliveira Oliveira and Pavel Pudlák *

August 1, 2016

Abstract

We introduce the notion of *monotone linear program* (MLP) for computing (partial) Boolean functions. We prove that any function which is computable by a circuit whose gates are MLPs can also be computed by a single MLP whose size is linear in the size of the circuit. We show the following results.

1. MLPs are superpolynomially more powerful than polynomial-size monotone Boolean circuits.
2. MLPs are exponentially more powerful than monotone span programs.
3. We show that the Lovász-Schrijver proof system cannot be polynomially simulated by the cutting planes proof system. This is the first results that shows some separation of these two proof systems.
4. We introduce the notion of strong MLP gate and show that circuits constituted of this type of gate can be used to prove a monotone feasible interpolation theorem for Lovász-Schrijver proof systems. In some restricted cases it is also possible to interpolate by an MLP.

Finally, we discuss connections between the problem of proving lower bounds on the size of MLPs and the problem of proving lower bounds on extended formulations of polytopes.

1 Introduction

Superpolynomial lower bounds on the size of Boolean circuits computing concrete Boolean functions have only been proved for circuits from some specific families of circuits. A prominent role among these families is played by *monotone Boolean circuits.* Exponential lower

*Both authors were supported by the ERC Advanced Grant 339691 (FEALORA)

bounds on monotone Boolean circuits were proved already in 1985 by Razborov [23]. In 1997 Krajíček discovered that lower bounds on monotone complexity of particular partial Boolean functions can be used to prove lower bounds on resolution proofs [15]. Incidentally, the functions used in Razborov's lower bound were just of the form needed for resolution lower bounds. Exponential lower bounds on resolution proofs had been proved before (coincidentally about at the same time as Razborov's lower bounds). However, Krajíček came up with a new general method, the so called *feasible interpolation,* that potentially could be used for other proof systems. Indeed, soon after his result, this method was used to prove exponential lower bounds on the cutting-plane proof system [19, 12]. That lower bound is based on a generalization of Razborov's lower bounds to a more general monotone computational model, the *monotone real circuits*. Another monotone computational model for which superpolynomial lower bounds have been obtained is the *monotone span program* model [2, 9]. An exponential lower bound on the size of monotone span programs have been recently obtained in [5]. For a long time the best known lower bound for this model of computation was of the order of $n^{\Omega(\log n)}$ [2]. Again, superpolynomial lower bounds on the size of monotone span programs can be used to derive lower bounds on the degree of Nullstellensatz proofs, as shown in [21][1]

The results listed above suggest that proving lower bounds on stronger and stronger models of monotone computation may be a promising approach towards proving lower bounds on stronger proof systems. Motivated by this line of research we decided to study another model for computing monotone Boolean functions. We call this model *monotone linear programs*, or briefly MLPs. Essentially, a monotone linear program for a monotone Boolean function F is a linear program that has the input Boolean variables as parameters. There are several equivalent ways how to introduce the parameters into the linear program. The important thing is that they must appear positively, otherwise the program would be as efficient as general Boolean circuits. There are also several ways how to define which inputs are accepted and which are rejected. The two basic ways are: 1. we say that an input is accepted iff the linear program has a solution, 2. we add a linear objective function and say that an input is accepted iff the maximum is above a certain threshold. In the second case, we assume that the program has a solution for all inputs. There is also an equivalent version stated in terms of zero-sum games. An important property of monotone linear programs is that circuits built from them can be simulated by a single monotone linear program.

We introduce the notion of *monotone linear programming gates* (MLP gates). These are gates of the form $g(y) = \max\{c \cdot x \mid Ax \leq y\}$ where the input is a tuple y of real variables and the output is the optimum value of a linear program parameterized by y. The only constraint is that this linear program has a solution for every y. The complexity of such a gate is measured as the number of rows in A. We say that a circuit made of MLP gates is a *monotone linear programming circuit*. In Proposition 4.2 we show that each MLP circuit C can be collapsed into a single MLP gate g_C whose size is at most twice the sum of sizes of gates occurring in C. It can be shown that the AND and the OR functions can be simulated

[1]We note however that strong degree lower bounds for Nullstellensatz proofs can be proved using more direct methods [3, 4, 10, 1].

by MLP gates, implying in this way that monotone Boolean circuits can be polynomially simulated by MLP circuits (Theorem 5.1).

In fact, MLP circuits are super-polynomially stronger than monotone Boolean circuits. On the one hand, Razborov has shown that that any MBC computing the *bipartite perfect matching function* $\mathrm{BPM}_m : \{0,1\}^{m^2} \to \{0,1\}$ must have size at least $n^{\Omega(\log n)}$. On the other hand, a classical results in linear programming theory [25] implies that the same function can be computed by MLP circuits of polynomial size.

In [2], Babai, Gál and Wigderson showed that there is a function that can be computed by span programs of linear size but which require superpolynomial-size monotone Boolean circuits. On the other hand, recently Cook et al. [5] proved that there is a function which can be computed by polynomial-size monotone Boolean circuits, but which require exponential-size monotone span programs. Therefore, monotone span programs (which we will abbreviate by MSP) and monotone Boolean circuits are incomparable in the sense that none of these models can polynomially simulate the other. Dual MLPs are very similar to monotone span programs of over reals.[2] In Theorem 5.5 we show that dual MLPs can polynomially simulate monotone span programs over reals. On the other hand, by combining the results in [5] with Theorem 5.5, we have that dual MLPs are exponentially stronger than monotone span programs over reals. Therefore, while MBCs are incomparable with MSPs, dual MLPs are strictly stronger than both models of computation.

Next we turn to the problem of providing a monotone interpolation theorem for Lovász-Schrijver proof systems [17]. Currently, size lower bounds for these systems have been proved only with respect to tree-like proofs [18], and therefore, it seems reasonable that a monotone interpolation theorem for this system may be a first step towards proving size lower bounds for general LS proof systems. Towards this goal, we introduce the notion of strong monotone linear programming gate, and show that circuits constituted of this kind of gates can be used to provide an interpolation theorem for LS-proof system. Unlike the case of MLP circuits, we don't know how to collapse a circuit with strong MLP gates into a single MLP gate. However, in Theorem 6.2 we show that for some restricted type of LS proofs, we can effectively interpolate by MLPs. This interpolation theorem implies two things. First, the cutting-plane proof system cannot polynomially simulate the LS proof system. The mutual relation of cutting-plane proof system and LS proof system is a longstanding open problem. Our result is the first nontrivial piece of information concerning this problem. Second, using this interpolation theorem and Fu's lower bound [8], we can show that MLP-circuits cannot be polynomially simulated by monotone real circuits. (This can also probably be derived more directly by strengthening Fu's lower bound on monotone real circuits from general graphs to bipartite graphs.)

Monotone linear programs programs are, in a sense, generalizations of monotone Boolean circuits and monotone span programs. The lower bounds for monotone Boolean circuits and monotone span programs were proved by two different techniques. Therefore it will be necessary to develop a new lower bound method for proving superpolynomial lower bounds on monotone linear programs. A possible approach may be based on strengthening lower

[2]See subsection 3.4 for the definition of dual MLP.

bounds on extended formulations, which is a related, but apparently easier problem. A lower bound on extended formulation is a lower bound on the number of inequalities needed to define an extension of a polytope to some higher dimension. Such lower bounds have been proven, in particular, for polytopes spanned by the 0-1 vectors representing minterms of monotone Boolean functions. To prove a lower bound on a monotone linear program, it will be needed to prove lower bounds on extended formulations of all polytopes of a certain form that separate the minterms from the maxterms. This is clearly a much harder problem, but there are results on extended formulations that go in this direction. However, our Corollary 6.5 suggests that this will surely not be easy. It gives an example of a monotone function such that the set of ones has only exponentially large extended formulation, but the minterms can be separated from a large subset of maxterms by a polynomial size dual MLP.

Acknowledgment. We would like to thank Pavel Hrubeš and Massimo Lauria for discussion and their valuable suggestions.

2 Preliminaries

Monotone Partial Functions: A partial Boolean function is mapping $F : \{0,1\}^n \to \{0,1,*\}$. If $p \in \{0,1\}^n$ is such that $F(p) = *$ then we regard F as being undefined on p. Since all our definitions make sense for partial Boolean functions, we will often omit the word "partial". Let p and p' be two Boolean strings in $\{0,1\}^n$. We say that a partial function $F : \{0,1\}^n \to \{0,1,*\}$ is monotone if $F(p) = 1$ whenever $p \geq p'$ and $F(p') = 1$.

Linear Programs: We use the following conventions. Variables x, y, z are used to denote real vectors, while variables p, q, r are used to denote strings of Boolean variables. $A \in \mathbb{R}^{m \times k}$ means that A is a real matrix with m rows and n columns. For vectors x and y, $x \leq y$ means $x_i \leq y_i$ for all coordinates i; the same for matrices and Boolean strings. As an abuse of notation, we write 0 (1) to denote vectors in which all coordinates are equal to 0 (1). For two vectors x and y, we will denote their scalar product by $x \cdot y$.

A *polytope* is the convex hull of a nonempty finite set of vectors in $\mathbb{R}^n$; in particular, a polytope is *nonempty and bounded*. A linear program is an optimization problem of the form

$$\max\{c^T \cdot x \mid Ax \leq b,\ x \geq 0\}. \tag{1}$$

Where $A \in \mathbb{R}^{m \times k}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^k$. The dual of the linear program of Equation 1 is defined as follows.

$$\min\{y^T \cdot b \mid A^T y \geq c,\ y \geq 0\}. \tag{2}$$

According to the *Linear Programming Duality*,

$$\max c^T \cdot x = \min y^T \cdot b, \tag{3}$$

provided that the maximum and minimum exist.

Lemma 2.1 *If $Ax \leq b$ defines a polytope, then for every $b' \geq 0$, $Ax \leq b + b'$ defines a polytope. In particular, $Ax \leq b + Bp$ defines a polytope for every $p \in \{0,1\}^n$.*

3 Representations of monotone Boolean functions

In this section we will define several ways of representing monotone Boolean functions using certain types of linear programs. In what follows, A is a matrix in $\mathbb{R}^{m\times k}$, B is a matrix in $\mathbb{R}^{m\times n}$ with $B \geq 0$, and b is a vector in $\mathbb{R}^m$ and c is a vector in $\mathbb{R}^k$. We note that by removing the condition that B is non-negative, the models introduced below would be able to define arbitrary Boolean functions on n variables. Thus the condition $B \geq 0$ is added to enforce monotonicity.

