Interdisciplinary Information Sciences Vol. 21, No. 4 (2015) 329–349
©Graduate School of Information Sciences, Tohoku University
ISSN 1340-9050 print/1347-6157 online
DOI 10.4036/iis.2015.L.03

Exploring the Limits of Subadditive Approaches: Parallels between Optimization and Complexity Theory

Kenya UENO*

*The Hakubi Center for Advanced Research and Graduate School of Informatics,
Kyoto University, Kyoto 606-8501, Japan*

In this paper, we review subadditive approaches which arise in the theory of mathematical programming and computational complexity. In particular, we explain the duality theorem of integer programming and techniques to prove formula-size lower bounds as fundamental subjects in mathematical programming and computational complexity, respectively. We discuss parallel visions of these two different areas by showing some connections between them.

KEYWORDS: formula complexity, integer programming, linear programming

1. Introduction

In this paper, we overview existing studies concerned with subadditivity in two distinct areas, optimization and complexity theory, and give parallels between optimization and complexity theory. The word "parallel" has (at least) two different meanings: "two lines do not intersect" and "analogy." Like the meanings of the word "parallel," the two areas (i.e., optimization and complexity theory) are independent and never to be unified, but have many similar notions between them. Therefore, some tools in one area may be useful to enhance the research in the other area. The aim of this paper is to give an introductory tutorial of interdisciplinary researches by moving back and forth between the two areas. Most of sections are divided into two parts: the one part is on optimization theory and the other part on complexity theory.

1.1 Mathematical Programming and Notion of Subadditivity

Linear programming (LP) and integer programming (IP) have a wide range of applications, and optimization techniques for linear and integer programming problems have been developed extensively by various researchers. From complexity-theoretic viewpoint, linear programming problem belongs to the complexity class **P**, which is the sets of problems solvable by polynomial time computation. On the other hard, integer programming problem is one of the representative **NP**-hard problems, which means that, if integer programming problem is proved to be in **P**, then we can conclude $\mathbf{P} = \mathbf{NP}$.

The duality theorem for linear programming problem is well known and can be found in almost all textbooks of mathematical programming as a basic result [32, 37]. Similarly, the duality theorem for integer programming problem has been also studied for a long period of time [8, 15, 26, 31, 32]. According to the duality theory in linear programming and integer programming, dual problems of linear programming problems have additive properties, while dual problems of integer programming problems have subadditive properties. In this sense, the gap between tractability and intractability in computation of linear and integer programming problems would be roughly explained by the difference between additivity and subadditivity.

The notion of subadditivity appears at many scenarios in the theory of computation. It is informally stated by the following sentences. Let S be a set which is closed under some binary operation like $+$ as follows.

$$x, y \in S \Rightarrow x + y \in S$$

Elements in S can be scalars, vectors or matrices depending on the situation. If a function $\mu : S \to \Re$ satisfies

$$\mu(x) + \mu(y) \geq \mu(x + y),$$

for any $x, y \in S$, then we call μ a *subadditive* function for S. If the inequality always holds with equality, i.e.,

Received September 10, 2013; Accepted September 12, 2014; J-STAGE Advance published October 20, 2015
2010 Mathematics Subject Classification: 68Q17, 90C05, 90C10
This work is supported by Grant-in-Aid for Young Scientists (B) (JSPS KAKENHI Grant Number 24700010), Grant-in-Aid for Scientific Research on Innovative Areas (MEXT KAKENHI Grant Number 24106006), and the Hakubi Project of Kyoto University.

*Corresponding author. E-mail: kenya@i.kyoto-u.ac.jp

$$\mu(x) + \mu(y) = \mu(x + y),$$

then we call μ an *additive* function for S. A simple example of subadditive functions is the square root function as we have

$$\sqrt{x} + \sqrt{y} \geq \sqrt{x+y}$$

for all $x, y \geq 0$.

In the contexts of mathematics, there is a notion of Caratheodory's outer measure, which is abstraction of the size of area and essentially subadditive. Its intuition of subadditivity is illustrated in Figure 1. There may be a lot of unexplored tools in measure theory which might be useful to show existence of certain subadditive functions.

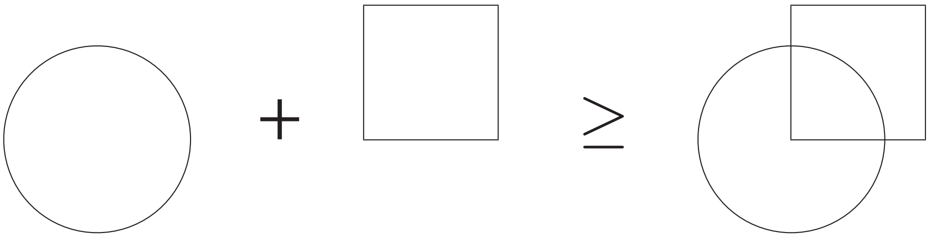


Fig. 1. Subadditivity of the size of area (Caratheodory's outer measure).

We have explained that in the theory of mathematical programming, additive property is easy to handle, while subadditive property is hard to handle. This is also the case for a fundamental problem in computational complexity as explained in the next subsection.

1.2 Circuit Complexity, Formula Complexity, and Limits of Computation

A circuit takes only a fixed-length input while a Turing machine can take an input whose length is not determined in advance. This is why circuits are called non-uniform computation models while Turing machines are called uniform computation models. Polynomial-time computation by a Turing machine can be simulated by polynomial-size circuits, where each circuit is constructed for each fixed length of inputs (see, e.g., [3]). Circuits are much stronger than Turing machines in the sense that we have a possibility to use more tricks for constructing smaller circuits due to the information of the input length determined in advance. This partially explains why it is hard to show circuit-size lower bounds in general. Nevertheless, circuit complexity is intensively studied to measure complexity of problems because we can analyze their concrete bounds by combinatorial approach. Proving a super-polynomial circuit-size lower bound means a separation between the most important complexity class **P** and other complexity classes beyond **P**. In particular, a super-polynomial circuit-size lower bound for a function in **NP** implies **P** $\neq$ **NP**. Despite many efforts, the current best circuit-size lower bound for an explicit function in **NP** is $5n - o(n)$ [13, 23].

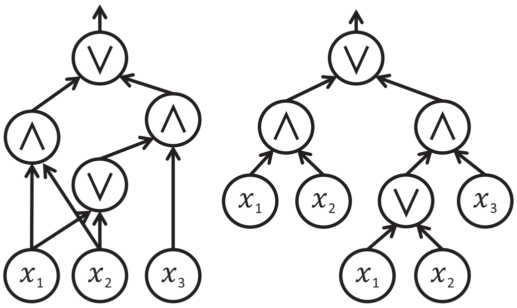


Fig. 2. Difference between circuit (the DAG on the left side) and formula (the tree on the right side).

Since it is extremely hard to obtain a strong circuit-size lower bound which yields a separation of complexity classes and even to give a slight improvement, it is a better way to aim at solving a weaker problem. Formula is a restriction of the circuit model where out-degree of each node in a formula is bounded by one, as shown in Figure 2. Formula is a fundamental computational model in its own right as such a structure appears in many fields of computer science, e.g., SAT. Deriving a formula-size lower bound for an explicit Boolean function is a fundamental problem in computational complexity theory as a weaker version of the circuit-size lower bound problem. The current best formula-size lower bound of $n^{3-o(1)}$ by Håstad [9] is larger than the best circuit-size lower bound.

Spira [43] proved that the circuit-depth lower bound problem is essentially equivalent to the formula-size lower bound problem in the sense that a Boolean functions is expressible by a polynomial size formula if and only if it is also expressible by a logarithmic depth circuit. Therefore, a super-polynomial formula-size lower bound for an explicit Boolean function f implies a super-logarithmic circuit-depth lower bound. This consequence yields a separation between $\mathbf{NC}^1$ and a complexity class including such f. For example, a super-polynomial formula-size lower bound for a function in **NP** implies $\mathbf{NC}^1 \neq \mathbf{NP}$. The complexity class $\mathbf{NC}^1$ (abbreviation of Nick's Class) named after Nick Pippenger is defined as the set of problems which are computable in logarithmic depth circuits. Depth complexity of circuits is quite important from several viewpoints. For example, a critical path in a hardware system essentially affects the performance of CPUs. Moreover, parallel time complexity is modeled as depth complexity of circuits in many studies. Thus, we can insist that proving circuit-depth lower bound is significant to clarify the limits of constructing efficient parallel algorithms. In addition to the depth complexity of Boolean circuits, the complexity class $\mathbf{NC}^1$ has been characterized by several ways like bounded-width polynomial-size branching programs [4].

Since proving a formula-size lower bound is one of the most fundamental problems in computation complexity theory, there are a lot of methods to derive improved lower bounds. One of the most classical methods to prove formula-size lower bounds is the one invented by Khrapchenko [20] who proved an n^2 formula-size lower bound for the parity function. As generalizations of Khrapchenko's method, there are a lot of techniques studied to improve formula-size lower bounds. However, improvements are very rare. Karchmer, Kushilevitz and Nisan [18] formulated the formula-size problem as an integer programming problem called the rectangle bound and introduced a technique called the LP bound, which gives a lower bound by using a feasible solution of the dual problem of its LP relaxation. In our recent studies, we have devised stronger versions of the LP bound [44, 45]. Both of them can be interpreted as exploration into subadditivity beyond the original LP bound [18], which is additive.

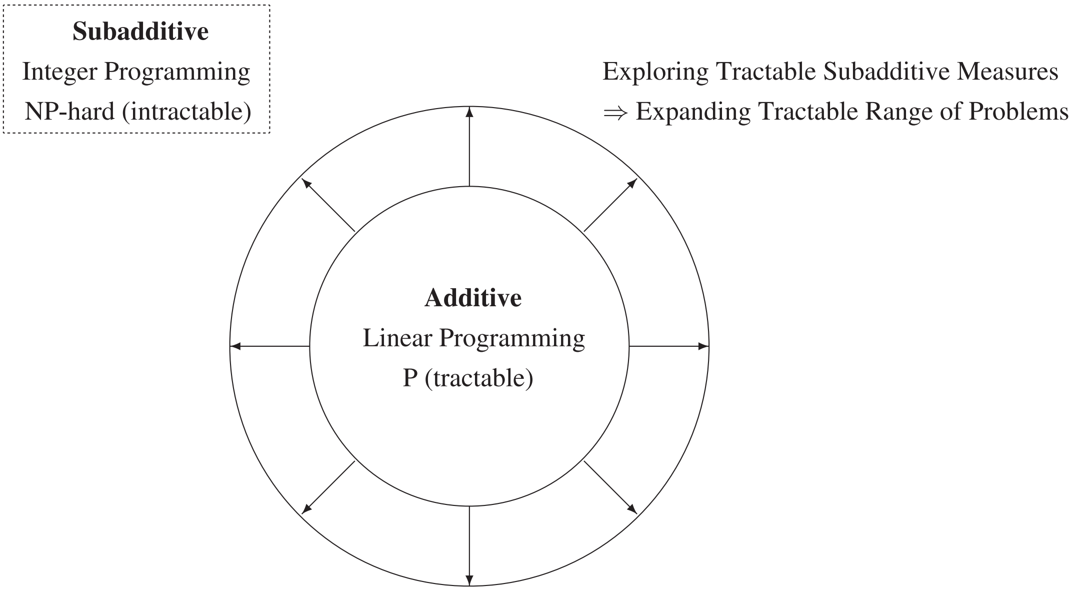


Fig. 3. Exploring the limits of subadditive approaches to enhance the possibility of computation.

1.3 Organization of This Paper

This paper is organized as follows. In Section 2, we briefly overview the use of subadditivity in optimization and in complexity theory. In Section 3, we give illustrative examples of some complexity measures by colored figures. In Section 4, we consider tight linear programming formulations of subadditive measures. Formulations are given for polytopes associated with formula complexity and more generally for set partition polytopes. In Section 5, we discuss clique constraints and their application to set partition polytopes in connection with subadditive functions. In particular, we give some interpretation of clique constraints for set partition polytopes in terms of subadditive functions. The paper is intended as an introductory lecture note and does not give an exhaustive survey. Since there are detailed treatments of the topics in existent survey and original articles [8, 21, 44, 45], we would like to put special emphasis on connection between optimization and complexity theory, and to give intuitive explanations by using colored figures.

2. Review on Use of Subadditivity in Optimization and Complexity Theory

We review the use of subadditivity in optimization and complexity theory in Section 2.1 and 2.2, respectively.

2.1 Linear Programming and Integer Programming: Additivity versus Subadditivity

In this subsection, we briefly overview the theory of integer programming duality to explain subadditive approaches in optimization. Although the duality theorem can be stated in more generalized forms such as conic mixed integer programming [31], we here focus on only integer programming for simplicity. We assume that the reader is familiar with the basics of the linear and integer programming theory. Please refer to the standard textbooks (e.g., [37]) for details on general theory and to the survey paper by Güzelsoy and Ralphs [8] for integer programming duality.

2.1.1 Linear Programming Relaxation and Linear Programming Duality

In this subsection, we review basics of linear programming and integer programming. To define a linear programming problem and an integer programming problem, we will use the following notation: A is an $m \times n$ constant matrix, b is an m-dimensional constant vector, c is an n-dimensional constant vector and x is an n-dimensional variable vector, where vectors are column vectors. In addition, a_j denotes the j-th column of A, and b_i, c_i, and x_i denote the i-th components of b, c and x, respectively. We define $[m] = \{1, \cdots, m\}$, and $[n] = \{1, \cdots, n\}$. We also denote by $\mathbb{N}$ the set of natural numbers, where it is assumed in this paper that $\mathbb{N}$ includes 0.

We consider the following integer programming problem:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b, \\ & x \in \mathbb{N}^n. \end{array} \tag{2.1}$$

We also consider the following linear programming relaxation of the integer programming problem (2.1):

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b, \\ & x \geq 0. \end{array} \tag{2.2}$$

In the following, we often restrict our attention to the case where each component of A and b is either 0 or 1; in such a case, the polytope given as the feasible region of (2.2) is called a *set partition polytope*.

The dual problem to the LP problem (2.2) is given as

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & A^T y + z = c, \\ & z \geq 0. \end{array} \tag{2.3}$$

Here, y is an m-dimensional variable vector and z is an n-dimensional variable vector. The constraints of the dual problem can be replaced with a single constraint $A^T y \leq c$ by deleting the variable vector z.

The duality theorem in linear programming is stated as follows.

Theorem 2.1 (Linear Programming Duality)**.**

- *(weak duality) For any x and (y, z) which are feasible in (2.2) and (2.3), respectively, we have $c^T x \geq b^T y$.*
- *(strong duality) If at least one of (2.2) and (2.3) has an optimal solution, then both of (2.2) and (2.3) have optimal solutions and their optimal value are the same.*

2.1.2 Integer Programming Duality and Subadditive Dual

In this subsection, we explain the duality theory in integer programming with connection to subadditivity. Some of the basic results developed in the theory of linear programming can be extended to integer programming. In particular, there are a long series of studies on integer programming duality, although it is not well known compared to linear programming duality.

A function $F : \Re^m \to \Re \cup \{+\infty\}$ is called *subadditive* if $F(y) + F(y') \geq F(y + y')$ holds for any $y, y' \in \Re^m$, where $\Re$ is the set of real numbers. We consider the following optimization problem:

$$\begin{array}{lll} \max & F(b) & \\ \text{s.t.} & F(a_j) \leq c_j & (j \in [n]), \\ & F(\vec{0}) = 0, & \\ & F : \text{subadditive}, & \end{array} \tag{2.4}$$

where $\vec{0} = (0, \cdots 0)$ and a function F is a variable of the problem.

In the literature of integer programming, a duality theorem is firstly shown by Johnson [14, 15]. The following duality theorem for the equality version of the integer programming problem (2.1) is given by Lasserre [26].

Theorem 2.2 (Integer Programming Duality [26])**.**

- *(weak duality) For any x and F which are feasible in (2.1) and (2.4), respectively, we have $c^T x \geq F(b)$.*
- *(strong duality) If at least one of (2.1) and (2.4) has an optimal solution, then both of (2.1) and (2.4) have optimal solutions and their optimal value are the same.*

Proof. We here show a proof based on Lasserre [26]. The first part of the theorem can be proven as follows:

$$F(b) = F(Ax) = F\left(\sum_{j=1}^{n} a_j x_j\right) \leq \sum_{j=1}^{n} F(a_j x_j) = \sum_{j=1}^{n} F(a_j) x_j \leq \sum_{j=1}^{n} c_j x_j = c^T x,$$

where we use the fact that F is subadditive and x_j is a nonnegative integer for $j \in [n]$.

To prove the second part of the theorem, we define a function $F^* : \Re^m \to \Re \cup \{\pm\infty\}$ by

$$F^*(d) = \inf\{c^T x \mid Ax = d,\ x \in \mathbb{N}^n\} \quad (d \in \Re^m). \tag{2.5}$$

We first show that F^* is a feasible solution of (2.4). By the weak duality and the assumption that at least one of (2.1) and (2.4) has an optimal solution, we have $F^*(d) > -\infty$. From this inequality and the definition of F^*, it can be shown that $F^*(\vec{0}) = 0$ and $F^*(d) > -\infty$ holds for all $d \in \Re^m$. It is easy to see that the function F^* satisfies $F(a_j) \leq c_j$ for all $j \in [n]$.

The subadditivity of the function F^* can be shown as follows, where $y, y' \in \Re^m$:

- Suppose $F^*(y) < +\infty$ and $F^*(y') < +\infty$, and let x and x' be optimal solutions of the integer programming problem (2.5) with $d = y$ and $d = y'$, respectively, i.e., $F^*(y) = c^T x$ and $F^*(y') = c^T x'$. Then, $x + x'$ is a feasible solution of the integer programming problem (2.5) with $d = y + y'$ because $y + y' = A(x + x')$. Therefore, we also have $F^*(y + y') \leq c^T(x + x') = F^*(y) + F^*(y')$.
- If $F^*(y) = +\infty$ or $F^*(y') = +\infty$, then it holds that $F^*(y) + F^*(y') = +\infty \geq F^*(y + y')$.

We finally show the second part of the theorem by using the function F^*. If (2.1) has an optimal solution x^*, then we have $c^T x^* = F^*(b)$ by the definition of the value $F^*(b)$. Hence, F^* is an optimal solution of (2.4) by the weak duality.

On the other hand, if (2.4) has an optimal solution, then $F^*(b) < +\infty$ holds, and therefore the definition of the value $F^*(b)$ implies that there exists an optimal solution x^* of (2.1) such that $c^T x^* = F^*(b)$. This implies, in particular, that x^* and F^* are optimal solutions of (2.1) and (2.4), respectively. □

The function F^* given by the optimal value $F^*(d)$ of the problem (2.5) is called the optimal value function. The difference between subadditivity and additivity can be roughly illustrated as in Figure 4 by using the optimal value function, where $m = 1$ is assumed. The red zigzag line corresponds to the optimal value function of the original integer programming problem, while the green straight line corresponds to the optimal value function of its linear programming relaxation. The green line can be obtained by calculating $F^*(d)$ after replacing $x \in \{0, 1\}$ by its relaxation $x \geq 0$. The complexity of the line reflects the complexity of the optimization problem.

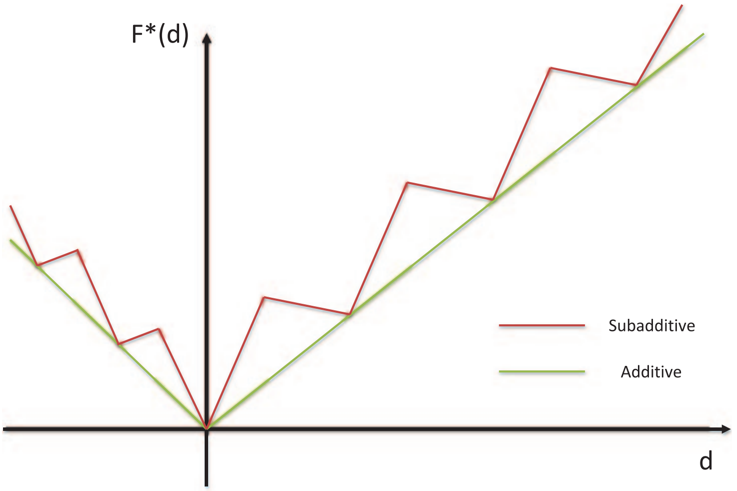


Fig. 4. The difference between additivity and subadditivity in terms of the optimal value function.

The set of all subadditive functions is huge and hard to represent in a compact manner. Hence, we need sophisticated techniques to cope with subadditive functions. To explore feasible solutions of the problem (2.4), a lot of theoretical and experimental results have been obtained in the literature as summarized in the survey of Güzelsoy and Ralphs [8]. They are also useful to explore hard problems in the theory of computational complexity as explained in the subsequent sections.

2.2 Subadditive Measures and LP-Based Methods in Complexity Theory

In this subsection, we explain basics notions in complexity theory, which are essential to understand linear programming based approaches to prove formula-size lower bounds. Approaches of these kinds are useful to prove lower bounds in many scenarios.

2.2.1 Formula Complexity and Formal Complexity Measure

First, we discuss a subadditive measure known as "formal complexity measure", which gives a lower bound on formula size. We give definitions of formula size and monotone formula size of a Boolean function. See the recent book [16] and also the classical one [47] for the detailed treatment on formula complexity.

Definition 2.3 (Formula Size). A *formula* is a binary tree with each leaf labeled by a literal (i.e., either a variable or its negation) and each internal node labeled by either of the binary connectives $\wedge$ and $\vee$. The *size* of a formula is defined as the number of its leaves. We define *formula size* $L(f)$ of a Boolean function f as the smallest size of a formula computing f. A *monotone* formula is a formula without negations. We also define $L_m(f)$ as the smallest size of a monotone formula computing a Boolean function f.

The complexity class $\mathbf{NC}^1$ denotes, in this paper, the non-uniform version of $\mathbf{NC}^1$, which is sometimes denoted by $\mathbf{NC}^1$/poly, and is defined as the set of problems computable by a logarithmic depth circuit. Spira [43] showed that it is equivalent to the class of languages computable by a polynomial size formula.

Theorem 2.4 ([43]). *Let $d(f)$ be the minimum depth of a circuit computing a Boolean function f. Then, $L(f) \leq 2^{d(f)}$ and $d(f) \leq O(\log L(f))$.*

The latter inequality is nontrivial while the former one is clear. By Theorem 2.4, proving a super-polynomial formula-size lower bound leads to a separation between $\mathbf{NC}^1$ and other complexity classes beyond $\mathbf{NC}^1$. Shannon [38] proved by a so-called counting argument that almost all Boolean functions with n input variables require circuit size of at least $\Omega(2^n/n)$. Using the counting argument, we can also show the following analogue for formulas.

Theorem 2.5 ([38]). *Almost all Boolean functions require formula size of at least $\Omega(2^n/\log n)$.*

Proof methods for formula-size lower bounds [18, 20] can be explained in terms of a notion of formal complexity measure, which is defined by a subadditivity constraint for formulas.

Definition 2.6 (Formal Complexity Measure). We define $\mathscr{B}$ as the set of Boolean functions and $\mathscr{N}$ as the set of natural numbers. A *formal complexity measure* is defined as a function $\mu : \mathscr{B} \to \mathscr{N}$ satisfying the following conditions:

- If f is computable by a literal or its negation, then $\mu(f) \leq 1$.
- Otherwise, μ satisfies
  - $\mu(f_1 \wedge f_2) \leq \mu(f_1) + \mu(f_2)$,
  - $\mu(f_1 \vee f_2) \leq \mu(f_1) + \mu(f_2)$.

Formula size itself is one of formal complexity measures, specifically, the largest one. An important property is that any formal complexity measure gives a formula-size lower bound.

Lemma 2.7. *For any Boolean function f and any formal complexity measure μ, $L(f) \geq \mu(f)$.*

Therefore, any sort of subadditive functions are useful to give formula-size lower bounds and worth to investigate. A good example such subadditive functions is a notion of graph entropy [42]. Note that a similar statement holds for the formula size of monotone formulas if we define a measure by removing "its negation" from the definition of formal complexity measure.

On the other hand, there are some known results which indicate the difficulty to give strong lower bounds of formal complexity measure. The next theorems state that a lower bound of formal complexity measure on a certain Boolean function must also work for many Boolean functions. Therefore, any proof of a strong lower bound on a certain Boolean function through formal complexity measure is as hard as that of many Boolean functions.