Definition 1 (MLP-Representations) *Let $F : \{0,1\}^n \to \{0,1\}$ be a Boolean function.*

(MLP-1) *We say that a triple (A, B, b) is an* MLP-1 *representation of F if for each $p \in \{0,1\}^n$,*

$$F(p) = \begin{cases} 1 \;\Rightarrow\; \exists x \geq 0,\ Ax \leq b + Bp, \\ 0 \;\Rightarrow\; \neg\exists x \geq 0,\ Ax \leq b + Bp. \end{cases} \tag{4}$$

(MLP-2) *We say that a 4-tuple (A, B, b, c) is an* MLP-2 *representation of F if $Ax \leq b$ defines a polytope, and for each $p \in \{0,1\}^n$,*

$$F(p) = \begin{cases} 1 \;\Rightarrow\; \max\{c \cdot x \mid Ax \leq b + Bp,\ x \geq 0\} \geq 0, \\ 0 \;\Rightarrow\; \max\{c \cdot x \mid Ax \leq b + Bp,\ x \geq 0\} < 0. \end{cases} \tag{5}$$

(MLP-3) *We say that a 4-tuple (A, B, b, c) is an* MLP-3 *representation of F if $Ax \leq b$ defines a polytope, and for each $p \in \{0,1\}^n$,*

$$F(p) = \begin{cases} 1 \;\Rightarrow\; \max\{c \cdot x \mid Ax \leq b + Bp,\ x \geq 0\} = 0, \\ 0 \;\Rightarrow\; \max\{c \cdot x \mid Ax \leq b + Bp,\ x \geq 0\} < 0. \end{cases} \tag{6}$$

(MLP-4) *We say that a 4-tuple (A, B, b, c) is an* MLP-4 *representation of F if $Ax \leq b$ defines a polytope, and for each $p \in \{0,1\}^n$,*

$$F(p) = \begin{cases} 1 \;\Rightarrow\; \min\{(b + Bp) \cdot y \mid -A^T y \leq -c,\ y \geq 0\} \geq 0, \\ 0 \;\Rightarrow\; \min\{(b + Bp) \cdot y \mid -A^T y \leq -c,\ y \geq 0\} < 0. \end{cases} \tag{7}$$

In each version, we define the complexity of the representation to be $m \cdot k$, the number of rows of A times the number of columns. When the matrices A, B and the vector b are rational, the sizes of the rational numbers in them are, in general, important parameters, but for the sake of simplicity, we focus only on the number of inequalities. The following lemma, which is proved in Appendix A, states that the four versions of MLPs defined above are equivalent, in the sense that one can be converted into the other with a small increase in size.

Lemma 3.1 *Let $F : \{0,1\}^n \to \{0,1\}$. Then for each $i, j \in \{1,2,3,4\}$, F has an MLP-i representation of size S if and only if F has an MLP-j representation of size $O(S)$. Additionally, one may assume without loss of generality that the matrix B is a $0/1$ matrix with at most one 1 in each row.*

3.1 Representation by Labeled Matrices

In the MLP representations defined in the beginning of this section, the polytope $Ax \leq b+Bp$ is parameterized by the input variables p via the matrix B. Here we define another way of parameterizing the polytopes with respect to the input variables. In this version, some rows of the matrix A are labeled by the Boolean variables p_j and rows may have no label. Formally a labeling for a matrix A with m rows is a function $\rho : \{1, ..., m\} \to \{p_1, ..., p_n, *\}$. For each $i \in \{1, ..., m\}$, $\rho(i) = p_j$ indicates that the i-th row is labeled with p_i, while $\rho(i) = *$ indicates that the i-th row has no label. A labeled matrix is a pair $A^\rho = (A, \rho)$ consisting of a matrix A and a labeling ρ of its rows. For each assignment $w \in \{0,1\}^n$ of the variables in p, we let $A^\rho_{[w]}$ be the sub-matrix obtained from A by deleting all rows labeled by some variable whose value was set to 1. We note that the unlabeled rows remain in $A^\rho_{[w]}$ for each $w \in \{0,1\}^n$. This notation can be straightforwardly applied to vectors by considering them as $m \times 1$ matrices. That is a labeled vector is a pair $b^\rho = (b, \rho)$ and we write $b^\rho_{[w]}$ to denote the vector obtained from b by deleting coordinate i if and only if $\rho(i)$ is a variable that receives the value 1.

Definition 2 (MLP-5 Representation) *Let $F : \{0,1\}^n \to \{0,1\}$ be a Boolean function. We say that a pair (A^ρ, b^ρ) is an MLP-5 representation of F if for each assignment $w \in \{0,1\}^n$ of variables $p = (p_1, ..., p_n)$,*

$$F(w) = \begin{cases} 1 \ \to \ \exists x \geq 0, A^\rho_{[w]} x \leq b^\rho_{[w]} \\ \\ 0 \ \to \ \neg\exists x \geq 0, A^\rho_{[w]} x \leq b^\rho_{[w]} \end{cases} \tag{8}$$

The following proposition states that MLP-5 representations are equivalent to MLP-i representations for $i \in \{1, ..., 4\}$.

Proposition 3.2 *Let $F : \{0,1\}^n \to \{0,1\}$. Then F has an MLP-5 representation of size S if and only if for each $i \in \{1, ..., 4\}$, F has an MLP-i representation of size $O(S)$.*

Proof. By Lemma 3.1 it is enough to show prove the Lemma only for MLP-1 representations. Assume that the triple (A, B, b) is an MLP-1 representation of F and let $p = (p_1, ..., p_n)$ be the input variables of F. By Lemma 3.1 we may assume that B is a $0/1$ matrix with at most one 1 in each row. We replace every inequality of the form

$$a_i \cdot x \leq b_i + p_j,$$

with two inequalities

$$\begin{array}{lcll} a_i \cdot x & \leq & b_i & \text{with label } p_j \\ a_i \cdot x & \leq & b_i + 1 & \text{without a label.} \end{array}$$

It is straightforward to check that the system of labeled inequalities obtained in this way corresponds to an MLP-5 representation of F.

For the converse, let (A^ρ, b^ρ) be an MLP-5 representation of F. Then just replace each inequality $a_i \cdot x \leq b_i$ labeled with p_j, by an inequality $a_i \cdot x \leq b_i + \alpha p_j$ where $\alpha \in \mathbb{R}$ is a number which is large enough to make the inequality irrelevant when $p_j = 1$. The system of inequalities obtained in this way corresponds to an MLP-1 representation of F. ■

3.2 Representation by Zero-Sum Games

A zero-sum game is defined by a matrix $A \in \mathbb{R}^{m \times k}$. This game has two players: a *Row Player* and a *Column Player*. A strategy for the Row Player is a vector $u \in \mathbb{R}^m$, with $u \geq 0$ and $|u|_1 = 1$ (that is, a probability distribution on $\{1, \ldots, m\}$). Similarly, a strategy for the *Column Player* is a vector $v \in \mathbb{R}^k$ with $v \geq 0$ and $|v|_1 = 1$ (that is, a probability distribution on $\{1, \ldots, k\}$). Such a strategy is *pure* if all weight is placed in a unique coordinate. Given strategies u, v for the two players, the *payoff* of the game defined by A when Row Player plays strategy u and Column Player plays strategy v is defined as $u^T A v$. The payoff of a strategy u for the Row Player is defined as $\min_v u^T A v$, while the payoff of a strategy v for Column Player is defined as $\max_u u^T A v$. We say that a strategy u is a *winning strategy* for Row Player, if for every strategy v of Column Player we have $u^T A v < 0$. On the other hand, a strategy v is a *winning strategy* for Column Player, if for every strategy u of Row Player we have $u^T A v > 0$.

Let $p = (p_1, ..., p_n)$ be Boolean variables. A double-labeled matrix is a triple $A^{\rho,\gamma} = (A, \rho, \gamma)$ where A is a $m \times k$ real matrix, $\rho : \{1, ..., m\} \to \{p_1, ..., p_n, *\}$ is a labeling of the rows of A and $\gamma : \{1, ..., k\} \to \{p_1, ..., p_n, *\}$ is a labeling of the columns of A. We say that a row i (column j) is unlabeled if $\rho(i) = *$ ($\gamma(j) = *$). For each assignment $w \in \{0,1\}^n$ of the variables in p, we denote by $A^{\rho,\gamma}_{[w]}$ the sub-matrix of A which is obtained by deleting rows labeled with variables that are set to 1, and by deleting columns labeled with variables that are set to 0 at the assignment w.

Definition 3 (Zero-Sum Representation) *Let $F : \{0,1\}^n \to \{0,1\}$ be a Boolean function on variables $p = (p_1, ..., p_n)$. We say that that a double-labeled matrix $A^{\rho,\gamma}$ is a zero-sum game representation of F if for every assignment $w \in \{0,1\}^n$,*

$$F(w) = \begin{cases} 1 \ \rightarrow \ \textit{Column Player has a winning strategy for the game } A^{\rho,\gamma}_{[w]}. \\ 0 \ \rightarrow \ \textit{Row Player has a winning strategy for the game } A^{\rho,\gamma}_{[w]}. \end{cases} \tag{9}$$

We note that the asymmetry in the way in which rows and columns are deleted guarantees that the function F is monotone. Intuitively, by setting a variable p_i to 1, Row Player is not anymore allowed to use the rows labeled with p_i and Column Player is now allowed to use the columns labeled with p_i. Therefore the space of strategies of Row Player shrinks, while the space of strategies of Column Player gets expanded. In this way, the payoff for Column player is at least as large as if the variable p_i were set to 0.

The next proposition states that zero-sum game representations are equivalent to MLP representations. However, we believe that it is worth to consider zero-sum game representations as a separate concept because they seem to be more amenable for the application of lower-bound techniques based on communication complexity theory. The idea is that the variables p may be split into two disjoint groups p' and p'' in such a way that rows are only labeled with variables from p', and columns are only labeled with variables from p''. This corresponds to the setting in which we want to compute a function $F(p)$ where the variables in p' are in possession of Row player while variables in p'' are in possession of Column Player.

We note that from the point of view of expressiveness, the way in which variables are distributed among Row Player and Column Player is irrelevant. More precisely, by making appropriate modifications to a double-labeled matrix $A^{\rho,\gamma}$, one can always transform a row label into a column label, and vice versa, in such a way the resulting double-labeled matrix $A^{\rho',\gamma'}$ represents the same function. Additionally, one can always consider that each variable labels at most one row and at most one column.