Theorem 2.8 ([36]). *For a formal complexity measure μ, we assume $\mu(f) > s$ for some Boolean function $f : \{0,1\}^n \to \{0,1\}$. Then, for at least $(1/4)$-fraction of all Boolean functions $g : \{0,1\}^n \to \{0,1\}$, we must have $\mu(g) \geq s/4$.*

We can also give the following bound by detailed analysis.

Theorem 2.9 ([36]). *If $\mu(f) > s$ holds for some Boolean function f, then for all $\varepsilon > 0$ we have*

$$\mu(g) \geq \Omega\left(\frac{s}{(n + \log(1/\varepsilon))^2}\right)$$

for at least $(1 - \varepsilon)$-fraction of all Boolean functions g on n input bits.

2.2.2 Protocol Partition Number and Rectangle Bound

We then explain two complexity measures of Boolean functions, called the protocol partition number and the rectangle bound. The protocol partition number exactly characterizes formula size through a communication game given by Karchmer and Wigderson [19]. The rectangle bound is useful to give a lower bound of formula size and can be formulated as an integer programming problem as explained in the next subsection.

We consider the following matrix called the communication matrix which comes from the game of Karchmer and

Wigderson [19].

Definition 2.10 (Communication Matrix). Given a Boolean function f, its *communication matrix* is defined as a matrix such that rows and columns of the matrix are indexed by the elements in $X = f^{-1}(1)$ and $Y = f^{-1}(0)$, respectively, and each cell of the matrix contains the set of indices i with $x_i \neq y_i$. For convenience, we often regard a communication matrix as a *relation* $R_f \subseteq X \times Y \times \{1, 2, \cdots, n\}$, which is given as

$$R_f = \{(x, y, i) \mid x \in X,\ y \in Y,\ x_i \neq y_i\}.$$

We also define a monotone version of the communication matrix and the relation associated with a monotone Boolean function f as

$$R_f^m = \{(x, y, i) \mid x \in X,\ y \in Y,\ x_i = 1,\ y_i = 0\}.$$

A *combinatorial rectangle* is a direct product $X' \times Y'$ with $X' \subseteq X$ and $Y' \subseteq Y$. A combinatorial rectangle $X' \times Y'$ is called *monochromatic* if every cell $(x, y) \in X' \times Y'$ contains the same index i.

A *cover* of a communication matrix R_f is a set of monochromatic rectangles such that every cell in $X \times Y$ is contained by at least one monochromatic rectangle in the set. We define two complexity measures related with cover as follows. For a combinatorial rectangle $X' \times Y'$, its *partition* is defined as a pair of $X'_1 \times Y'$ and $X'_2 \times Y'$ with $X' = X'_1 \cup X'_2$ and $X'_1 \cap X'_2 = \emptyset$, or a pair of $X' \times Y'_1$ and $X' \times Y'_2$ with $Y' = Y'_1 \cup Y'_2$ and $Y'_1 \cap Y'_2 = \emptyset$.

Definition 2.11 (Rectangle Bound and Protocol Partition Number). The *rectangle bound* for a communication matrix R_f, denoted as $C^D(R_f)$, is the minimum size of a cover with disjoint monochromatic rectangles. A *protocol partition* of R_f is a set of disjoint monochromatic rectangles obtained by a recursive partition of $X \times Y$. The minimum number of disjoint monochromatic rectangles in a protocol partition of R_f is denoted as $C^P(R_f)$, called *the protocol partition number*.

The following relation is known between the protocol partition number and the rectangle bound. For a proof, we recommend [28].

Theorem 2.12 ([18]). *For any Boolean function f, we have $C^D(R_f) \leq C^P(R_f) \leq 2^{O(\log^2 C^D(R))}$.*

The first inequality in Theorem 2.12 is easy to see from the definitions of the rectangle bound and the protocol partition number because a protocol partition itself is a cover with disjoint monochromatic rectangles.

Karchmer and Wigderson [19] show that formula complexity is equal to the protocol partition number.

Theorem 2.13 ([19]). *For any Boolean function f, we have $C^P(R_f) = L(f)$ and $C^P(R_f^m) = L_m(f)$.*

2.2.3 Linear Programming Relaxation of Rectangle Bound

Karchmer, Kushilevitz and Nisan [18] formulate the rectangle bound for a communication matrix as an integer programming problem and consider its LP relaxation. Let M be the set of all monochromatic rectangles and x_r be a variable associated with a monochromatic rectangle $r \in M$. The definition of the rectangle bound can be regarded as the optimal value of a certain set partition problem formulated as follows:

$$\begin{aligned} \min \quad & \sum_{r \in M} x_r \\ \text{s.t.} \quad & \sum_{r \ni c} x_r = 1 \quad \text{(for each cell } c \text{ in the matrix)}, \\ & x_r \in \{0, 1\} \quad \text{(for each monochromatic rectangle } r \in M). \end{aligned} \tag{2.6}$$

Its LP relaxation is given as

$$\begin{aligned} \min \quad & \sum_{r \in M} x_r \\ \text{s.t.} \quad & \sum_{r \ni c} x_r = 1 \quad \text{(for each cell } c \text{ in the matrix)}, \\ & x_r \geq 0 \quad \text{(for each monochromatic rectangle } r \in M). \end{aligned} \tag{2.7}$$

The dual problem of the LP relaxation (2.7) is given as

$$\begin{aligned} \max \quad & \sum_{c} w_c \\ \text{s.t.} \quad & \sum_{c \in r} w_c \leq 1 \quad \text{(for each monochromatic rectangle } r \in M), \\ & w_c \in \Re \quad \text{(for each cell } c \text{ in the matrix)}. \end{aligned} \tag{2.8}$$

From the LP duality theorem, construction of a feasible solution of the dual problem gives a formula-size lower bound. We define the *linear programming bound* **LP**(R) (LP bound, for short) as the optimal values of the linear programs (2.7) and (2.8) above, which are equal by the duality theorem.

Karchmer, Kushilevitz and Nisan [18] show that the LP bound cannot prove a lower bound larger than $4n^2$ for non-monotone formula size in general. Lee [29] proves that the LP bound of Karchmer, Kushilevitz and Nisan [18] is better (i.e., larger) than the quantum adversary bound of Laplante, Lee and Szegedy [24], which in turn subsumes most of known techniques such as Khrapchenko [20], its extension by Koutsoupias [22] and a key lemma used in the proof of Håstad [9] showing the current best formula-size lower bound of $n^{3-o(1)}$.

In Figure 5, we summarize the idea of LP based methods to prove formula-size lower bounds. Since the dual problem of the LP relaxation is a maximization problem, the objective function value of any feasible solution is smaller than that of the optimal solution. Some ideas to extend this method using subadditivity will be discussed in the subsequent sections with connection to the subadditive dual theory in integer programming.

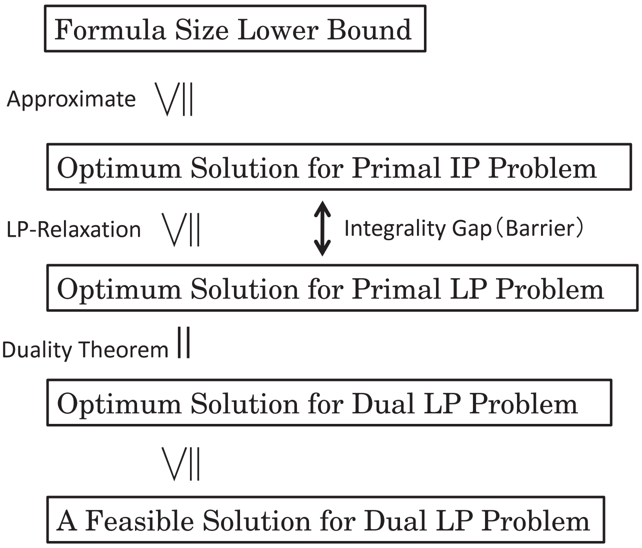
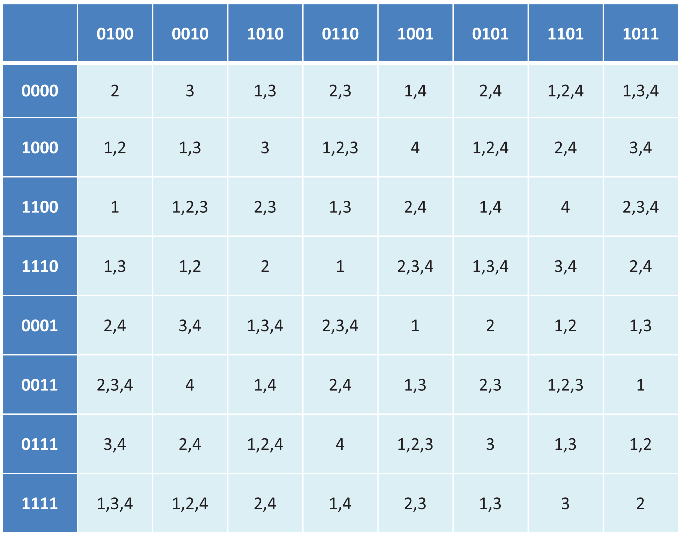

Formula Size Lower Bound
Approximate $\veebar$
Optimum Solution for Primal IP Problem
LP-Relaxation $\veebar$
Integrality Gap (Barrier)
Optimum Solution for Primal LP Problem
Duality Theorem ||
Optimum Solution for Dual LP Problem
$\veebar$
A Feasible Solution for Dual LP Problem

Fig. 5. Linear programming based approaches to prove formula-size lower bounds.

3. Illustrative Examples of Protocol Partition Number, Rectangle Bound, and LP Bound

In this section, we explain complexity measures of formulas by using colored figures. For this purpose, we consider a 4-bit Boolean function called Ambainis' function [1] which outputs 1 if and only if $x_1 \le x_2 \le x_3 \le x_4$, or $x_1 \ge x_2 \ge x_3 \ge x_4$ holds. In Figure 6, we present a formula for Ambainis' function and the associated communication matrix.

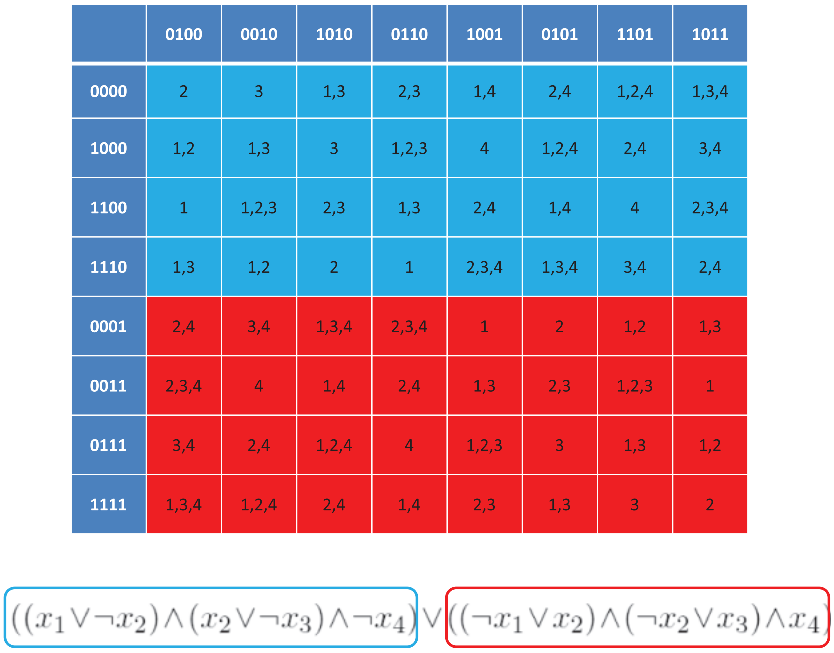


	0100	0010	1010	0110	1001	0101	1101	1011
0000	2	3	1,3	2,3	1,4	2,4	1,2,4	1,3,4
1000	1,2	1,3	3	1,2,3	4	1,2,4	2,4	3,4
1100	1	1,2,3	2,3	1,3	2,4	1,4	4	2,3,4
1110	1,3	1,2	2	1	2,3,4	1,3,4	3,4	2,4
0001	2,4	3,4	1,3,4	2,3,4	1	2	1,2	1,3
0011	2,3,4	4	1,4	2,4	1,3	2,3	1,2,3	1
0111	3,4	2,4	1,2,4	4	1,2,3	3	1,3	1,2
1111	1,3,4	1,2,4	2,4	1,4	2,3	1,3	3	2

$$((x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge \neg x_4) \vee ((\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge x_4)$$

Fig. 6. A communication matrix and a formula for Ambainis' function.

In Figure 7, we divide the whole rectangle into two combinatorial rectangles colored by blue and red. The blue rectangle on the upper side corresponds to blue-enclosed sub-formula on the left side. The red rectangle on the lower side corresponds to red-enclosed sub-formula on the right side.



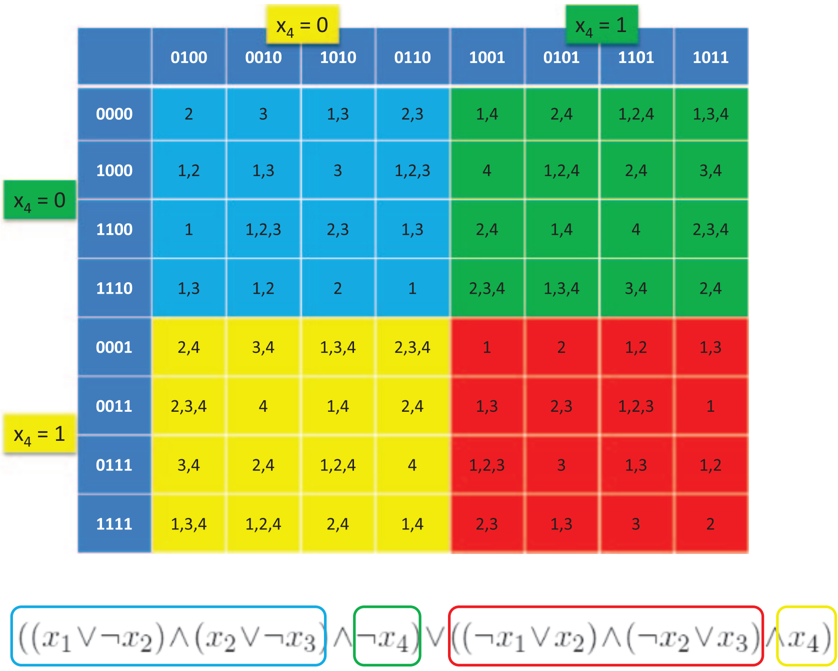
	0100	0010	1010	0110	1001	0101	1101	1011
0000	2	3	1,3	2,3	1,4	2,4	1,2,4	1,3,4
1000	1,2	1,3	3	1,2,3	4	1,2,4	2,4	3,4
1100	1	1,2,3	2,3	1,3	2,4	1,4	4	2,3,4
1110	1,3	1,2	2	1	2,3,4	1,3,4	3,4	2,4
0001	2,4	3,4	1,3,4	2,3,4	1	2	1,2	1,3
0011	2,3,4	4	1,4	2,4	1,3	2,3	1,2,3	1
0111	3,4	2,4	1,2,4	4	1,2,3	3	1,3	1,2
1111	1,3,4	1,2,4	2,4	1,4	2,3	1,3	3	2

$$((x_1 \lor \neg x_2) \land (x_2 \lor \neg x_3) \land \neg x_4) \lor ((\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \land x_4)$$

Fig. 7. A protocol partition at depth 1 for Ambainis' function.

In Figure 8, the blue rectangle in Figure 7 is partitioned into green and blue rectangles, and the red rectangle in Figure 7 is partitioned into yellow and red rectangles. Similarly to Figure 7, each rectangle corresponds to each sub-formula enclosed by four colors. Green and yellow rectangles are monochromatic but have different properties. The yellow rectangle corresponds to the positive literal x_4 because the 4th bit is 1 in each row and 0 in each column. The green rectangle corresponds to the negative literal $\neg x_4$ because the 4th bit is 0 in each row and 1 in each column.

In Figure 9, we finally partition the whole rectangles into these ten monochromatic rectangles. Each of the ten

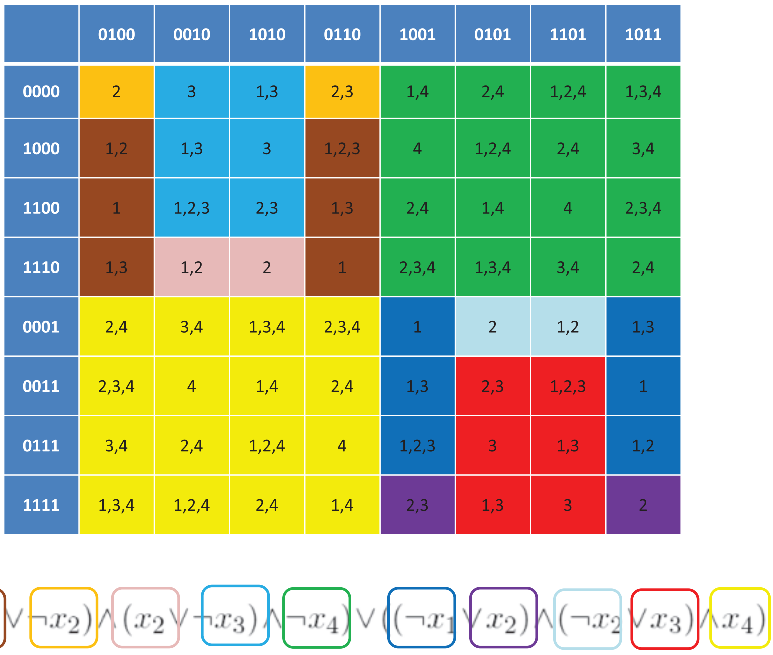


$x_4 = 0$
$x_4 = 1$
$x_4 = 0$
$x_4 = 1$

	0100	0010	1010	0110	1001	0101	1101	1011
0000	2	3	1,3	2,3	1,4	2,4	1,2,4	1,3,4
1000	1,2	1,3	3	1,2,3	4	1,2,4	2,4	3,4
1100	1	1,2,3	2,3	1,3	2,4	1,4	4	2,3,4
1110	1,3	1,2	2	1	2,3,4	1,3,4	3,4	2,4
0001	2,4	3,4	1,3,4	2,3,4	1	2	1,2	1,3
0011	2,3,4	4	1,4	2,4	1,3	2,3	1,2,3	1
0111	3,4	2,4	1,2,4	4	1,2,3	3	1,3	1,2
1111	1,3,4	1,2,4	2,4	1,4	2,3	1,3	3	2

$$((x_1 \lor \neg x_2) \land (x_2 \lor \neg x_3) \land \neg x_4) \lor ((\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \land x_4)$$

Fig. 8. A protocol partition at depth 2 for Ambainis' function.



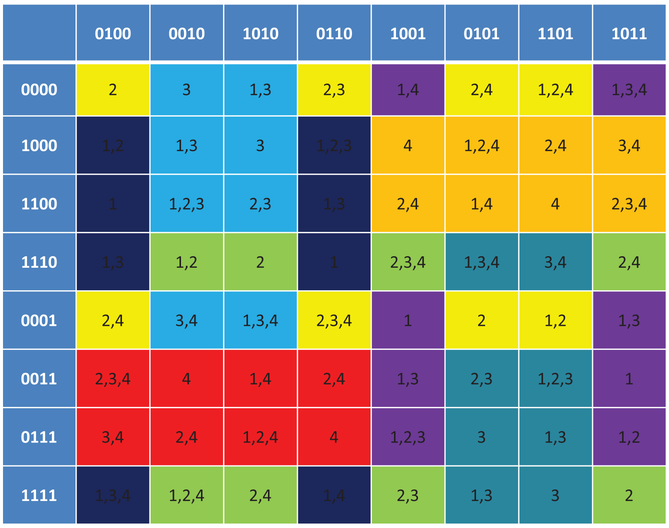
	0100	0010	1010	0110	1001	0101	1101	1011
0000	2	3	1,3	2,3	1,4	2,4	1,2,4	1,3,4
1000	1,2	1,3	3	1,2,3	4	1,2,4	2,4	3,4
1100	1	1,2,3	2,3	1,3	2,4	1,4	4	2,3,4
1110	1,3	1,2	2	1	2,3,4	1,3,4	3,4	2,4
0001	2,4	3,4	1,3,4	2,3,4	1	2	1,2	1,3
0011	2,3,4	4	1,4	2,4	1,3	2,3	1,2,3	1
0111	3,4	2,4	1,2,4	4	1,2,3	3	1,3	1,2
1111	1,3,4	1,2,4	2,4	1,4	2,3	1,3	3	2

$$\boxed{((x_1}\vee\boxed{\neg x_2)}\wedge\boxed{(x_2}\vee\boxed{\neg x_3)}\wedge\boxed{\neg x_4)}\vee\boxed{((\neg x_1}\vee\boxed{x_2)}\wedge\boxed{(\neg x_2}\vee\boxed{x_3)}\wedge\boxed{x_4)}$$

Fig. 9. A protocol partition at depth 3 for Ambainis' function.

colored monochromatic rectangles corresponds to each of ten literals. We can verify by an exhaustive search that the formula of size ten for Ambainis' function is optimal.

In Figure 10, eight colored monochromatic rectangles cover the whole rectangle. It shows that an upper bound of eight for the rectangle bound of Ambainis' function. The rectangle partition by eight rectangles, however, is not a protocol partition because any bisection of the whole rectangle divides at least one of the eight monochromatic rectangles. The example shows that there are some cases in which the rectangle bound is strictly smaller than the formula size.

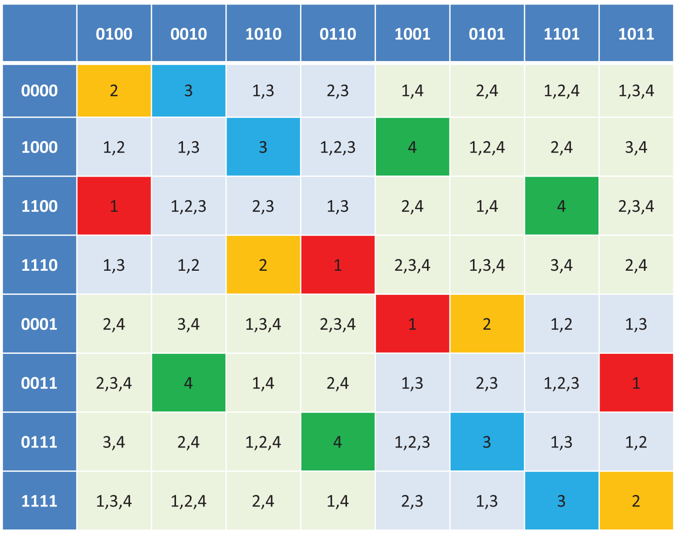


	0100	0010	1010	0110	1001	0101	1101	1011
0000	2	3	1,3	2,3	1,4	2,4	1,2,4	1,3,4
1000	1,2	1,3	3	1,2,3	4	1,2,4	2,4	3,4
1100	1	1,2,3	2,3	1,3	2,4	1,4	4	2,3,4
1110	1,3	1,2	2	1	2,3,4	1,3,4	3,4	2,4
0001	2,4	3,4	1,3,4	2,3,4	1	2	1,2	1,3
0011	2,3,4	4	1,4	2,4	1,3	2,3	1,2,3	1
0111	3,4	2,4	1,2,4	4	1,2,3	3	1,3	1,2
1111	1,3,4	1,2,4	2,4	1,4	2,3	1,3	3	2

Fig. 10. A rectangle partition of Ambainis' function.

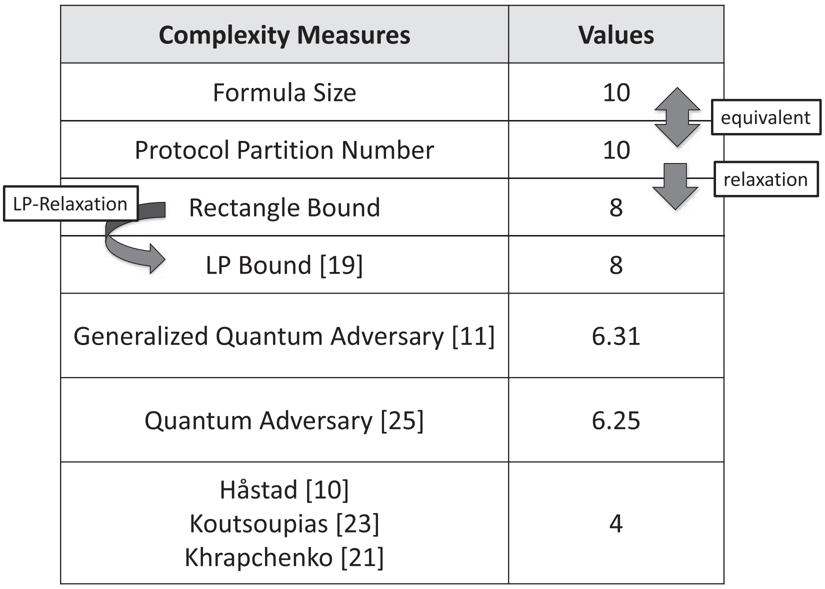
In Figure 11, we illustrate how to obtain a feasible solution of the dual of the LP relaxation for Ambainis' function. We assign weight $1/2$ to each colored cell containing only one index, and assign 0 to other cells. It is easy to see that the sum of all weights is 8 and the assignment satisfies all the constraints of the dual of the LP relaxation. The solution shows that the LP bound of 8 is the same as the rectangle bound.