Proposition 3.3 *A function $F : \{0,1\}^n \to \{0,1\}$ has a zero-sum game representation $A^{\rho,\gamma}$ of size S if and only if it has an MLP-1 representation of size $O(S)$. Additionally, the same statement holds if we assume that all rows (all columns) are unlabeled, as well as if we assume that each variable labels at most one row and at most one column.*

Proof. Let $A^{\rho,\gamma}$ be a zero-sum game representation of a function F in which no column is labeled. In other words, $\gamma(j) = *$ for each column of A. Then this matrix can be viewed as a single-labeled matrix A^{ρ}. It should be clear that the system of inequalities

$$\begin{array}{c} A^{\rho}x \leq 0, \\ \sum_{j=1}^{k} x_j = 1 \end{array} \tag{10}$$

is an MLP-5 representation of F.

For the converse, let (A^{ρ}, b) be an MLP-5 representation of F, and assume that the corresponding system of inequalities

$$A^{\rho} \leq b^{\rho} \tag{11}$$

contains the unlabeled equality $\sum_j x_j = 1$ (which is represented by two unlabeled inequalities $\sum_j x_j \leq 1$ and $\sum_j x_j \geq 1$). This assumption will be removed later. Then by adding appropriate multiples of the inequality $\sum_j x_j = 1$ to each row, System 11 can be transformed into a system of the form

$$\begin{array}{c} (A')^{\rho}x \leq 0^{\rho} \\ \sum_{j=1}^{k} x_j = 1 \end{array} \tag{12}$$

Such that for each $w \in \{0,1\}^n$, the system $A^{\rho}_{[w]} \leq b^{\rho}_{[w]}$ has a solution if and only if the system

$$\begin{array}{c} (A')^{\rho}_{[w]}x \leq 0^{\rho}_{[w]}, \\ \sum_{j=1}^{k} x_j = 1 \end{array} \tag{13}$$

has a solution. Additionally, we have the following immediate claim.

Claim 1 *For each $w \in \{0,1\}^n$, System 13 has a solution if and only if Column Player has a strategy to get payoff ≥ 0 in the zero-sum game defined by $(-A')^{\rho}_{[w]}$.*

Now let ε be a small enough positive number, and let A'' be the matrix obtained from by adding ε to each entry of $-A'$. Then we have that for each $w \in \{0,1\}^n$, Column Player gets a payoff ≥ 0 in the game $(-A)^{\rho}_{[w]}$ if and only if Column player gets a payoff > 0 in the game $(A'')^{\rho}_{[w]}$. Therefore, if we let γ be a labeling of the columns of A'' such that $\gamma(j) = *$ for every $j \in \{1, ..., k\}$. The double-labeled matrix $(-A')^{\rho,\gamma}$ is a zero-sum game representation of F.

Now assume that the equality $\sum_j x_j = 1$ does not belong to the system of inequalities $A^{\rho}x \leq b^{\rho}$ defined by the MLP-5 representation (A^{ρ}, b^{ρ}). First we select a large enough positive real number α such that for every p for which there exists a solution x, we have $\sum_j x_j \leq \alpha$. Then if we take a (dummy) variable x_{k+1} and add the equality $\sum_{j=1}^{k+1} x_j/\alpha = 1$, the new system has a solution if and only if the old one has. Finally, we make a change of variables by setting $y_j := x_j/\alpha$ for each $j \in \{1, ..., k\}$ and by setting $y_{j+1} := x_{j+1}$. Clearly, the new system of inequalities on variables y_j has a solution if and only if the old one has, and now the equation $\sum_{j=1}^{k+1} y_j = 1$ belongs to the system.

Now we show that the way in which variables are distributed among Row Player and Column Player is immaterial. Assume that p_j labels some columns of $A^{\rho,\gamma}$. Add a new row to A which has -1 on the columns labeled with p_j and 0 elsewhere. Label this new row with p_j, and remove the labels p_j from the columns. Let $(A')^{\rho',\gamma'}$ be the matrix obtained by this process. Let w be an assignment of the variables in p. If p_j is set to 1, then the new added row is not present in the matrix $(A')^{\rho',\gamma'}_{[w]}$, and therefore the column player is free to chose the columns that were labeled by p_j in the original matrix A. On the other hand, if p_j is set to zero, then the row player has a strategy with payoff < 0 for any strategy of Column Player that sets non-zero weight in some column that was previously labeled by p_j. Hence, in such case, any winning strategy for Column player must put weight 0 in these columns. A symmetric argument shows that row labels can be transformed into column labels. In this case, the difference is that in this case we create a new column which has 1 in every row labeled by p_j. Subsequently we label this new column with p_j, and remove the label p_j from the rows. Then, if $p_j = 1$, any winning strategy for Row Player must set weight 0 on all rows that were previously labeled by p_j. Note that in either case, a unique row or column labeled by p_j is created. □ ■

3.3 Sharp representation

Definition 4 (Sharp MLP representation) *Suppose $Ax \leq b$ defines a polytope. Then (A, B, b, c) is a sharp representation of a Boolean function $F : \{0,1\}^n \to \{0,1\}$ if for each $p \in \{0,1\}^n$,*

$$F(p) = \begin{cases} 1 \;\Rightarrow\; \max\{c \cdot x \mid Ax \leq b + Bp\} = 1, \\ 0 \;\Rightarrow\; \max\{c \cdot x \mid Ax \leq b + Bp\} = 0. \end{cases} \tag{14}$$

This computational model seems to be weaker—we do not know how to efficiently transform a general MLP representation into a sharp one. In the definition of MLP-3 representation we have the value of the maximum when $F(p) = 1$. Nevertheless we do not know how to fix both values without increasing exponentially the size of the representation.

3.4 Dual monotone functions and dual representations

Let F be a partial monotone function. The dual monotone function F^d is the function defined by $F^d(p_1, \ldots, p_n) := \neg F(\neg p_1, \ldots, \neg p_n)$. We do not know how to get an efficient MLP representation of F^d from a representation of F. We note that if (A, B, b, c) is an MLP-2 representation of F then the MLP-4 representation of F obtained by linear programming duality computes the same function F, and not the function F^d. However, it is possible to define a notion of dual MLP representation that indeed computes the dual function. This is possible for all versions and modifications of MLPs. Here we give only one.

Definition 5 (Dual MLP Representation) *Suppose $Ax \geq b$ defines a polytope. Then a tuple (A, B, b, c) is a dual MLP representation of a Boolean function F if for every $p \in \{0,1\}^n$,*

$$F(p) = \begin{cases} 1 \;\Rightarrow\; \min\{c \cdot x \mid Ax \geq b + Bp\} > 0 \\ 0 \;\Rightarrow\; \min\{c \cdot x \mid Ax \geq b + Bp\} < 0 \end{cases} \tag{15}$$

Proposition 3.4 *If a monotone Boolean function F has a representation by an MLP of size S, then F^d has a dual representation by an MLP of the same size.*

Proof. Let (A, B, b, c) be an MLP-2 representation of F. Then we have

$$F(\neg p_1, \ldots, \neg p_n) = \tfrac{1}{2}\mathrm{sgn}\{\max c \cdot x \mid Ax \leq b + B(\bar{1} - p)\} + \tfrac{1}{2}.$$

Hence

$$F^d(p_1, \ldots, p_n) = \frac{1}{2}\mathrm{sgn}\{\min -c \cdot x \mid -Ax \geq -b - B\bar{1} + Bp\} + \frac{1}{2}.$$

■

4 Circuits with Monotone Linear Programming Gates

Monotone linear programs can be used to represent a wider class of functions, not just monotone Boolean functions. Furthermore, they can be combined into circuits. Our aim is to show that such circuits can be transformed into a single monotone linear program.

4.1 Maximization and Minimization Gates

We will extend our definition of functions computed by monotone linear program so that it can be applied to arbitrary real numbers. Since we want combine these functions into circuits, we will call the more general representation a *monotone linear programming gate*.

There are two kinds of such gates, maximization and minimization gates. If a monotone linear program consist solely of maximization gates, it is easy to combine the gates into a single one. If the circuit has some minimization gates we must first replace them by maximization gates.

An $m \times n$ maximization MLP gate is a function $\ell : \mathbb{R}^m \to \mathbb{R}$ of the form

$$\ell(y) = \max_{Ax \leq y} c \cdot x \tag{16}$$

where A is an $m \times n$ real matrix, $y = (y_1, ..., y_m)$ is a tuple with m *input variables*, $x = (x_1, ..., x_n)$ is a tuple with n *internal variables*, and $c = (c_1, ..., c_n)$ is a tuple with n real *constants*.

An $n \times m$ minimization gate is a function $f : \mathbb{R}^m \to \mathbb{R}$ of the form.

$$\ell(y) = \min_{Bx \geq c} y \cdot x \tag{17}$$

where B is an $n \times m$ matrix, $y = (y_1, ..., y_m)$ is a tuple with m *input variables*, $x = (x_1, ..., x_m)$ is a tuple with m *internal variables*, and $c = (c_1, ..., c_n)$ is a tuple with n real *constants*. Note that there is an asymmetry between maximization and minimization gates, in the sense that in a maximization gate ℓ, the input variables y are part of the constraints of the linear program, while in a minimization gate ℓ, the input variables y belong to the objective function of the linear program.

By the linear programming duality, a maximization gate given by (A, c) computes the same partial function as the minimization gate given by (A^T, c).

Proposition 4.1 *An MLP gate defines a partial function defined on a polyhedron (if it is nonempty). It is piecewise linear and concave.*

Proof. The first fact is obvious. To prove that ℓ, given by (A, c), is piecewise linear and concave, let us represent it by

$$\ell(y) = \max\{x_0 \mid \exists x,\ Ax \leq y, x_0 \leq c \cdot x\},$$

where x_0 is a new variable. Let P be the projection of the polyhedron defined by the inequalities on the space of the coordinates x and x_0. Then the graph of the function ℓ (i.e., the set of pairs $(x, \ell(x))$ for which $\ell(x)$ is defined) is formed by the upper facets of P, where upper means in the direction $x_0 \to \infty$. Since P is a polyhedron, ℓ is piecewise linear and concave. ∎

4.2 Monotone Linear Programming Circuits

The definition of linear-programming circuits is fairly standard. Nevertheless, we will define it in detail, in order to introduce a suitable notation that we will need later.