In Figure 12, we give a summary of the complexity measures discussed in this section. By Theorem 2.13, the protocol partition number of 10 is the same as the formula size. While both of the rectangle bound and the LP bound have the same value of 8 in this case, there is an integrality gap between integer programming and its linear



	0100	0010	1010	0110	1001	0101	1101	1011
0000	2	3	1,3	2,3	1,4	2,4	1,2,4	1,3,4
1000	1,2	1,3	3	1,2,3	4	1,2,4	2,4	3,4
1100	1	1,2,3	2,3	1,3	2,4	1,4	4	2,3,4
1110	1,3	1,2	2	1	2,3,4	1,3,4	3,4	2,4
0001	2,4	3,4	1,3,4	2,3,4	1	2	1,2	1,3
0011	2,3,4	4	1,4	2,4	1,3	2,3	1,2,3	1
0111	3,4	2,4	1,2,4	4	1,2,3	3	1,3	1,2
1111	1,3,4	1,2,4	2,4	1,4	2,3	1,3	3	2

Fig. 11. A solution structure for the dual of the LP bound for Ambainis' function.



Complexity Measures	Values
Formula Size	10
Protocol Partition Number	10
Rectangle Bound	8
LP Bound [19]	8
Generalized Quantum Adversary [11]	6.31
Quantum Adversary [25]	6.25
Håstad [10] Koutsoupias [23] Khrapchenko [21]	4

equivalent
relaxation
LP-Relaxation

Fig. 12. Comparison of complexity measures for Ambainis' function.

programming relaxation in general. Other complexity measures such as Khrapchenko [20] and quantum adversary bounds [10, 24] are smaller than the value of the LP bound as mentioned in Section 2.2.3.

4. Tight LP Formulations for Integer Programming Problems via Subadditivity

In this section, we discuss tight LP formulations for some IP problems by using subadditivity. In the first subsection we introduce a new lower bound for the formula size by using an extended formulation of the LP bound, and in the second subsection we discuss a tight LP formulation for a more general IP problem (2.1).

4.1 Quasi-Additive Bound: LP Formulation of Formal Complexity Measure

In this subsection, we devise a stronger version of the LP bound named "the quasi-additive bound," which is derived from a concept of subadditive rectangle measure by Hrubeš, Jukna, Kulikov and Pudlák [11]. While the new bound is a natural extension of the original LP bound (2.8) defined as a relaxation of the rectangle bound in Section 2.2.3, it is shown that the new bound can surpass the rectangle bound [45]. In fact, the quasi-additive bound is strong enough to give the protocol partition number of any Boolean function.

Let R be a communication matrix of a Boolean function, and denote by Γ the set of all combinatorial rectangles. We also denote by $M \subseteq \Gamma$ the set of all monochromatic rectangles of R. We call $\mu : \Gamma \to \Re$ a *subadditive rectangle measure* if it satisfies the following two properties:

(1) Normalization: $\mu(m) \leq 1$ for each monochromatic rectangle $m \in M$.

(2) Subadditivity: $\mu(r) \leq \mu(r_1) + \mu(r_2)$ for each combinatorial rectangle $r \in \Gamma$ and its arbitrary partition r_1 and r_2.

Hrubeš et al. [11] show the following theorem by a simple inductive argument.

Theorem 4.1 ([11]). *For any communication matrix R and any subadditive rectangle measure μ, the value $\mu(X \times Y)$ gives a lower bound of the protocol partition number $C^P(R)$, where $X \times Y$ is the whole rectangle of the communication matrix R.*

A remarkable fact is that, if we strengthen the condition "Subadditivity" to

(3) Additivity: $\mu(r) = \mu(r_1) + \mu(r_2)$ for each combinatorial rectangle $r \in \Gamma$ and its arbitrary partition r_1 and r_2, then the resulting rectangle measure gives the same bound as the LP bound (2.8).

We then give a definition of quasi-additive bound.

Definition 4.2 (Quasi-Additive Bound [45]). Let C be the set of all cells in a communication matrix R. We define the *quasi-additive bound* $\mathbf{QA}(R)$ as the optimal value of the following LP formulation:

$$\begin{aligned}
\max \quad & \sum_{c \in C} V_c \\
\text{s.t.} \quad & \sum_{c \in r} V_c + \sum_{c \notin r} V_{c,r} \leq 1 \quad \text{(for each } r \in M), \\
& \sum_{c \notin r_1} V_{c,r_1} + \sum_{c \notin r_2} V_{c,r_2} \geq \sum_{c \notin r} V_{c,r} \quad \text{(for each } r \in \Gamma \text{ and its arbitrary partition } r_1 \text{ and } r_2).
\end{aligned} \tag{4.1}$$

This bound is called the quasi-additive bound because it can be regarded as an extension of the original LP bound (2.8) which has additive property. It is potentially strong enough to prove the tight bound for the protocol partition number of any Boolean function. On the other hand, the numbers of variables and inequalities explode exponentially compared to the original LP bound. We can eliminate the redundancy of the formulation (4.1) of the quasi-additive bound by summarizing variables as $\overline{V}_r = \sum_{c \notin r} V_{c,r}$ for each combinatorial rectangle r and adding a constraint $\overline{V}_{X \times Y} = 0$, although the size of the formulation is still exponentially large.

From now on, we clarify the potential of the quasi-additive bound to prove formula-size lower bounds. First, we show that the quasi-additive bound is stronger than the LP bound and gives a lower bound of the protocol partition number.

Lemma 4.3 ([45]). *For any communication matrix R, $\mathbf{LP}(R) \leq \mathbf{QA}(R) \leq C^P(R)$.*

Proof. If we set $V_{c,r} = 0$ for each c and r in the LP formulation (4.1), the resulting LP problem is nothing but the formulation of the LP bound (2.8). Hence, we have $\mathbf{LP}(R) \leq \mathbf{QA}(R)$.

To prove $\mathbf{QA}(R) \leq C^P(R)$, we regard $\mu(r) = \sum_{c \in r} V_c + \sum_{c \notin r} V_{c,r}$ as a rectangle measure and show that μ is a subadditive rectangle measure. We have $\mu(X \times Y) = \sum_{c \in C} V_c$, which is equal to the objective function value of (4.1), because $C = X \times Y$ is the whole rectangle associated with R. The condition "Normalization" for μ is equivalent to the first constraint in (4.1), while "Subadditivity" for μ is equivalent to the second constraint in (4.1) due to the additivity of $\sum_{c \in r} V_c$. Thus, if assignments of V_c and $V_{c,r}$ satisfy the first and second constraints of (4.1), then μ is a subadditive rectangle measure. Consequently, we have $\mathbf{QA}(R) \leq C^P(R)$ because $\mu(X \times Y) \leq C^P(R)$ by Theorem 4.1. □

There are some examples for which quasi-additive bound surpasses the rectangle bound. This fact is interesting since an extension of LP relaxation can be stronger than the original IP formulation.

Theorem 4.4 ([45]). *There exists a communication matrix R such that $\mathbf{QA}(R) > C^D(R)$.*

In fact, we can prove that the quasi-additive bound is strong enough to give the protocol partition number.

Theorem 4.5 ([45]). *For any communication matrix R, $\mathbf{QA}(R) = C^P(R)$.*

Proof. By Lemma 4.3, it suffices to show that $\mathbf{QA}(R) \geq C^P(R)$. For this, we construct a feasible solution of the problem (4.1) whose objective value is equal to $C^P(X \times Y)$ ($= C^P(R)$) by using the protocol partition number $C^P(r)$ for each combinatorial rectangle r. We arbitrarily take each assignment of V_c under the condition $\sum_{c \in C} V_c = C^P(X \times Y)$. Then, we assign $V_{c,r}$ so as to satisfy $\sum_{c \notin r} V_{c,r} = C^P(r) - \sum_{c \in r} V_c$; such assignments exist because every variable $V_{c,r}$ always appears in the form of $\sum_{c \notin r} V_{c,r}$. By definition, we have $C^P(r) = 1$ for each monochromatic rectangle r. Moreover, $C^P(r)$ is subadditive for each combinatorial rectangle r because the protocol partition number is defined as the minimization over arbitrary partitions. Therefore, the assignments satisfy all the constraints of the quasi-additive bound. □

From Theorems 2.13 and 4.5 we obtain the following relationship between the quasi-additive bound and the formula complexity.

Corollary 4.6. *For any Boolean function f, $\mathbf{QA}(R_f) = L(f)$ and $\mathbf{QA}(R_f^m) = L_m(f)$.*

By Theorem 4.5, we can regard the quasi-additive bound (maximization problem) as a dual problem of the protocol partition number (minimization problem). We should emphasize that any feasible solution of the formulation (4.1) of the quasi-additive bound gives a lower bound of the formula size by Corollary 4.6, while protocol partitions give only upper bounds of the formula size.

We illustrate how to construct a feasible solution of (4.1) and obtain a lower bound of the formula size by using an example of the 3-bit majority function. For a positive integer n, the majority function $\mathbf{MAJ}_n : \{0,1\}^n \to \{0,1\}$ is defined by

$$\mathbf{MAJ}_n(x_1, \cdots, x_n) = \begin{cases} 1 & \left(\sum_{i=1}^{n} x_i \geq \left\lceil \frac{n}{2} \right\rceil\right), \\ 0 & \text{(otherwise).} \end{cases}$$

Proposition 4.7. $\mathbf{QA}(R_{\mathbf{MAJ}_3}) \geq 5$.

Proof. We consider a submatrix of the communication matrix of the 3-bit majority function shown in Figure 13. We focus on a triplet of three cells $c_1 = (011, 100), c_2 = (101, 010), c_3 = (110, 001)$, each of which has three indices. We give the weight $V_c = -1/3$ for each cell with three indices (i.e., c is either c_1, c_2, or c_3), $V_c = 1$ for each cell from the other six cells with one index, and $V_c = 0$ for each cell not in the submatrix.

	100	010	001
110	2	1	1,2,3
101	3	1,2,3	1
011	1,2,3	3	2

Fig. 13. A submatrix of the communication matrix of $\mathbf{MAJ_3}$.

We give each assignment of $V_{c,r}$ in the following way. If a combinatorial rectangle r contains two cells of the triplet, e.g., c_2 and c_3, we assign $V_{c_1,r} = -1/3$ for the remaining cell c_1. If a combinatorial rectangle r contains one cell of the triplet, e.g., c_1, we assign $V_{c_2,r} = 1/6$ and $V_{c_3,r} = 1/6$ for the remaining two cells. In the remaining cases, we assign $V_{c,r} = 0$. Then, we can verify that the assignments satisfy all the constraints of the quasi-additive bound. As a consequence, we have the lower bound of 5. □

While the quasi-additive bound is derived from the notion of formal complexity measure, its LP dual may be of independent interest. For $r, r_1, r_2 \in \Gamma$, we denote $r \mapsto (r_1, r_2)$ if a pair of two combinatorial rectangles r_1 and r_2 is a partition of a combinatorial rectangles r. Then, the LP dual of the quasi-additive bound (4.1) is formulated as follows, where we use new variables $Z^r_{r_1,r_2}$ associated with each $r \mapsto (r_1, r_2)$, and $X \times Y \in \Gamma$ denotes the whole combinatorial rectangle:

$$\begin{array}{lll} \min & \sum_{r \in M} Z_r & \\ \text{s.t.} & \sum_{c \in r \in M} Z_r = 1 & \text{(for each } c \in C\text{)}, \\ & \sum_{r \mapsto (\hat{r}, r_2)} Z^r_{\hat{r}, r_2} = \sum_{\hat{r} \mapsto (r_1, r_2)} Z^{\hat{r}}_{r_1, r_2} + Z_{\hat{r}} & \text{(for each } \hat{r} \in M \text{ with } \hat{r} \neq X \times Y\text{)}, \\ & \sum_{r \mapsto (\hat{r}, r_2)} Z^r_{\hat{r}, r_2} = \sum_{\hat{r} \mapsto (r_1, r_2)} Z^{\hat{r}}_{r_1, r_2} & \text{(for each } \hat{r} \in \Gamma \text{ with } \hat{r} \notin M \text{ and } \hat{r} \neq X \times Y\text{)}, \\ & Z^r_{r_1, r_2} = Z^r_{r_2, r_1} \geq 0 & \text{(for each } r \mapsto (r_1, r_2)\text{)}, \\ & Z_r \geq 0 & \text{(for each } r \in M\text{)}. \end{array} \tag{4.2}$$

We can interpret the value $\sum_{\hat{r} \mapsto (r_1, r_2)} Z^{\hat{r}}_{r_1, r_2}$ as the total number of protocol partitions from $\hat{r}$. On the other hand, we can interpret $\sum_{r \mapsto (\hat{r}, r_2)} Z^r_{\hat{r}, r_2}$ as the total number of protocol partitions into $\hat{r}$ and another combinatorial rectangle. Therefore, if $\hat{r}$ is not monochromatic, these numbers should coincide in principle. We can regard this interpretation as some kind of the conservation law for the protocol partition number. Relevant to this formulation, Kamiyama [17] showed that the quasi-additive bound is equivalent to the dual problem of the LP relaxation of some integer programming computing the protocol partition number.