A *linear-programming circuit* is a labeled directed acyclic graph $C = (V, E, \hat{\ell}, \xi)$ where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, $\hat{\ell}$ is a function that labels each vertex v in V with a linear programming gate $\hat{\ell}(v)$, and ξ is a function that labels each edge $(v, v') \in E$ with a positive integer in such a way that the following conditions are satisfied.

1. If v is a source (a vertex without input edges), then $\hat{\ell}(v)$ is either a real number, or a variable.
2. For each $v \in V$, if v has k input edges, then these edges are injectively labeled by ξ with numbers from $1, ..., k$. Additionally, $\hat{\ell}(v)$ is either an $m \times k$ minimization gate for some $m \in \mathbb{N}$, or a $k \times n$ maximization gate for some $n \in \mathbb{N}$.
3. C has a unique sink (a vertex with no outgoing edges). This vertex, denoted by $\mathrm{Out}(C)$, is called the output of C.

The numbering of the edges of the circuit is used to determine which input edge of a vertex v corresponds to which input variable of the gate $\hat{\ell}(v)$. The variables labeling minimal vertices of C are called the input variables of C. Let C be a circuit with input variables $u = (u_1, ..., u_r)$. We denote by $C(a_1, ..., a_r)$ the circuit which is obtained from C by initializing each input variable u_i with the value $a_i \in \mathbb{R}$. The value $\mathrm{val}(v)$ of a vertex v in $C(a_1, ..., a_r)$ is inductively defined as follows: If v is an input vertex, then the value of v is the real number $\hat{\ell}(v)$ associated with v. Now assume that the value of each vertex of $C(a_1, ..., a_r)$ at depth at most d has been determined, and let v be a vertex at depth $d + 1$. Let $(v_1, v), ..., (v_k, v)$ be the input edges of v, where for each $i \in \{1, ..., k\}$, $\xi(v_i, v) = i$. Then the value of v is defined as

$$\mathrm{val}(v) = \hat{\ell}(v)(\mathrm{val}(v_1), ..., \mathrm{val}(v_k)).$$

Note that in general the values of some gates may be undefined, since the gates only compute partial functions. In such a case the computation is undefined.

Let $\mathrm{Out}(C)$ be the output vertex of C. The value $\mathrm{val}(C(a_1, ..., a_r))$ of the initialized circuit $C(a_1, ..., a_r)$ is defined as the value of its output vertex: $\mathrm{val}(C(a_1, ..., a_r)) = \mathrm{val}(\mathrm{Out}(C))$.

A linear programming circuit with r input variables $u = (u_1, ..., u_r)$ defines a partial function $f_C : \mathbb{R}^r \to \mathbb{R}$ as follows.

$$f_C(u_1, ..., u_r) = \mathrm{val}(C(u_1, ..., u_r)). \tag{18}$$

4.3 Reducing an MLP circuit to a single MLP gate

Proposition 4.2 *Let $C = (V, E, \hat{\ell}, \xi)$ be a linear-programming circuit with input variables y. Then one can construct a maximization gate ℓ defined by a linear program L of size $O(|C|^2)$ such that $\ell(y) = f_C(y)$.*

Proof. Let $C = (V, E, \hat{\ell}, \xi)$ be a linear programming circuit, and let v be a vertex of C. W.l.o.g. we may assume that all gates are maximization MLP gates. We denote by $\ell^v(y^v) = \max\{c^v \cdot x^v \mid A^v \cdot x^v \leq y^v\}$ the $m_v \times n_v$ maximization gate which labels v. We denote by $pre(v)$ the set of all vertices of C which reach the vertex v. Let $v_{j_1}, ..., v_{j_r}$ be the input vertices in $pre(v)$. Define the following linear program.

$$\begin{array}{lll} L^v_C(y_1^{v_{j_1}}, ..., y_1^{v_{j_r}}) := & \max\ c^v \cdot x^v & \text{subject to} \\ & \sum_{j=1}^{n_u} A^u_{ij} x^u_i \leq y^u_i, & \text{for } u \in pre(v),\ \ 1 \leq i \leq m_u \\ & y^u_i \leq c^w \cdot x^w, & \text{for } (w,u) \in E \cap pre(v) \times pre(v),\ \xi(w,u) = i \end{array} \tag{19}$$

In words, the objective function of L^v_C is the same objective function of the linear programming gate labeling v, and the constraints of L^v_C are formed by the sets of all inequalities associated with vertices in $pre(v)$, together with a new constraint $x^u_i = c^w \cdot x^w$ for each edge $(w,u) \in E \cap pre(v) \times pre(v)$ labeled with i. Intuitively, if w is the i-th in-neighbor of u, then the i-th input variable y^u_i of the gate ℓ^u is identified with the objective function $c^w \cdot x^w$ of the gate ℓ^w.

Claim 2 *Let $L^v_C(y^{v_{j_1}}, ..., y^{v_{j_r}})$ be the linear program associated with v. Let $a \in \{0,1\}^n$ be an assignment of the input variables of C. Let a_{j_1}, ..., a_{j_r} be the values assigned to the variables $y^{v_{j_1}}, ..., y^{v_{j_r}}$. Then $VAL(L^v_C(a_1, ..., a_r)) = \mathrm{val}(v)$.*

In particular, this claim implies that for each input $a \in \{0,1\}^n$ the value of the circuit $C(a)$ at its output vertex v^*, i.e. $\mathrm{val}(v^*)$, is equal to the value of the linear program $L^{v^*}_C(a)$. The proof of this claim proceeds by induction on the depth of v. In the base case, v is an input vertex. In this case the claim is trivial. Now assume that the claim is valid for every vertex of depth at most d, and let v be a vertex of depth $d+1$. Let $w^1, ..., w^r$ be the input vertices of v, where for each $i \in \{1, ..., r\}$, the edge (w^i, v) is labeled with i. Then the constraints of of L^v_C consist of the union of all constraints in $L^{w_1}_C, ..., L^{w_r}_C$ together with the following constraints.

$$\begin{array}{ll} \sum_{j=1}^{n_v} A^v_{ij} x^v_i \leq y^v_i & 1 \leq i \leq m_v \\ y^v_i \leq c^{w_i} \cdot x^{w_i} & \end{array} \tag{20}$$

Additionally, $\mathrm{val}(L^v_C) = \max c^v \cdot x^v$. Since $\mathrm{val}(L^v_C)$ is monotone on on all variables y^v_i, and since y^v_i is by the induction hypothesis the maximum value that $c^{w_i} \cdot x^{w_i}$ can attain, we have that $L(^v_C)$ is maximized when $y^v_i = \mathrm{val}(w_i)$. Therefore, by the definition of evaluation for a circuit with linear programming gates, we have that $\mathrm{val}(L^v_C) = \mathrm{val}(v)$. □ ■

5 Monotone Linear Programs vs Monotone Boolean Circuits

In this section we show that monotone linear programs simulate monotone Boolean circuits, but not vice versa.

Theorem 5.1 *Given a monotone Boolean circuit of size S, one can construct a monotone linear program of size at most $O(S)$ that sharply represents the same monotone Boolean function.*

Proof. The gates $\wedge, \vee$ can be represented by

$$p_1 \wedge p_2 \;=\; \max\{x \mid x \le p_1,\ x \le p_2\},$$

$$p_1 \vee p_2 \;=\; \max\{x_1 + x_2 \mid x_1 \le p_1,\ x_2 \le p_2,\ x_1 + x_2 \le 1\}.$$

Since these two programs are sharp representations (the gates output 0s and 1s), we can combine them into an MLP circuit that simulates the given Boolean circuit. By Proposition 4.2, we can transform the circuit into a single monotone linear program. ∎

Let BPM_n be the Boolean function defined on $\{0,1\}^{n^2}$ that takes on value 1 iff the vector $p \in \{0,1\}^{n^2}$ represents a bipartite graph with a perfect matching. Using a well-known result, we will show that BPM_n has a small representation by an MLP.

Theorem 5.2 (Schrijver [25], Corollary 8.6a) *The perfect matching polytope of a bipartite graph $E \subseteq I \times J$ is determined by the following inequalities*

1. $x \ge 0$,
2. $\sum_{j;(i,j)\in E} x_{ij} = 1,\ i \in I$,
3. $\sum_{i;(i,j)\in E} x_{ij} = 1,\ j \in J$.

Corollary 5.3 *BPM_n has a representation by a monotone linear program of polynomial size.*

Proof. Consider the following set of inequalities.

1. $x \ge 0$,
2. $\sum_j x_{ij} = 1,\ i \in I$,
3. $\sum_i x_{ij} = 1,\ j \in J$,
4. $x \le p$.

Clearly, every assignment to p that represents a graph with a perfect matching satisfies the inequalities. By Theorem 5.2, every solution x of 1.-3. is a convex combination of matchings. Thus if x, p is a solution of 1.-4. then $\epsilon x' \leq p$ for some matching x' and $\epsilon > 0$. So 1.-4. is a 1-MLP for BPM_n. ∎

Recall that Razborov proved an $n^{\Omega(\log n)}$ on the size of monotone Boolean circuits computing the BPM_n function [23]. Thus we have:

Corollary 5.4 *MLPs cannot be polynomially simulated by monotone Boolean circuits.*

For monotone formulas the gap is even exponential, as Raz and Wigderson proved a linear lower bound on the depth of monotone Boolean circuits for BPM_n [22].

5.1 Monotone Span Programs

Monotone span programs (MSP) were introduced by Karchmer and Wigderson [14]. Monotone span programs are defined over any field K. Such a program is given by a vector vector $c \in K^k$ and a labeled matrix $A^\rho = (A, \rho)$ where $A \in K^{m \times k}$, and $\rho : \{1, ..., m\} \to \{p_1, ..., p_n, *\}$ labels rows in A with variables in p_i or with the symbol $*$ (meaning that the row is unlabeled). For an assignment $p := w$, let $A^\rho_{\langle w \rangle}$ be the matrix obtained from A by deleting all rows labeled with variables which are set to 0.[3] A span program (A^ρ, c) represents a Boolean function F if

$$F(w) = 1 \;\Leftrightarrow\; \exists y,\; y^T A^\rho_{\langle w \rangle} = c^T, \tag{21}$$

that is, $F(p) = 1$ iff c is in the span of the rows of $A_{\langle w \rangle}$. In this section we show that dual MLPs polynomially simulate MSPs over the reals and for some functions are exponentially stronger. In fact, an equivalent definition of a dual MLP is very similar to the definition of MSP:

$$F(w) = 1 \;\Leftrightarrow\; \exists y \geq 0,\; y^T A^\rho_{\langle w \rangle} = c^T.$$

The formal difference is only in the additional condition $y \geq 0$, but it makes a huge difference in the power of the computational model. The fact that this definition is equivalent to other definitions of dual MLPs can be proved in the same manner as the equivalence of the MLP-5 representation to MLP-1 in Subsection 3.1.

Proposition 5.5 *Let $F : \{0,1\}^n \to \{0,1\}$ be a Boolean function. If F has a monotone span program representation of size S, then F has a dual MLP representation of size $O(S)$.*

Proof. Given a monotone span program (A, ρ, c), take the matrix A' that for each row a_i of A has both a_i and $-a_i$, assign the same labels to the negated rows, and let c be the same. It is clear that such a dual MSP, as defined above, computes the same function. ∎

[3] The difference between $A^\rho_{\langle w \rangle}$ and $A^\rho_{[w]}$ is that in the latter we delete rows labeled with variables that are set to 1.

Recently, it was shown in [5] that there is a family of functions $\mathrm{GEN}_n : \{0,1\}^n \to \{0,1\}$ which can be computed by polynomial-size monotone Boolean circuits but which require monotone span programs over the reals of size $\exp(n^{\Omega(1)})$. Since, by Theorem 5.1, MLPs, as well as dual MLPs, can polynomially simulate monotone Boolean circuits, we have an exponential separation between MLP representations and monotone span program representations.

Corollary 5.6 *Monotone span programs over the reals cannot polynomially simulate monotone linear programs or dual monotone linear programs.*

6 Lovász-Schrijver and Cutting-Plane proof systems

6.1 The Lovász-Schrijver proof system

Lovász-Schrijver proof system is a refutation system based on the Lovász-Schrijver method for solving integer linear programs [17]. During the past two decades several variants (probably nonequivalent) of this system have been introduced. In this work we will be only concerned with the basic system LS. In Lovász-Schrijver systems the domain of variables is restricted to $\{0,1\}$, i.e., they are Boolean variables. Given a unfeasible set of inequalities Φ, the goal is to use the axioms and rules of inference defined below to show that the inequality $0 \geq 1$ is implied by Φ.

- Axioms:
  1. $0 \leq p_j \leq 1$
  2. $p_i^2 - p_i = 0$ (integrality).
- Rules:
  1. *positive linear combinations;*
  2. *multiplication:* from a *linear inequalities*

     $$\sum_i c_i p_i - d \geq 0,$$

     derive

     $$p_i(\sum_i c_i p_i - d) \geq 0 \quad \text{and} \quad (1 - p_i)(\sum_i c_i p_i - d) \geq 0.$$
  3. *weakening rule:*

     From $\sum_i c_i p_i - d \geq 0$, derive $\sum_i c_i p_i - d' \geq 0$ for any $d' < d$.

Note that we may produce positive linear combinations from quadratic inequalities, but we can only apply the multiplication rule to linear inequalities. Axiom (2) correspond to two inequalities, but it suffices to use $p_i^2 - p_i \geq 0$, since the other inequality $p_i^2 - p_i \leq 0$ follows from Axiom (1) and Rule (2). We also observe that the weakening rule is derivable using only $0 \leq p_i \leq 1$ and linear combinations.

The LS proof system is implicationally complete which means that if an inequality $\sum_i c_i p_i - d \geq 0$ is semantically implied by an initial set of inequalities Φ, then $\sum_i c_i p_i - d \geq 0$ can be derived from Φ by a sequence of LS-rules [17].

Lower bounds on the size of LS proofs were only proved for the tree-like proofs [18]. For dag-like proofs, it is still an open problem to prove superpolynomial lower bounds. LS proof system is stronger than Resolution: it polynomially simulate Resolution and it proves the Pigeon Hole Principle by polynomial size proofs, whereas Resolution requires exponential size to prove the Pigeon Hole Principle [11]. The relation between LS and Cutting-Planes with respect to strength is still not known.

In this paper we will consider general (i.e., DAG-like) proofs. Thus a proof Π of $\sum_i c_i p_i - d \geq 0$ from Φ is a sequence of inequalities such that every inequality in the sequence is either an element of Φ or is derived from previous ones using some LS rule. We say that Π is a refutation of the set of inequalities Φ, if the last inequality is $-d \geq 0$ for some $d > 0$.

6.2 Feasible interpolation

Feasible interpolation is a method that can sometimes be used to translate circuit lower bounds into lower bounds for the size of refutations of Boolean formulas and linear inequalities. Let $\Phi(p, q, r)$ be an unsatisfiable Boolean formula which is a conjunction of formulas $\Phi_1(p, q)$ and $\Phi_2(p, r)$ where q and r are disjoint sets of variables. Since $\Phi(p, q, r)$ is unsatisfiable, it must be the case that for each assignment a of the variables p, either $\Phi_1(a, q)$ or $\Phi_2(a, r)$ is unsatisfiable, or both. Given a proof Π of unsatisfiability for $\Phi(p, q, r)$, an *interpolant* is a Boolean circuit $C(p)$ such that for every assignment a to the variables p,

1. if $C(a) = 0$, then $F_1(x, a)$ is unsatisfiable, and
2. if $C(a) = 1$, then $F_2(y, a)$ is unsatisfiable.

If both formulas are unsatisfiable, then $C(a)$ can be either of the two values. It was shown in [15] that given a resolution refutation Π of a CNF formula, one can construct an interpolant $C(p)$ whose size is polynomial in the size of Π. Krajíček's interpolation theorem has been generalized, by himself and some other authors, to other proof systems including cutting-planes proof system and the Lovász-Schrijver proof system [6].

In principle such *feasible interpolation* theorems could be used to prove lower bounds on the size of proofs if we could prove lower bounds on circuits computing some particular functions. But since we are not able to prove essentially any lower bounds on general Boolean circuits, feasible interpolation gives us only conditional lower bounds. For instance, the assumption that $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{coNP}$, an apparently weaker assumption than $\mathbf{NP} \neq \mathbf{coNP}$, implies that proofs in a system with feasible interpolation cannot be polynomially bounded.

However, in some cases one can show that there exist monotone interpolating circuits of polynomial size provided that all variables p appear positively in $\Phi_1(p,q)$, (or negatively in $\Phi_2(p,r)$). In the case of resolution proofs such circuits are simply monotone Boolean circuits [15, 16]. In the case of cutting planes one can interpolate using *monotone real circuits* [19]. Monotone real circuits are circuits have Boolean inputs and outputs, but they compute with real numbers. The gates are any monotone binary functions. Razborov's lower bound on the clique function has been generalized to monotone real circuits [19, 12]. Another proof system for which one can prove superpolynomial lower bounds using monotone feasible interpolation is the Nullstellensatz Proof System [21]. In this calculus the monotone interpolants are computed by monotone span programs[4] [21].

The results mentioned above suggest that if a proof system has the feasible interpolation property, then it may also have monotone feasible interpolation property for a suitable kind of monotone computation. We will show that the Lovász-Schrijver proof system has the monotone feasible interpolation property with the interpolants computed by circuits with strong MLP gates.

Definition 6 (Strong MLP Representation) *Suppose $Ax \leq b$ defines a polytope and let B and C be nonnegative matrices. Then we define a Boolean function by*

$$F(p) = \begin{cases} 0 \;\Rightarrow\; \max\{(c+Cp)\cdot x \mid Ax \leq b + Bp\} < 0 \\ 1 \;\Rightarrow\; \max\{(c+Cp)\cdot x \mid Ax \leq b + Bp\} > 0 \end{cases} \tag{22}$$

and say that this is a strong MLP representation.

In the same way we define a *strong MLP gate*. Namely, this is the real function

$$\ell(y,z) := \max\{y \cdot x \mid Ax \leq z\}.$$

A strong MLP gate can compute a quadratic function, while the ordinary MLP gates only compute piece-wise linear functions. Therefore, we believe that strong representations of Boolean functions are more efficient than the basic ones. Furthermore, it seems unlikely that circuits made of strong MLP gates can be easily reduced to a single strong MLP program. Hence proving lower bounds for this model of computation seems much harder.

6.3 Feasible interpolation for the Lovász-Schrijver system

A natural way to view a proof, or a refutation in LS as a sequence of *linear inequalities* $L_1 \geq 0, \ldots, L_m \geq 0$ such that for every $i = 1, \ldots, m$, L_i is a linear combination of initial inequalities, previous inequalities $L_j \geq 0$, $j < i$, products of pairs of previous inequalities $L_j \cdot L_k \geq 0$, $j, k < i$, and the axioms $0 \leq p_i \leq 1$, $p_i^2 - p_i = 0$. In other words, in means to view quadratic inequalities only as an auxiliary means to derive the next linear inequality. This is sometimes referred to as *lift and project*, which means that we temporarily lift equations

[4] In the context of polynomial calculus, alternative methods (e.g. [1, 13]) yield stronger lower bounds than the monotone interpolation technique.

to higher dimension, where we consider $x_j x_k$, $1 \leq j < k \leq n$, and then we project back to x_j, $j = 1, \ldots, n$. We will also call these steps *lift-and-project steps* or simply *lap-steps*.

We will show that one *lap*-step can be simulated by a strong MLP in a particular way. We will use this simulation in the proof of the theorem below.

Let $F_j(q)$, $j \in J$ and $F(q)$ be linear forms in variables q, and let c_j, $j \in J$ and c be constants. Suppose that the inequality $F(q) - e \geq 0$ was derived from $F_j(q) - c_j \geq 0$, $j \in J$ in one *lap*-step. So we have

$$F(q) - c =$$
$$\sum_{ij} \alpha_{ij} q_i (F_j(q) - c_j) + \sum_{ij} \beta_{ij} (1 - q_i)(F_j(q) - c_j) +$$
$$\sum_{j} \gamma_j (F_j(q) - c_j) +$$
$$\sum_{i} \delta_i (q_i - q_i^2) +$$
$$\sum_{j} \xi_j (F_j(q) - c_j),$$

for some $\alpha_{ij}, \beta_{ij}, \gamma_j, \delta_i, \xi_j \geq 0$. Suppose that the linear forms $F_j(q)$ and the constants $\alpha_{ij}, \beta_{ij}, \gamma_j, \delta_i$ are fixed. Let us view c_j, $j \in J$ as input variables, c as the output and ξ_j, $j \in J$ as internal variables of a linear program that we describe below. Given the parameters, we want to find a solution that gives the maximum c that satisfies the equality. More precisely, for a given input assignment to c_j, $j \in J$, we want to maximize

$$\sum_j \gamma_j c_j + \sum_{ij} \beta_{ij} c_j + \sum_j \xi_j c_j.$$

under the constraints that the homogeneous part of the expression gives the form $F(q)$. Since β_{ij}, γ_j and c_j are given, this is the same task as to maximize $\sum_j \xi_j c_j$.