4.2 Tight Representations for Set Partition Polytopes Derived from Subadditive Dual

In the previous subsection, we have discussed tight LP formulation concerned with formula complexity particularly. In this subsection, we generalize this idea and introduce a novel tight LP formulation for set partition polytopes treated in Section 2. It is derived from the theory of integer programming duality, as a dual for the subadditive dual.

Associated with the integer programming problem (2.1), we consider the following linear programming problem, where $I(v) = \{i \mid v_i = 1\}$ for a vector v:

$$\begin{array}{lll} \max & b^T y & \\ \text{s.t.} & a_j^T y + z_{I(a_j)} \le c_j & (j \in [n]), \\ & z_{I_1} + z_{I_2} \ge z_{I_1 \cup I_2} & (I_1, I_2 \subseteq [m],\ I_1 \cap I_2 = \emptyset), \\ & z_{I(b)} = z_\emptyset = 0. & \end{array} \tag{4.3}$$

The problem (4.3) is equivalent to the dual (2.4) of the integer programming problem (2.1), as follows.

Lemma 4.8. *Suppose that the LP formulation (2.4) has an optimal solution $F^* : \Re^n \to \Re \cup \{+\infty\}$ such that $F^*(v) < +\infty$ for every $v \in \{0,1\}^m$. Then, for $A \in \{0,1\}^{m\times n}$ and $b \in \{0,1\}^m$, the LP formulations (2.4) and (4.3) are equivalent in the sense that they have the same optimal value.*

Proof. We first show that the optimal value of (4.3) is less than or equal to the optimal value of (2.4). For this, we show that for every feasible solution of (4.3), there exists a feasible solution of (2.4) with has the same objective function value.

For a feasible solution z, y of (4.3), we define a function $F : \{0,1\}^n \to \Re$ by

$$F(v) = v^T y + z_{I(v)} \quad (v \in \{0,1\}^n). \tag{4.4}$$

Then, we have

$$F(a_j) \le c_j \iff a_j^T y + z_{I(a_j)} \le c_j, \tag{4.5}$$

$$F \text{ is subadditive in the domain } \{0,1\}^n \iff z_{I_1} + z_{I_2} \ge z_{I_1\cup I_2}\ (I_1, I_2 \subseteq [m],\ I_1 \cap I_2 = \emptyset); \tag{4.6}$$

in addition, it holds that $F(b) = b^T y$ and $F(\vec{0}) = 0$ since $z_{I(b)} = z_\emptyset = 0$. Since $F(v)$ is a finite value for every $v \in \{0,1\}^n$, it is not difficult to obtain a subadditive function $\tilde{F} : \Re^n \to \Re \cup \{+\infty\}$ such that $\{v \in \Re^n \mid \tilde{F}(v) < +\infty\} = \{v \in \Re^n \mid v \ge 0\}$ and $\tilde{F}(v) = F(v)$ for $v \in \{0,1\}^n$. Hence, the function $\tilde{F}$ is a feasible solution of (4.3) having the same objective function value as z, y.

We then show that the optimal value of (4.3) is more than or equal to the optimal value of (2.4), which can be done in a similar way as the proof above. Let $F : \Re^n \to \Re \cup \{+\infty\}$ be an optimal solution of (2.4) such that $F(v) < +\infty$ for every $v \in \{0,1\}^m$. Then, we can easily find z and y in (4.3) satisfying the equation (4.4) and $z_{I(b)} = z_\emptyset = 0$. Moreover, (4.5) and (4.6) imply that such z and y give a feasible solution of the problem (4.3). Since $b^T y = F(b)$ holds by (4.4) and $z_{I(b)} = 0$, we see that the optimal value of (4.3) is more than or equal to the optimal value of (2.4). □

We consider the LP dual of the above formulation (4.3). We denote by $\mathcal{I}$ the set of all (unordered) pairs of nonempty $I_1, I_2 \subsetneq [m]$ such that $I_1 \cap I_2 \ne \emptyset$. For each $(I_1, I_2) \in \mathcal{I}$, we associate a variable P_{I_1,I_2}. Then, the following dual problem of (4.3) is formulated as follows:

$$\begin{array}{lll} \min & c^T x & \\ \text{s.t.} & Ax = b, & \\ & x \ge 0, & \\ & \displaystyle\sum_{I_2:(I_1,I_2)\in\mathcal{I}} P_{I_1,I_2} = \sum_{(I'_1,I'_2)\in\mathcal{I}:I'_1\cup I'_2=I_1} P_{I'_1,I'_2} + x_j & (I_1 = I(a_j) \text{ for some } j \in [n]), \\ & \displaystyle\sum_{I_2:(I_1,I_2)\in\mathcal{I}} P_{I_1,I_2} = \sum_{(I'_1,I'_2)\in\mathcal{I}:I'_1\cup I'_2=I_1} P_{I'_1,I'_2} & (\forall I_1 \subseteq [m] : I_1 \notin \{I(a_j) \mid j = 1,2,\ldots,n\} \cup \{\emptyset, I(b)\}), \\ & P_{I_1,I_2} \ge 0 & ((I_1, I_2) \in \mathcal{I}). \end{array} \tag{4.7}$$

From Theorem 2.2 and Lemma 4.8, we can obtain the following theorem.

Theorem 4.9. *Suppose that $A \in \{0,1\}^{m\times n}$ and $b \in \{0,1\}^m$. Then, the optimal value of the LP formulation (4.7) is equal to that of the IP formulation (2.1).*

To see the difference between the LP formulation (4.7) and the original LP relaxation (2.2), we show two examples. First, we consider the case where the constraint $Ax = b$ in (2.2) is given as

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

It is easy to see that $(x_1, x_2, x_3) = (1/2, 1/2, 1/2)$ is the unique solution of (2.2), while there exists no integral solution. Additional constraints in the LP formulation (4.7) are given as follows:

$$P_{\{1,2\},\{3\}} = P_{\{1\},\{2\}} + x_1 \quad (I_1 = \{1,2\}), \tag{4.8}$$

$$P_{\{2,3\},\{1\}} = P_{\{2\},\{3\}} + x_2 \quad (I_1 = \{2,3\}), \tag{4.9}$$

$$P_{\{1,3\},\{2\}} = P_{\{1\},\{3\}} + x_3 \quad (I_1 = \{1,3\}), \tag{4.10}$$

$$P_{\{1\},\{2\}} + P_{\{1\},\{3\}} + P_{\{1\},\{2,3\}} = 0 \quad (I_1 = \{1\}), \tag{4.11}$$

$$P_{\{2\},\{1\}} + P_{\{2\},\{3\}} + P_{\{2\},\{1,3\}} = 0 \quad (I_1 = \{2\}), \tag{4.12}$$

$$P_{\{3\},\{1\}} + P_{\{3\},\{2\}} + P_{\{3\},\{1,2\}} = 0 \quad (I_1 = \{3\}), \tag{4.13}$$

and $P_{I_1,I_2} \geq 0$ for any $(I_1, I_2) \in \mathcal{I}$. Substituting (4.8) to (4.13), we have

$$x_1 = -(P_{\{1\},\{2\}} + P_{\{2\},\{3\}} + P_{\{3\},\{1\}}) \leq 0.$$

Similarly, we have $x_2, x_3 \leq 0$. Consequently, these additional constraints in (4.7) rule out the solution $(x_1, x_2, x_3) = (1/2, 1/2, 1/2)$.

Next, we consider the case where the constraint $Ax = b$ of (2.2) is given as

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

In this case, (2.2) has two integral solutions $(x_1, x_2, x_3, x_4) = (1, 0, 1, 0), (0, 1, 0, 1)$ and infinitely many non-integral solutions given by the convex combination of the two integral vectors, i.e.,

$$(x_1, x_2, x_3, x_4) = \alpha \cdot (1, 0, 1, 0) + (1 - \alpha) \cdot (0, 1, 0, 1)$$

with $0 \leq \alpha \leq 1$. That is, the set of feasible solutions in (2.2) is equal to the convex hull of the set of feasible solutions in the integer programming problem (2.1).

The additional constraints in (4.7) are given as:

$$\begin{aligned}
P_{\{1,2\},\{3\}} + P_{\{1,2\},\{4\}} + P_{\{1,2\},\{3,4\}} &= P_{\{1\},\{2\}} + x_1 && (I_1 = \{1,2\}), \\
P_{\{2,3\},\{1\}} + P_{\{2,3\},\{4\}} + P_{\{2,3\},\{1,4\}} &= P_{\{2\},\{3\}} + x_2 && (I_1 = \{2,3\}), \\
P_{\{3,4\},\{1\}} + P_{\{3,4\},\{2\}} + P_{\{3,4\},\{1,2\}} &= P_{\{3\},\{4\}} + x_3 && (I_1 = \{3,4\}), \\
P_{\{1,4\},\{2\}} + P_{\{1,4\},\{3\}} + P_{\{1,4\},\{2,3\}} &= P_{\{1\},\{4\}} + x_4 && (I_1 = \{1,4\}), \\
P_{\{1,3\},\{2\}} + P_{\{1,3\},\{4\}} + P_{\{1,3\},\{2,4\}} &= P_{\{1\},\{3\}} && (I_1 = \{1,3\}), \\
P_{\{2,4\},\{1\}} + P_{\{2,4\},\{3\}} + P_{\{2,4\},\{1,3\}} &= P_{\{2\},\{4\}} && (I_1 = \{2,4\}), \\
P_{\{1\},\{2\}} + P_{\{1\},\{3\}} + P_{\{1\},\{4\}} + P_{\{1\},\{2,3\}} + P_{\{1\},\{2,4\}} + P_{\{1\},\{3,4\}} + P_{\{1\},\{2,3,4\}} &= 0 && (I_1 = \{1\}), \\
P_{\{2\},\{1\}} + P_{\{2\},\{3\}} + P_{\{2\},\{4\}} + P_{\{2\},\{1,3\}} + P_{\{2\},\{1,4\}} + P_{\{2\},\{3,4\}} + P_{\{2\},\{1,3,4\}} &= 0 && (I_1 = \{2\}), \\
P_{\{3\},\{1\}} + P_{\{3\},\{2\}} + P_{\{3\},\{4\}} + P_{\{3\},\{1,2\}} + P_{\{3\},\{1,4\}} + P_{\{3\},\{2,4\}} + P_{\{3\},\{1,2,4\}} &= 0 && (I_1 = \{3\}), \\
P_{\{4\},\{2\}} + P_{\{4\},\{3\}} + P_{\{4\},\{1\}} + P_{\{4\},\{1,2\}} + P_{\{4\},\{1,3\}} + P_{\{4\},\{2,3\}} + P_{\{4\},\{1,2,3\}} &= 0 && (I_1 = \{4\}), \\
P_{\{1,2,3\},\{4\}} &= P_{\{1\},\{2,3\}} + P_{\{2\},\{1,3\}} + P_{\{3\},\{1,3\}} && (I_1 = \{1,2,3\}), \\
P_{\{1,2,4\},\{3\}} &= P_{\{1\},\{2,4\}} + P_{\{2\},\{1,4\}} + P_{\{4\},\{1,2\}} && (I_1 = \{1,2,4\}), \\
P_{\{1,3,4\},\{2\}} &= P_{\{1\},\{3,4\}} + P_{\{3\},\{1,4\}} + P_{\{4\},\{1,3\}} && (I_1 = \{1,3,4\}), \\
P_{\{2,3,4\},\{1\}} &= P_{\{2\},\{3,4\}} + P_{\{3\},\{2,4\}} + P_{\{4\},\{2,3\}} && (I_1 = \{2,3,4\}),
\end{aligned}$$

and $P_{I_1,I_2} \geq 0$ for any $(I_1, I_2) \in \mathcal{I}$. If we assign

$$\begin{aligned}
P_{\{1,2\},\{3,4\}} &= x_1, \\
P_{\{2,3\},\{1,4\}} &= x_2, \\
P_{\{3,4\},\{1,2\}} &= x_3, \\
P_{\{1,4\},\{2,3\}} &= x_4,
\end{aligned}$$

and 0 to all other variables P_{I_1,I_2}, we can easily verify that these assignments satisfy all the above constraints. Therefore, the additional constraints are redundant in this case.