This gives us a linear program P_1 where input constants c_j appear in the objective function as well as in the constraints. Specifically, the constraints for ξ_{ij} are

$$f_i = \sum_j \alpha_{ij} (f_{ij} - c_j) + \sum_i \beta_{ij} (-f_{ij} + c_j) + \sum_j \gamma_j f_{ij} + \sum_j \xi_j f_{ij}$$

for $i = 1, \ldots, n$, where f_i and f_{ij} are the coefficients of the forms F and F_j respectively.

Note that the coefficient at c_j is $\sum_i -\alpha_{ij} + \beta_{ij}$, which can have any sign. Thus these constraints are *not monotone* in the input c_j, $j \in J$. Therefore, we relax the problem and allow to use weaker inequalities: instead of the constant c_j in $F_j(q) - c_j \geq 0$, we will allow any constant less than or equal to c_j. This is allowed, because of the weakening rule. So the first part of the expression will get the form

$$\sum_{ij} \alpha_{ij} q_i (F_j(q) - \eta_{ij}) + \sum_{ij} \beta_{ij} (1 - q_i)(F_j(q) - \zeta_{ij}).$$

Now the new constraints are

$$\eta_{ij} \leq c_j, \quad \zeta_{ij} \leq c_j, \quad \text{for } i = 1, \ldots, n,\ j \in J,$$

$$f_i = \sum_j \alpha_{ij}(f_{ij} - \eta_{ij}) + \sum_j \beta_{ij}(-f_{ij} + \zeta_{ij}) + \sum_j \gamma_j f_{ij} + \sum_j \xi_j f_{ij},$$

for $i = 1, \ldots, n,\ j \in J$, and we want to maximize

$$\sum_{ij} \beta_{ij}\zeta_{ij} + \sum_j \xi_j c_j.$$

Thus we have obtained a linear program P_2 such that in its constraints the input variables c_j occur only positively, i.e., P_2 is a strong MLP. Clearly, the maximum of $\sum_{ij} \beta_{ij}\zeta_{ij} + \sum_j \xi_j c_j$ in P_2 is at least as large as in P_1, because we can substitute $\eta_{ij} := c_j$, $\zeta_{ij} := c_j$ in P_2.

Theorem 6.1 *Let $\Phi(p,q) \cup \Gamma(p,r)$ be a set of unsatisfiable inequalities such that the variables p occur in Φ only with negative coefficients. Let an LS refutation of $\Phi(p,q) \cup \Gamma(p,r)$ with m lap-steps be given. Then one can construct a circuit with m strong MLP gates that represents a Boolean function f such that for every $a \in \{0,1\}^n$,*

1. *if $f(a) = 0$, then $\Phi(a,q)$ is unsatisfiable, and*
2. *if $f(a) = 1$, then $\Gamma(a,r)$ is unsatisfiable.*

The size of the strong gates is polynomially bounded by the size of the proof.

Proof. We start by recalling the idea of feasible interpolation for LS in the non-monotone case as presented in [20].

Let

$$E_1(p) + F_1(q) + G_1(r) \geq e_1, \ \ldots\ , E_m(p) + F_m(q) + G_m(r) \geq e_m$$

be the linear inequalities of an *LS* refutation of $\Phi(p,q) \cup \Gamma(p,r)$. Since the last inequality is a contradiction, the linear forms E_m, F_m, G_m are zeros and $e_m > 0$. Let $a \in \{0,1\}^n$ be an assignment to variables p. Substituting a into the proof we get a refutation

$$F_1(q) + G_1(r) \geq e_1 - E_1(a), \ \ldots\ , F_m(q) + G_m(r) \geq e_m - E_m(a)$$

of $\Phi(a,q) \cup \Gamma(a,r)$ (note that the last inequality is $0 \geq e_m$ as in the proof above). Our aim now is to split the restricted proof into two proofs

$$F_1(q) \geq c_1, \ \ldots\ , F_m(q) \geq c_m \quad \text{and} \quad G_1(r) \geq d_1, \ \ldots\ , G_m(r) \geq d_m$$

in such a way that the linear forms of the first sequence are the linear forms of a proof from $\Phi(a,q)$, the linear forms of the second are the linear forms of a proof from $\Gamma(a,r)$ and

$$c_j + d_j \geq e_j - E_j(a) \quad \text{for } j = 1, \ldots, m.$$

Since $e_m > 0$, we have that either $c_m > 0$, or $d_m > 0$ or both are true. Hence at least one of the proofs is a refutation of its initial inequalities. We will show how e_m can be computed and this will give us the circuit for interpolation.

We now describe how such a splitting can be constructed.

First, suppose $E_j(p) + F_j(q) + G_j(r) \geq e_j$ is an element of Φ. Then $G_j(r) \equiv 0$. This inequality will be split into

$$F_j(q) \geq e_j - E_j(a) \quad \text{and} \quad 0 \geq 0.$$

Since all coefficients in E_j are negative, $e_j - E_j(a)$ can be computed from a using an MLP gate. If $E_j(p) + F_j(q) + G_j(r) \geq e_j$ is an element of Γ, we split the inequality into

$$0 \geq 0 \quad \text{and} \quad G_j(r) \geq e_j - E_j(a).$$

Now suppose that $E_t(p) + F_t(q) + G_t(r) \geq e_t$ follows from previous inequalities and suppose we have already split the previous part of the proof. Substituting a into the jth *lap*-step we obtain an equality of the following form.

$$F_t(q) + G_t(r) + E_t(a) - e_t =$$

$$\sum_{ij} \alpha_{ij} a_i (F_j(q) + G_j(r) + E_j(a) - e_j) + \sum_{ij} \beta_{ij}(1 - a_i)(F_j(q) + G_j(r) + E_j(a) - e_j) +$$

$$\sum_{ij} \alpha'_{ij} q_i (F_j(q) + G_j(r) + E_j(a) - e_j) + \sum_{ij} \beta'_{ij}(1 - q_i)(F_j(q) + G_j(r) + E_j(a) - e_j) +$$

$$\sum_{ij} \alpha''_{ij} r_i (F_j(q) + G_j(r) + E_j(a) - e_j) + \sum_{ij} \beta''_{ij}(1 - r_i)(F_j(q) + G_j(r) + E_j(a) - e_j) +$$

$$\sum_{i} \gamma'_i (q_i - q_i^2) + \sum_{i} \gamma''_i (r_i - r_i^2) +$$

$$\sum_{j} \delta_j (F_j(q) + G_j(r) + E_j(a) - e_j). \tag{23}$$

In the sums we have $j < t$ and the indices range over the sets of indices of the corresponding variables p, q, r. All these linear combinations are nonnegative, i.e., $\alpha_{ij}, \alpha'_{ij}, \alpha''_{ij}$, $\beta_{ij}, \beta'_{ij}, \beta''_{ij}$, γ'_i, γ''_i, $\delta_j \geq 0$.

Now we substitute $-c_j - d_j$ for $E_j(a) - e_j$. Then everything splits naturally into an LS proof in variables q and an LS proof in variables r, except for the terms of the following form.

$$\sum_{ij} \alpha'_{ij} q_i (G_j(r) - d_j) + \sum_{ij} \beta'_{ij}(1 - q_i)(G_j(r) - d_j) +$$

$$\sum_{ij} \alpha''_{ij} r_i (F_j(q) - c_j) + \sum_{ij} \beta''_{ij}(1 - r_i)(F_j(q) - c_j).$$

Let $P(q, r)$ denote the above polynomial. The key observation is that in this expression all quadratic terms $q_i r_{i'}$ must cancel, because they do not occur elsewhere. Furthermore, the

inequality $P(q,r) \geq 0$ is a consequence of the inequalities $F_j(q) - c_j \geq 0$ and $G_j(r) - d_j \geq 0$ in the domain of real numbers. Hence

$$P(q,r) = \sum_{j<t} \xi_j (F_j(q) - c_j) + \sum_{j<t} \xi'_j (G_j(r) - d_j),$$

for some $\xi_j, \xi'_j \geq 0$. (For the sake of simplicity, we assume that the inequalities $0 \leq q_i \leq 1$ and $0 \leq r_i \leq 1$ are included in Φ and Γ.) Thus we can split also this part of the proof into the q part and r part. In particular, the constant $e_t - E_t(a)$ splits into some constants c'_t and d'_t:

$$e_t - E_t(a) = c'_t + d'_t. \tag{24}$$

The q part of the proof gives us

$$\begin{aligned} F_t(q) - c'_t = \; & \textstyle\sum_{ij} \alpha_{ij} a_i (F_j(q) - c_j) + \sum_{ij} \beta_{ij} (1 - a_i)(F_j(q) - c_j) + \\ & \textstyle\sum_{ij} \alpha'_{ij} q_i (F_j(q) - c_j) + \sum_{ij} \beta'_{ij} (1 - q_i)(F_j(q) - c_j) + \\ & \textstyle\sum_{ij} \beta''_{ij} (F_j(q) - c_j) + \\ & \textstyle\sum_i \gamma' (q_i - q_i^2) + \\ & \textstyle\sum_j \delta_j (F_j(q) - c_j) + \\ & \textstyle\sum_j \xi_j (F_j(q) - c_j). \end{aligned} \tag{25}$$

To construct this proof one needs to find values of c'_t and ξ_j so that the equations (24) and (25) are satisfied. Instead of computing c'_t that satisfies (24), we will maximize the constant term in (25), because it suffices to have c_t and d_t such that

$$e_t - E_t(a) \leq c_t + d_t.$$

This means that the constraints are given by the coefficients at variables q_i (one equation for very i) and we maximize $\sum_j \xi_j c_j$.