We finally give a brief review of tight LP formulations of combinatorial optimization problems. Since Edmonds [6] gave a tight LP formulation of the matching problem on undirected graphs, finding tight LP formulations of combinatorial optimization problems becomes an important research topic in polyhedral combinatorics. From this viewpoint, a technique called lift-and-project method is useful since it systematically incorporate tighter constraints into any LP formulation. There are several lift-and-project methods such as Sherali and Adams [39, 40], Balas, Ceria and Cornuéjols [2], Lovász and Schrijver [30], and Lasserre [25]. These techniques have attracted much attention in researches on polyhedral combinatorics and optimization. Laurent [27] gave a comparison among these techniques. Among the several techniques, the technique of Sherali and Adams [39, 40] has some advantages as the strongest one for LP formulations with relatively simpler descriptions. These lift-and-project methods can give tight LP formulations of any 0-1 integer programming problem. Sherali and Lee [41] studied an application of the lift-and-project method of Sherali-Adams [39, 40] to the set partition polytope.

In contrast, our tight LP formulation for the integer programming problem (2.1) given in this subsection is completely different from those given by the lift-and-project methods. Although the number of constrains explodes exponentially because the problem is NP-hard in general, the description is relatively succinct.

5. Clique Inequalities and Subadditivity

In this section, we first review a technique to improve formula-size lower bounds obtained from the LP bound. This improvement is achieved by using clique inequalities, which give an important class of inequalities in the theory of integer programming (see, e.g., [32, 37]). We illustrate the power of this technique by an application to the 3-bit majority function. We then discuss connection between clique constraints and subadditive functions.

5.1 LP Bound with Clique Inequalities

We explain a stronger LP bound using clique inequalities. Let $G = (V, E)$ be an undirected graph such that the node set V is given by the set of all monochromatic rectangles (i.e., $V = M$), and two nodes are connected by an edge in E if and only if the corresponding two monochromatic rectangles intersect. A clique is a set of nodes such that every pair of nodes in the set is connected by an edge. Given a set q of monochromatic rectangles corresponding to a clique in the graph G, a clique inequality is given by

$$\sum_{r \in q} x_r \leq 1.$$

This constraint is valid for all (integral) feasible solutions of the integer programming problem (2.6) since (2.6) is a disjoint cover problem, i.e., we can assign the value one to at most one monochromatic rectangle in a clique in G.

We consider the following LP problem obtained by adding all clique inequalities to the LP bound (2.7):

$$\begin{array}{lll} \min & \sum_{r \in M} x_r & \\ \text{s.t.} & \sum_{r \ni c} x_r = 1 & \text{(for each cell } c \text{ in the matrix)}, \\ & \sum_{r \in q} x_r \leq 1 & \text{(for each clique } q \text{ in } G), \\ & x_r \geq 0 & \text{(for each monochromatic rectangle } r \in M). \end{array} \tag{5.1}$$

Its dual problem can be written as

$$\begin{array}{lll} \max & \sum_{c} w_c + \sum_{q} z_q & \\ s.t. & \sum_{c \in r} w_c + \sum_{q \ni r} z_q \leq 1 & \text{(for each monochromatic rectangle } r \in M), \\ & z_q \leq 0 & \text{(for each clique } q \text{ in } G). \end{array} \tag{5.2}$$

Intuitively, this formulation can be interpreted as follows. Each cell c is assigned the weight w_c. The summation of weights over all cells in a monochromatic rectangle is bounded by one. This bound is decreased by the value z_q if it is contained in a clique q.

Using an example of the 3-bit majority function considered in Section 4.1, we explain how to prove a formula-size lower bound by using the problem (5.2). The original LP bound (2.7) provides a lower bound of at most 4.5, which can be easily checked by giving an upper bound to the primal problem. As formula size must be an integer, this already implies a lower bound of 5 on the formula size of the 3-bit majority function. By using clique constraints, we can directly show the lower bound of 5.

We call a cell *singleton* if it contains only one index. A notion of singleton cell plays an important role in the proof given below because the indices of singleton cells in a monochromatic rectangle must be the same.

Proposition 5.1 ([44])**.**

$$L(\mathbf{MAJ}_3) = L_m(\mathbf{MAJ}_3) = 5.$$

Proof. We have a monotone formula $(x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$ for $\mathbf{MAJ}_3$, which implies $L_m(\mathbf{MAJ}_3) \leq 5$. From the definition, $L(\mathbf{MAJ}_3) \leq L_m(\mathbf{MAJ}_3)$. To show $L(\mathbf{MAJ}_3) = L_m(\mathbf{MAJ}_3) = 5$, it suffices to prove the inequality $L(\mathbf{MAJ}_3) \geq 5$. For this, we construct a feasible solution of the dual problem (5.2) which has the objective function value equal to 5.

The communication matrix of the 3-bit majority function is shown in Figure 14. We consider a set q^* of all monochromatic rectangles of the communication matrix which contain at least two singleton cells. Note that if a rectangle contains two cells (α_1, β_1) and (α_2, β_2), it also contains both (α_1, β_2) and (α_2, β_1). Hence, a monochromatic rectangle is in q^* if and only if it contains one of the three 2×2 monochromatic rectangles in Figure 15. It is clear from this observation that every pair of monochromatic rectangles in q^* intersects, and therefore q^* corresponds to a clique.

	100	010	001	000
110	2	1	1,2,3	1,2
101	3	1,2,3	1	1,3
011	1,2,3	3	2	2,3
111	2,3	1,3	1,2	1,2,3

Fig. 14. The communication matrix of $\mathbf{MAJ_3}$.

	100	010	001
110		1	1,2,3
101		1,2,3	1
011			

	100	010	001
110	2		1,2,3
101			
011	1,2,3		2

	100	010	001
110			
101	3	1,2,3	
011	1,2,3	3	

Fig. 15. Three 2×2 monochromatic rectangles containing two singleton cells.

As a solution of the dual problem (5.2), we assign weights $w_c = 1$ for all singleton cells in the submatrix and $w_c = 0$ for other cells. There are six singleton cells in the submatrix and hence the total weight $\sum_c w_c$ is 6. We assign $z_{q^*} = -1$ for the clique q^*, and $z_q = 0$ for all other cliques $q \neq q^*$. Then, the objective function of the dual problem (5.2) is equal to $6 - 1 = 5$.

Now, we show that w and z defined above satisfy all constraints of the dual problem (5.2). It is easy to see from Figure 14 that every monochromatic rectangle contains at most two singleton cells. For a monochromatic rectangle containing at most one singleton cell, the constraints are clearly satisfied because the summation of weights in the monochromatic rectangle (i.e., $\sum_{c \in r} w_c$) is 0 or 1, while $\sum_q z_q \leq 0$ holds since $z_q \leq 0$. For a monochromatic rectangle which contains exactly two singleton cells, we have $\sum_{c \in r} w_c = 2$, while the monochromatic rectangle is contained in the clique q^* and therefore $\sum_q z_q \leq -1$. Thus, all the constraints are satisfied. □

The best formula-size upper and lower bounds of majority functions are $O(n^{4.57})$ [33] and $\lceil n/2 \rceil^2$ ($= (l+1)^2$ when $n = 2l+1$), respectively, which can be proven by the classical result of Khrapchenko [20]. The best monotone upper and lower bounds of majority functions are $O(n^{5.3})$ [46] and $\lfloor n/4 \rfloor n(1 + \log \frac{n}{n-2})$ [35], respectively. Using the idea described above, we can slightly improve the non-monotone formula-size lower bound as follows, while no previously known techniques has been able to improve it since 1971.

Theorem 5.2 ([44]). *It holds that* $L(\mathbf{MAJ}_{2l+1}) \geq (l+1)^2 + 1$.

Although our improvements of lower bounds seem to be small, it breaks a stiff barrier (known as the certificate complexity barrier [24]) of previously known proof techniques.

Here, we illustrate a key idea for the proof of Theorem 5.2. The complete proof is available in the original paper [44]. The idea is based on the decomposition theory of Boolean functions [5, 12, 34]. Namely, we can decompose the majority function with an odd number of input bits into compositions of 3-bit majority functions. A similar idea also works for communication matrices. We consider 3×3 submatrices obtained as follows. From $2l+1$ input bits, we fix $2l-2$ arbitrary bits and assume that they have the same number of 0's and 1's. Then, we consider the remaining three bits. If the $2l+1$ input bits compose a minterm (i.e., the input bits give a "minimal" vector among all vectors satisfying the majority function), the three bits are either 110, 101 or 011. If the $2l+1$ input bits compose a maxterm (i.e., the input bits give a "maximal" vector among all vectors which do not satisfy the majority function), the three bits are either 100, 010 or 001. Thus, we have a 3×3 submatrix, which has the same structure as the communication matrix of the 3-bit majority function as shown in Figure 16.

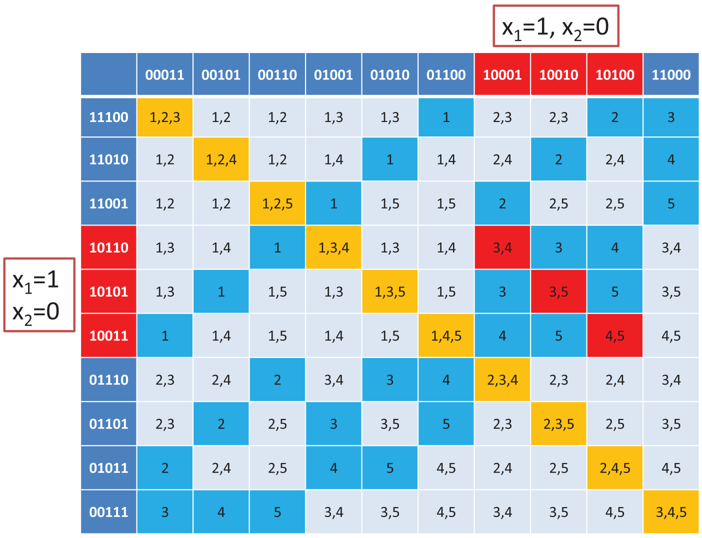
5.2 Generator Subadditive Functions and Clique Constraints

In the previous section, we consider clique constraints for the LP bound. While clique constraint seems to be irrelevant to the notion of subadditivity at the first glance, they are closely related, as explained below. To relate the two notions, we review a subadditive approach studied by Klabjan [21].

The survey paper by Güzelsoy and Ralphs [8] shows an excellent overview of several methods to construct subadditive functions, which are classified as follows:

(1) Obtained from known families of relaxations.

(2) Obtained as a by-product of a primal solution algorithm, such as branch-and-bound.


x_1=1, x_2=0
x_1=1
x_2=0

	00011	00101	00110	01001	01010	01100	10001	10010	10100	11000
11100	1,2,3	1,2	1,2	1,3	1,3	1	2,3	2,3	2	3
11010	1,2	1,2,4	1,2	1,4	1	1,4	2,4	2	2,4	4
11001	1,2	1,2	1,2,5	1	1,5	1,5	2	2,5	2,5	5
10110	1,3	1,4	1	1,3,4	1,3	1,4	3,4	3	4	3,4
10101	1,3	1	1,5	1,3	1,3,5	1,5	3	3,5	5	3,5
10011	1	1,4	1,5	1,4	1,5	1,4,5	4	5	4,5	4,5
01110	2,3	2,4	2	3,4	3	4	2,3,4	2,3	2,4	3,4
01101	2,3	2	2,5	3	3,5	5	2,3	2,3,5	2,5	3,5
01011	2	2,4	2,5	4	5	4,5	2,4	2,5	2,4,5	4,5
00111	3	4	5	3,4	3,5	4,5	3,4	3,5	4,5	3,4,5

Fig. 16. The communication matrix for the 5-bit majority function.