However, we need a strong *monotone* linear program. We cannot use these constraints, because c_t need not be monotone in c_j, $j < t$. So we relax the proof and use inequalities possibly weaker than $F_j(q) \geq c_j$ and we do get a strong MLP as described before the theorem. ■

6.4 Feasible interpolation for a restricted class of proofs

There are a lot of dependencies among the variables that appear in formula (25) that we have not made use of, so it is not excluded that in fact the dual MLP gates suffice. This is certainly true at least in some special cases. In this section we will show that a particular type of restricted type of LS proofs can be interpolate by dual MLPs. We will consider the following conditions.

(a) The initial inequalities have the form $\Phi(p,q) \cup \Gamma(p,r)$ the common variables p occur in Φ only with negative coefficients;

(b) the multiplication rule and the integrality axiom is only used for variables r.

We will give an example of such a proof in the next subsection.

Theorem 6.2 *Given an LS proof* Π *of* $\Phi(p,q) \cup \Gamma(p,r)$ *satisfying the conditions (a) and (b), one can construct a dual MLP that represents a Boolean function* f *such that for every* $a \in \{0,1\}^n$,

1. *if* $f(a) = 0$, *then* $\Phi(a,q)$ *is unsatisfiable, and*

2. *if* $f(a) = 1$, *then* $\Gamma(a,r)$ *is unsatisfiable.*

The size of the MLP is polynomial in the size of the proof Π.

Proof. To prove this theorem we only need to change the last part of the proof of Theorem 6.1 that starts with formula (25). Due to the condition above, lines 1,2 and 4 will not be present in the formula. Thus the formula simplifies to

$$F_t(q) - c'_t = \sum_{ij} \beta''_{ij}(F_j(q) - c_j) + \sum_j \delta_j(F_j(q) - c_j) + \sum_j \xi_j(F_j(q) - c_j).$$

Hence the constraints for ξ are given simply by

$$F_t(q) = \sum_{ij} \beta''_{ij} F_j(q) + \sum_j \delta_j F_j(q) + \sum_j \xi_j F_j(q).$$

These constraints do not contain any input variables c_j. Hence to split the *lap*-step we only need to maximize $\sum_j \xi_j c_j$ under the constraints. This is neither a minimization, nor a maximization, but it is a dual concept (dual in the sense of the duality of monotone Boolean functions). Using linear programming duality, we can replace such gates by gates that have inputs in the constraints and where one minimizes a fixed objective function. One can check that it is possible to combine such gates into a single one, as we did it for the primal concept in Proposition 4.2. Thus we get a dual MLP that interpolates $\Phi(p,q) \cup \Gamma(p,r)$. ∎

6.5 Cutting Planes vs. Lovász-Schrijver proofs

The cutting plane proof system is defined by:

- Axiom:

$$0 \leq p_j \leq 1.$$

- Rules:

1. *positive linear combinations;*
2. *rounding rule:* Suppose that all c_i are integers. Then

$$\text{from } \textstyle\sum_i c_i p_i \geq d, \text{ derive } \sum_i c_i p_i \geq \lceil d \rceil.$$

In [8] Fu defined a certain family of tautologies and showed that their negations, represented by linear inequalities, cannot be refuted by cutting plane proofs of polynomial size. He used the cutting-plane interpolation theorem of [20] and an extension of Razborov's lower bound $n^{\Omega(\log n)}$ to monotone real circuits. Below we define a set of unsatisfiable (in $\mathbb{N}$) inequalities Φ_n for every n that are weaker than Fu's inequalities,[5] hence they also require $n^{\Omega(\log n)}$ cutting-plane refutations. (This can also be proved directly using the cutting-plane interpolation theorem and Fu's lower bound on monotone real circuits.) But we also show that Φ_n have polynomial size LS proofs.

Definition 7 *Φ_n is the set of inequalities defined as follows.*

Variables: x_{ij} (matching X), w_{ij} (edges of a graph W), v_i, u_i (subsets U, V), $i, j \in [n]$, $i \neq j$. The pairs of indices ij are understood to be unordered *pairs.*

Inequalities:

1. $\sum_{i,i\neq j} x_{ij} = 1$ *(for every j two inequalities, meaning: X defines a perfect matching)*

2. $x_{ij} \leq w_{ij}$ *(matching X is a subgraph of W)*

3. $v_j + w_{ij} \leq u_i + 1$ *(if $j \in V$ and $(i,j) \in W$, then $i \in U$)*

4. $(\sum_i u_i) + 1 \leq \sum_j v_j$ *($|U| < |V|$)*

The interpretation suggested in the parentheses says that $([n], W)$ is a graph that has a perfect matching and there are two subsets of vertices U and V such that all edges that have one vertex in V have the other vertex in U, and $|U| < |V|$, which is, clearly, impossible.

Theorem 6.3 *The tautology in Definition 7 has an LS proof of polynomial size. Furthermore, the proof satisfies conditions (a) and (b) above.*

Proof. The following is a polynomial size LS refutation of 1.–4.

5. $v_j + x_{ij} \leq u_i + 1$	from 2. and 3.
6. $x_{ij}v_j + x_{ij}^2 \leq x_{ij}u_i + x_{ij}$	multiplying 5. by x_{ij}
7. $x_{ij}v_j \leq x_{ij}u_i$	subtracting $x_{ij}^2 = x_{ij}$ from 6.
8. $\sum_{ij} x_{ij}v_j \leq \sum_{ij} x_{ij}u_i$	sum of 7. over all $i \neq j$
9. $\sum_j v_j \sum_{i;i\neq j} x_{ij} \leq \sum_i u_i \sum_{j;j\neq i} x_{ij}$	rewriting 8.
10. $\sum_j v_j \leq \sum_i u_i$	from 9. and 1.

[5] We will give a polynomial size derivation of Φ_n in the full version of this paper.

11. $1 \leq 0$ from 4. and 10.

To satisfy conditions (a) and (b), let w be the common variables p, let u, v be variables q and let x be variables r. ■

Corollary 6.4 *The cutting-plane proof system does not polynomially simulate the Lovász-Schrijver proof system.*

This is the first nontrivial result concerning the longstanding open problem to determine the mutual relation of the two proof systems.

Corollary 6.5 *Let F_n be the partial Boolean function with $\binom{n}{2}$ variables that is 1 on inputs that encode graphs with perfect matchings and 0 on inputs that encode complete bipartite graphs $K_{s,t}$, $s \neq t$, $s + t = n$. Then there exists a dual MLP of polynomial size that represents F.*

Proof. Note that graphs with matchings satisfy inequalities 1. and 2. of Φ_n, and bipartite graphs $K_{s,t}$, $s \neq t$, $s + t = n$ satisfy 3. and 4. (There are surely more graphs that satisfy 3. and 4., but for the sake of simplicity we only consider complete bipartite ones.) Hence, by Theorems 6.2 and 6.3 there exists a dual MLP with the required properties. ■

7 Monotone linear programs and extended formulations

If a polytope $P \subseteq \mathbb{R}^n$ is given by a polynomial number of inequalities with polynomial size coefficients, we can easily decide whether a vector $v \in \mathbb{R}^n$ belongs to P. An important observation is that even if P requires an exponential number of inequalities to be defined, we may still be able to test whether $v \in P$ efficiently if we can find a polytope $R \subseteq \mathbb{R}^{n+m}$ in a higher dimension such that P is a projection of R and R can be described by a polynomial number of inequalities with polynomially small coefficients. This is because the latter problem is, in fact, a linear programming problem. Formally stated, let A be a matrix that defines R, i.e.,

$$(v, y) \in R \;\Leftrightarrow\; A(v, y) \leq b.^6$$

Then

$$v \in P \;\Leftrightarrow\; \exists y\; A(v, y) \leq b.$$

We say that $A(v, y) \leq b$ is an *extended formulation* of P. Nevertheless, for many important polytopes, e.g. the cut polytope, tsp polytope, etc. (add other polytopes) , it has been shown that also every extended formulation requires an exponential number of inequalities [7, 24].

[6] (v, y) denotes the vector $(v_1, \ldots, v_n, y_1 \ldots, y_m)$.

Defining a partial Boolean function by a linear program is a closely related task, but not an equivalent one. For a partial Boolean function F, let $Ones(F)$, and $Zeros(F)$ denote the set of all inputs $a \in \{0,1\}^n$ such that $F(a) = 1$, and $F(a) = 0$ respectively. Let P_F denote the convex hull of $Ones(F)$. Defining F by a linear program is the same as finding an extended formulation of some polyhedron Q that contains $Ones(F)$ and is disjoint from $Zeros(F)$. Note that $Q \supseteq Ones(F)$ is equivalent to $Q \supseteq P_F$. Finding such a Q that is defined by a small number of inequalities is, clearly, a simpler task than finding a small extended formulation of P_F. Eg. the (nonbipartite) matching function is computable by a polynomial size Boolean circuit, and hence it can be defined by a small (not necessarily monotone) linear program. Nevertheless, the corresponding polytope P_F does not have a sub-exponential extended formulation [24].

Let us now turn to *monotone* linear programs. In this case we *do not* have an example of a *total* function F that has polynomial size MLP, and $Ones(F)$ does not have polynomial size extended formulation. But we do have an example in which a dual MLP of polynomial size can separate $Ones(F)$ from a "natural" subset of maxterms. This is Corollary 6.5. This suggests that it may be difficult to use the techniques of extended formulation lower bounds to prove lower bounds on MLPs.

As we noted above, if we want to prove a lower bound MLPs, we have to use the fact that they are monotone. What does monotonicity mean in geometric term? Let $\mathbb{R}^n_+$ denote the *nonnegative cone* $\{v \in \mathbb{R}^n \mid v \geq 0\}$. To represent a monotone Boolean function F by an MLP means that we want to find a polyhedron Q such that

$$P_F + \mathbb{R}^n_+ \subseteq Q, \quad \text{and} \quad Q \cap Zeros(F) = \emptyset. \tag{26}$$

Notice that if P_F is nonempty, then $Q = Q + \mathbb{R}^n_+$. Furthermore, we can replace P_F in $P_F + \mathbb{R}^n_+$ by the convex hull of minterms of F and $Zeros(F)$ by maxterms of F.

Let P^*_F denote the convex hull of the minterms of a monotone Boolean function F. If P^*_F lays on a hyperplane, we may reduce the task of separating $P_F + \mathbb{R}^n$ from $Zeros(F)$ to the task of separating P^*_F from some other polytopes. Let H be a hyperplane such that $P^*_F \subseteq H$. We project the zeros of F to H by applying the following map for each v such that $F(v) = 0$:

$$v \;\mapsto\; S_v := H \cap \{u \mid u \leq v\}. \tag{27}$$

If the weights of maxterms are bigger than the weights of minterms, then each S_v is a simplex. The task is now to separate P^*_F from $\bigcup_v S_v$ where the union is over the maxterms of F.

Proposition 7.1 *The minimum size of an extended formulation of a polytope Q such that*

$$P^*_F \subseteq Q, \quad \textit{and} \quad Q \cap \bigcup_v S_v = \emptyset \tag{28}$$

is, up to a constant factor, equal to the minimum size MLP computing F.

Proof. It is clear from the discussion above that an MLP for F defines a polyhedron Q that separates the sets $P_F + \mathbb{R}^+$ and $Zeros(F)$ according to Equation (26). Then Q (or $Q \cap H$, if you prefer) separates the sets in Equation (28).

To prove the other inequality, let us assume that we have an extended formulation $\exists y\ A(x,y) \leq b$ of some Q satisfying Equation (28). Clearly we can assume, moreover, that $Q \subseteq H$. Then an MLP for F is given by

$$x \leq p,\ A(x,y) \leq b. \tag{29}$$

■

All monotone Boolean functions for which lower bounds have been proved have the property that maxterms have essentially bigger weight than minterms. Thus the simplices S_v are relatively large.

Example. Let F be the partial monotone Boolean function where minterms are k-cliques in a graph on n vertices and maxterms are complete $(k-1)$-partite graphs. Suppose $k = n^\alpha$ for some $0 < \alpha < 1$. Then H is the hyperplane of all vectors of weight $\binom{k}{2}$. The weight of maxterms is $\approx kn = n^{1+\alpha}$ while the weight of minterms is $\approx n^{2\alpha}$.

A possible approach to lower bounds could be to show that any polytope Q that separates P_F from $\bigcup_v S_v$ must be close to P_F and then use the techniques that show that an extended formulation of polytopes close to P_F is always exponential.

8 Conclusion

We have presented a computational model for computing monotone Boolean functions. We believe that for this model, it is possible to prove explicit lower bounds, like it was done for monotone Boolean circuits, although it may be not easy. Nevertheless, there is always the danger that a proposed model for computing monotone functions is too strong, so strong that it can simulate general Boolean circuits computing monotone Boolean functions. Then such a model would be uninteresting, because it would not be be possible to prove lower bounds for explicit functions without making a breakthrough in computational complexity.

Here is an example of a model that is too strong. A *nondeterministic monotone circuit* for a Boolean function $F(p)$ is a monotone circuit $C(p,q,r)$, where q and r are strings of variables of equal length such that

$$F(p) = 1 \ \Leftrightarrow\ \exists q\ C(p,q,\neg q) = 1.$$

Note that this is a fully syntactic definition—the form of the circuit ensures that the function it computes is monotone. Yet this kind of circuits are equivalent to general nondeterministic circuits.

Proposition 8.1 *If a monotone function F is computed by a nondeterministic circuit of size S, then there exists a monotone nondeterministic circuits of size $O(S)$ that computes F*

Proof. Suppose

$$F(p) = 1 \;\Leftrightarrow\; \exists q\; C(p, \neg p, q, \neg q) = 1,$$

where C is monotone. Then we can represent F as follows

$$F(p) = 1 \;\Leftrightarrow\; \exists q, r\; C(r, \neg r, q, \neg q) \wedge \bigwedge_i (p_i \vee \neg r_i) = 1.$$

■

If the computational model of monotone linear programs is not too strong, then the main problem is to prove lower bounds for explicit partial Boolean functions.

Here are some more problems.

1. Given a representation of F by a monotone LP, can one construct a representation of the dual function by a monotone LP of size at most polynomially larger? Apparently, this is also an open problem for span programs.

2. Can one bound the coefficients in MLPs? Specifically, given an MLP of polynomial size, can one modify it so that the coefficients in the matrix and the vectors are of polynomial size? This is also open for monotone span programs over reals.

3. Can one prove an exponential lower bound in a nonconstructive way? If the answer to the previous problem is positive, then this follows by simple counting.

4. Is there a *total* function F that has polynomial size MLP, or dual MLP, but $Ones(F)$ does not have polynomial size extended formulation?

References

[1] M. Alekhnovich and A. A. Razborov. Satisfiability, branch-width and tseitin tautologies. In *Proc. of the 43rd Symposium on Foundations of Computer Science*, pages 593–603, 2002.

[2] L. Babai, A. Gál, and A. Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19(3):301–319, 1999.

[3] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlák. Lower bounds on hilbert's nullstellensatz and propositional proofs. *Proceedings of the London Mathematical Society*, 3(1):1–26, 1996.

[4] S. R. Buss and T. Pitassi. Good degree bounds on nullstellensatz refutations of the induction principle. *Journal of computer and system sciences*, 57(2):162–171, 1998.

[5] S. A. Cook, T. Pitassi, R. Robere, and B. Rossman. Exponential lower bounds for monotone span programs. *ECCC*, TR16-64.

[6] S. Dash. Exponential lower bounds on the lengths of some classes of branch-and-cut proofs. *Mathematics of Operations Research*, 30(3):678–700, 2005.

[7] S. Fiorini, S. Massar, S. Pokutta, H. R. Tiwary, and R. D. Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *Journal of the ACM (JACM)*, 62(2):17, 2015.

[8] X. Fu. Lower bounds on sizes of cutting planes proofs for modular coloring principles. *Proof Complexity and Feasible Arithmetics*, pages 135–148, 1998.

[9] A. Gál and P. Pudlák. A note on monotone complexity and the rank of matrices. *Information Processing Letters*, 87(6):321–326, 2003.

[10] D. Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1):613–622, 2001.

[11] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[12] A. Haken and S. A. Cook. An exponential lower bound for the size of monotone real circuits. *Journal of Computer and System Sciences*, 58(2):326–335, 1999.

[13] R. Impagliazzo, P. Pudlák, and J. Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999.

[14] M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference (San Diego, CA, 1993)*, pages 102–111. IEEE Comput. Soc. Press, Los Alamitos, CA, 1993.

[15] J. Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(02):457–486, 1997.

[16] J. Krajíček. Interpolation and approximate semantic derivations. *Mathematical Logic Quarterly*, 48(4):602–606, 2002.

[17] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[18] T. Pitassi and N. Segerlind. Exponential lower bounds and integrality gaps for tree-like lovasz-schrijver procedures. *SIAM Journal on Computing*, 41(1):128–159, 2012.

[19] P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(03):981–998, 1997.

[20] P. Pudlák. On the complexity of the propositional calculus. *London Mathematical Society Lecture Note Series*, pages 197–218, 1999.

[21] P. Pudlak and J. Sgall. Algebraic models of computation and interpolation for algebraic proof systems. In *Proc. of Feasible Arithmetic and Proof Complexity, DIMACS Series in Discrete Math. and Theoretical Comp. Sci.*, volume 39, pages 279–295, 1998.

[22] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM (JACM)*, 39(3):736–744, 1992.

[23] A. A. Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical Notes*, 37(6):485–493, 1985.

[24] T. Rothvoß. The matching polytope has exponential extension complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 263–272. ACM, 2014.

[25] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 2000.

A Proof of Lemma 3.1

MLP-1 → MLP-2. Assume that (A, B, b) is an MLP-1 representation of F. Let x' be an additional variable. We claim that F has the following MLP-2 representation.

$$F(p) = \begin{cases} 1 \;\Rightarrow\; \max\{-x' \mid Ax \le x' + b + Bp,\ x \ge 0, x' \ge 0\} \ge 0, \\ \\ 0 \;\Rightarrow\; \max\{-x' \mid Ax \le x' + b + Bp,\ x \ge 0, x' \ge 0\} < 0. \end{cases}$$

To see this, note that if $F(p) = 1$, then there exists x such that $Ax \le b + Bp$. Therefore, the maximum is attained by setting $x' = 0$. On the other hand, if $F(p) = 0$ then for every $x \ge 0$, we have that $Ax > b + Bp$. Hence $x' > 0$. This implies that the maximum is negative. According to the definition of MLP-2, $Ax \le x' + b$ should define a polytope. It is clear that it defines a nonempty polyhedron, because we can take a sufficiently large x' to satisfy all inequalities. To make it bounded it suffices to bound the variables x, x' by a sufficiently large constant that will not change the definition of F.

MLP-2 → MLP-1. Assume that F is MLP-2 representable. Consider the system $Ax \le b + Bp$ augmented by the equation $c \cdot x \ge 0$. If $F(p) = 1$, then the maximum of $c \cdot x$ among all x satisfying $Ax \ge b + Bp$ is greater than 0. Therefore, there exists some $x \ge 0$ for which $c \cdot x \ge 0$. If $F(p) = 0$, then the maximum value of $c \cdot x$ is less than 0 and the new system has no solution.

MLP-2 → MLP-3. Assume that (A, B, b, c) is an MLP-2 representation of F. Consider an additional variable x'. Then we have that for each p,

$$F(p) = \begin{cases} 1 \to\ \max\{c \cdot x - x' \mid Ax \le b + Bp,\ x \ge 0,\ 0 \le x' \le K,\ c \cdot x - x' \le 0\} = 0, \\ \\ 0 \to\ \max\{c \cdot x - x' \mid Ax \le b + Bp,\ x \ge 0,\ 0 \le x' \le K,\ c \cdot x - x' \le 0\} < 0, \end{cases}$$

where K is a sufficiently large constant. The inequality $x' \leq K$ is only added to formally satisfy our definition that requires that the set of solutions is always bounded.

MLP-3 $\rightarrow$ MLP-2. Immediate.

MLP-4 $\leftrightarrow$ MLP-2: Follows straightforwardly by linear programming duality (2).

Restricting B. Now we claim that B can be assumed to be a 0/1 matrix with at most one 1 in each row. Assume that (A, B, b) is an MLP-1 representation of F, for some $i \in \{1, 2, 3\}$. Take n new variables $y = (y_1, ..., y_n)$, and replace the system of inequalities $Ax \leq b + Bp$ by the system

$$0 \leq y \leq 1,\ y \leq p,\ Ax - By \leq b.$$

The fact that the same claim holds for MLP-4 representations of F follows from the equivalence between MLP-2 representations and MLP-4 representations. This concludes the proof of Lemma 3.1. ■