(3) Constructed explicitly in closed form using a finite procedure.

Among them, we here pick up a method called “generator subadditive function” by Klabjan [21].

Definition 5.3 (Generator Subadditive Function [21]). Given a vector $\alpha \in \Re^m$, *a generator subadditive functions* $F_\alpha : \Re^m_+ \to \Re$ is defined as

$$F_\alpha(d) = \alpha^T d - \max\left\{\sum_{i \in E}(\alpha^T a_i - c_i)x_i \mid A^E x \le d, x \in Z_+^{|E|}\right\}, \tag{5.3}$$

where $E = \{i \in [n] \mid \alpha^T a_i > c_i\}$ is called the generator set and A^E is the submatrix of A consisting of the columns with indices in E.

The next theorem shows that any generator subadditive function F_α is a feasible solution of the IP dual problem (2.4), and therefore the value $F_\alpha(b)$ gives a lower bound for the optimal value of the IP problem (2.1).

Theorem 5.4 ([21]). *For any α, the generator subadditive function F_α satisfies the following properties:*

(1) *F_α is subadditive and $F_\alpha(0) = 0$.*

(2) *$F_\alpha(a_i) \le \alpha^T a_i \le c_i$ for all $i \in [n] \setminus E$.*

(3) *$F_\alpha(a_i) \le c_i$ for all $i \in E$.*

Moreover, Klabjan [21] developed a methodology to compute an “optimal” generator subadditive function in the sense that it is an optimal solution of the IP dual problem (2.4) and the value $F_\alpha(b)$ matches the optimal value of the (2.1).

In the following, we explain the connection between generator subadditive function and the primal-dual pair of LPs (5.1) and (5.2), where we again use an example of the 3-bit majority function as in the previous subsection. In particular, we show that the formula-size lower bound of 5 for the 3-bit majority function can be also obtained by using some generator subadditive function with an appropriate α, in a similar way as we did in the proof of Proposition 5.1.

Recall that in the LP problem (5.1), each row of the matrix A and each component of the parameter vector α correspond to a cell of the communication matrix, while each column a_i of the matrix A corresponds to a monochromatic rectangle of the communication matrix. Also, note that $c_i = 1$ for all i. We consider a generator subadditive function F_α with the parameter vector α given as in the assignment of w_c in the proof of Proposition 5.1, i.e., $\alpha_c = 1$ for all the singleton cells and $\alpha_c = 0$ for other cells. Then, for each column vector a_i, the value $\alpha^T a_i$ is equal to the number of singleton cells in the monochromatic rectangle corresponding to a_i. Since $\alpha^T a_i > 1$ for each $i \in E$, the generator set E is given as the set of monochromatic rectangles containing two singleton cells, and therefore it corresponds to a clique q^* considered in the proof of Proposition 5.1.

In the following, we restrict our attention to the case $d = \vec{1} = (1, 1, \ldots, 1)$ since the function F_α is a feasible solution of the IP dual problem (2.4) and therefore the value $F_\alpha(\vec{1})$ gives a formula-size lower bound. If $d = \vec{1}$, then the inequality $A^E x \le d$ in the definition of generator subadditive function (5.3) is equivalent to the clique inequality for q^*. Since we have $\alpha^T a_i - c_i = 2 - 1 = 1$ for all $i \in E$, the maximum in (5.3) is achieved by setting any one variable x_i with $i \in E$ to 1 and any other variables to 0. Hence, we have

$$-\max\left\{\sum_{i \in E}(\alpha^T a_i - c_i)x_i \mid A^E x \le \vec{1}, x \in Z_+^{|E|}\right\} = -1.$$

This value corresponds to the assignment of $z_{q^*} = -1$ in Section 5.1. We have $\alpha \cdot \vec{1} = \sum_c w_c$, which is equal to the number of singleton cells, i.e., $\alpha \cdot \vec{1} = 6$. Hence, it holds that

$$F_\alpha(\vec{1}) = \alpha^T\vec{1} - \max\left\{\sum_{i\in E}(\alpha^T a_i - c_i)x_i \mid A^E x \leq \vec{1}, x \in Z_+^{|E|}\right\} = 6 - 1 = 5.$$

That is, we obtain the lower bound of 5 by using a generator subadditive function.

In the discussion above, the inequality $A^E x \leq \vec{1}$ is regarded as a clique constraint. This kind of reinterpretation of clique constraint would be helpful to generalize ideas and to borrow some techniques developed in the theory of integer programming duality and subadditive functions. For example, there are some generalized notions of clique constraint such as orthonormal constraint [7] and rank constraint.

To explain the notion of rank constraint, we consider the graph G defined in Section 5.1. Let H be a (node-)induced subgraph of G, and denote by $\alpha(H)$ the maximum size of an independent set in H. Recall that a node set is an independent set of a graph if any pair of vertices in the set is not connected by an edge. Then, the *rank constraint* is given as

$$\sum_{r\in H} x_r \leq \alpha(H),$$

where $r \in H$ means the node corresponding to a rectangle r is contained in the induced subgraph H. If H is a clique, then we have $\alpha(H) = 1$, and therefore a rank constraint for such H is the same as a clique constraint.

Validity of a rank constraint for the integer programming problem (2.6) can be shown as follows. Recall that each node in the graph G represents a rectangle and two vertices in G is connected by an edge if and only if the corresponding rectangles intersect. Since the rectangles in a partition of a communication matrix are mutually disjoint, the set of such rectangles correspond to an independent set of the graph G. Hence, any induced subgraph H of G contains such rectangles at most $\alpha(H)$.

Associated with the LP bound with rank constraints, we consider its LP dual formulated as follows:

$$\begin{array}{lll}
\max & \sum_{c} w_c + \sum_{H} \alpha(H) z_H & \\
s.t. & \sum_{c\in r} w_c + \sum_{H\ni r} z_H \leq 1 & \text{(for each monochromatic rectangle } r\text{)}, \\
& z_H \leq 0 & \text{(for each subgraph } H\text{)}.
\end{array} \tag{5.4}$$

Similarly to the case of clique constraints, we can show a formula-size lower bound by giving a feasible solution for this dual problem.

6. Conclusions

In this paper, we have reviewed some topics in which subadditivity plays an important role. In particular, we have explained some connection between the theory of integer programming duality and linear programming based technique proving formula-size lower bounds. These topics were explained from the viewpoints of the difference between subadditivity and additivity.

Some existing framework to derive formula-size lower bounds gets stuck because of the difficulty to cope with subadditivity. Based on the **P** $\neq$ **NP** conjecture, subadditivity is hard to handle in general. It has also revealed, however, that there exist some special kinds of subadditivity which can be handled in practice. They can be a new frontier in both research areas of algorithms and complexity theory.

We believe that intrinsic property of computation can be captured as the difference between additivity and subadditivity as presented in this paper. While its complete understanding seems to be difficult due to the **P** $\neq$ **NP** conjecture, exploring some tractable subadditive functions would be hopeful ways to seek new frontiers in both mathematical programming and computational complexity.

Acknowledgment

We would like to thank anonymous reviewers for many valuable comments improving the presentation of this paper.

REFERENCES

[1] A. Ambainis. Polynomial degree vs. quantum query complexity. *Journal of Computer and System Sciences*, 72(2):220–238, 2006.

[2] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[3] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I, 2nd edition*. Springer, 1994.

[4] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[5] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM SIGACT News*, 34(4):38–52, 2003.

[6] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[7] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

[8] M. Güzelsoy and T. K. Ralphs. Integer Programming Duality. In J. Cochran, editor, *Encyclopedia of Operations Research and Management Science*. Wiley, 2011.

[9] J. Håstad. The shrinkage exponent of De Morgan formulas is 2. *SIAM Journal on Computing*, 27(1):48–64, 1998.

[10] P. Høyer, T. Lee, and R. Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 526–535, 2007.

[11] P. Hrubeš, S. Jukna, A. Kulikov, and P. Pudlák. On convex complexity measures. *Theoretical Computer Science*, 411:1842–1854, 2010.

[12] T. Ibaraki and T. Kameda. A theory of coteries: Mutual exclusion in distributed systems. *IEEE Transactions on Parallel and Distributed Computing*, PDS-4(7):779–794, July 1993.

[13] K. Iwama and H. Morizumi. An explicit lower bound of $5n - o(n)$ for Boolean circuits. In *27th International Symposium of Mathematical Foundations of Computer Science (MFCS 2002)*, volume 2420 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2002.

[14] E. L. Johnson. Cyclic groups, cutting planes and shortest paths. In T. Hu and S. Robinson, editors, *Mathematical Programming*, pages 185–211. Academic Press, 1973.

[15] E. L. Johnson. *Integer Programming: Facets, Subadditivity, and Duality for Group and Semi-group problems*. SIAM, 1980.

[16] S. Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.

[17] N. Kamiyama. A note on the quasi-additive bound for Boolean functions. *Journal of Math-for-Industry*, 4:119–122, 2012.

[18] M. Karchmer, E. Kushilevitz, and N. Nisan. Fractional covers and communication complexity. *SIAM Journal on Discrete Mathematics*, 8(1):76–92, 1995.

[19] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990.

[20] V. M. Khrapchenko. Complexity of the realization of a linear function in the case of π-circuits. *Mathematical Notes*, 9:21–23, 1971.

[21] D. Klabjan. Subadditive approaches in integer programming. *European Journal of Operational Research*, 183(2):525–545, 2007.

[22] E. Koutsoupias. Improvements on Khrapchenko's theorem. *Theoretical Computer Science*, 116(2):399–403, 1993.

[23] O. Lachish and R. Raz. Explicit lower bound of $4.5n - o(n)$ for boolean circuits. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, (STOC 2001)*, pages 399–408. ACM Press, 2001.

[24] S. Laplante, T. Lee, and M. Szegedy. The quantum adversary method and classical formula size lower bounds. *Computational Complexity*, 15(2):163–196, 2006.

[25] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[26] J. B. Lasserre. Integer programming duality and superadditive functions. *Contemporary Mathematics*, 374:139–150, 2005.

[27] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28(3):470–496, 2003.

[28] T. Lee. *Kolmogorov Complexity and Formula Size Lower Bounds*. PhD thesis, University of Amsterdam, January 2006.

[29] T. Lee. A new rank technique for formula size lower bounds. In *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, volume 4393 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2007.

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

[31] D. A. Morán R., S. S. Dey, and J. P. Vielma. A strong dual for conic mixed-integer programs. *SIAM Journal on Optimization*, 22(3):1136–1150, 2012.

[32] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[33] M. S. Paterson, N. Pippenger, and U. Zwick. Optimal carry save networks. In *Boolean function complexity*, volume 169 of *London Mathematical Society Lecture Note Series*, pages 174–201. Cambridge University Press, 1992.

[34] E. L. Post. *The two-valued iterative systems of mathematical logic*, volume 5 of *Annals Mathematical Studies*. Princeton University Press, 1941.

[35] J. Radhakrishnan. Better lower bounds for monotone threshold formulas. *Journal of Computer and System Sciences*, 54(2):221–226, 1997.

[36] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[37] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[38] C. Shannon. The synthesis of two–terminal switching circuits. *Bell System Technical Journal*, 28:59–98, 1949.

[39] H. D. Sherali and W. P. Adams. A hierarchy of relaxation between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3:411–430, 1990.

[40] H. D. Sherali and W. P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, 52(1):83–106, 1994.

[41] H. D. Sherali and Y. Lee. Tighter representations for set partitioning problems. *Discrete Applied Mathematics*, 68(1-2):153–167, 1996.

[42] G. Simonyi. Graph entropy: A survey. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 20:399–

441, 1995.

[43] P. Spira. On time-hardware complexity tradeoffs for boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences*, pages 525–527, 1971.

[44] K. Ueno. A stronger LP bound for formula size lower bounds via clique constraints. *Theoretical Computer Science*, 434(1):87–97, 2012.

[45] K. Ueno. Breaking the rectangle bound barrier against formula size lower bounds. *International Journal of Foundations of Computer Science*, 24(8): 1339–1354, 2013.

[46] L. G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984.

[47] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.