ALGORITHMIC AND COMPLEXITY RESULTS FOR BOOLEAN AND PSEUDO-BOOLEAN FUNCTIONS

BY ARITANAN G. GRUBER

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

Written under the direction of

Endre Boros

and approved by

New Brunswick, New Jersey

January, 2015

ABSTRACT OF THE DISSERTATION

Algorithmic and Complexity Results for Boolean and Pseudo-Boolean Functions

by Aritanan G. Gruber

Dissertation Director: Endre Boros

This dissertation presents our contributions to two problems.

In the first problem, we study the hardness of approximation of clause minimum and literal minimum representations of pure Horn functions in n Boolean variables. We show that unless $\mathsf{P} = \mathsf{NP}$, it is not possible to approximate in polynomial time the minimum number of clauses and the minimum number of literals of pure Horn CNF representations to within a factor of $2^{\log^{1-o(1)} n}$. This is the case even when the inputs are restricted to pure Horn 3-CNFs with $O(n^{1+\varepsilon})$ clauses, for some small positive constant ε. Furthermore, we show that even allowing sub-exponential time computation, it is still not possible to obtain constant factor approximations for such problems unless the Exponential Time Hypothesis is false.

In the second problem, we study quadratizations of pseudo-Boolean functions, that is, transformations that given a pseudo-Boolean function $f(x)$ in n variables, produce a quadratic pseudo-Boolean function $g(x, y)$ in $n + m$ variables such that $f(x) = \min_{y\in\{0,1\}^m} g(x, y)$ for all $x \in \{0, 1\}^n$. We present some new termwise procedures, leading to improved experimental results, and then take a global perspective and start a systematic investigation of some structural properties of the class of all quadratizations of a given function.

We show that all pseudo-Boolean functions in n variables can be quadratized and y-linear quadratized (no quadratic products involving solely auxiliary variables) with at most $O(2^{n/2})$ and $O\big(\frac{2^n}{n}\log n\big)$ auxiliary variables, respectively, and that almost all those functions require $\Omega(2^{n/2})$ and $\Omega(2^n/n)$ auxiliary variables in any quadratization and any y-linear quadratization, respectively. We obtain the bounds $O(n^{d/2})$ and $\Omega(n^{d/2})$ for quadratizations of degree-d pseudo-Boolean functions, and bounds of $n-2$ and $\Omega(n/\log n)$ for y-linear quadratizations (and $\Omega(\sqrt{n})$ for quadratizations) of symmetric pseudo-Boolean functions. All our upper bounds are constructive, so they provide new (y-linear) quadratization algorithms.

We then finish with a characterization of the set of all quadratizations of negative monomials with one auxiliary variable, a result that was surprisingly difficult to obtain, and whose proof at the moment is rather long and intricate.

Acknowledgements

First and foremost I would like to express my deepest gratitude to my adviser, Professor Endre Boros. His guidance was always sharp, his comments were always insightful, and his knowledge and ability to keep learning and producing new results were always inspiring. It is no surprise that his research style can be felt along these pages and it is just not possible for me to overestimate the amount of knowledge that I have learned from him. Furthermore, he was always understanding and kept believing in me during my recovery period (see below), a time when I myself had serious doubts. More than an exemplary leader and researcher, he is an outstanding human being.

I also would like to express my gratitude to the remaining members of my committee, Professors Adi Ben-Israel, Ondrej Čepek, Vladimir Gurvich, and Alexander Kogan, to my co-authors, Professors Martin Anthony, Yves Crama, Ramin Zabih, and Ph.D. candidate Alex Fix, as I learned plenty in our collaborations, and to Professor Michael Saks for inviting me to participate in the Theory of Computing Reading Group that he was running — once again, it is not possible to overestimate the amount of knowledge I have acquired on those meetings.

A tragic bicycle accident sustained by the end of my second year into the Ph.D. program almost put an end to my academic career, and I am deeply indebted to Elizabeth Clark, Irina Efremova, Roopinder Grewal, Nicky Isaacson, and Kevin Nini for all the attention and excellent medical care dispensed to me during a slow, lengthy, and hard recovery.

During such period and also in any other moment, I was fortunate enough to count on the humor, the good heart, the dedication, and the loyalty of a strong collection of friends: Mara and Tibérius Bonates (OR), Ana Paula Centeno (CS) and Fábio Oliveira (CS), Janaína, Lucas, and Carlos Oliveira, György Mátyásfalvi (OR), Lauren

Matricardi and Matthew Oster (OR), Daniela Vianna (CS), Daniel Vianna Toso, and Rodrigo Toso (CS). Each one provided me invaluable support and friendship whenever needed, becoming my family away from home. You guys represent! And I have learned a lot about life and about a myriad of other things from each and every one of you.

I also enjoyed many good scientific, technical, refreshing, and relaxing (small) talks with many RUTCOR, CS, and Math colleagues and friends, like Tsvetan Asamov, Marta Cavaleiro, Gianlucca Gazzola, Javier Herrero, Ai Kagawa, Jinwook Lee, Naomi Miller, Peter Mursic, Olga Myndyuk, Mariya Naumova, Anh Ninh, David Pápp, Minh Pham, Mohammad Ranjbar, Anupama Reddy, Svetlana Soloveva, Ricardo Soto, Tomáš Vyskočil, Emre Yamangil, Wang Yao; Luke Friedman, Nikos Leonardos, John McCabe, André Madeira, Luiz Ramos, Fengming Wang; Justin Gilmer, and Simão Herdade. And also spent some good times with the always helping RUTCOR staff: Lynn Agre, Katie D'Agosta, Terry Hart, Clare Smietana, and Chong.

Many brazilian friends and former professors of mine also offered some invaluable support along the way. Among the friends, I would like to mention Pam Albuquerque (in memoriam), Solange Formiga Albuquerque, Cassio Polpo de Campos, Carlos Cardonha, Eduardo Teixeira Dias, Marcel Kenji de Carli Silva, Sérgio Gabriel "Batata" Tavares, and Maya Yoshiko Yamamoto. You guys also represent! A honorable mention goes to my "brother" Mario Leston Rey, who was always available when I wanted to chat (about virtually any topic) and who has being a source of guidance and inspiration through the years. Among the Professors: Paulo Feofiloff, Cristina Gomes Fernandes, Carlos Eduardo "Carlinhos" Ferreira, Dilma Menezes da Silva, and Yoshiko Wakabayashi.

Last but not least, I would like to express all my gratitude and love to my mom, Maria Aparecida Borges Garcia Gruber, to my dad, Carlos Roberto Gruber (in memoriam), to my sister, Yanê Borges Garcia Gruber, and to my longtime girlfriend, Kátia Sobrinho Favorito. You guys sacrificed a lot while always offering me love, comfort, and a pletora of other good things in the long years this dissertation was in the making. I love you guys with each and every fiber of my being! I hope I can somehow make up for at least a fraction of all the time I was busy, or absent, or lost and found again. Thank you so much for everything.

During the coursework, recovery, and research stages I was supported by a fellowship jointly offered by the brazilian Coordination for the Improvement of Higher Education Personnel (CAPES) and the Fulbright Commission, process number BEX-2387050/15061676, and as a graduate research assistant on the National Science Foundation grants NSF-IIS-0803444, Pseudo-Boolean Methods in Computer Vision, and NSF-IIS-1161476, Graph Cut Algorithms for Domain-specific Higher-order Priors.

Dedication

To my family, for their endless support and encouragement.

Table of Contents

1 Introduction

"Go to the roots of calculations! Group the operations. Classify them according to their complexities rather than their appearances! This, I believe, is the mission of future mathematicians." – Évariste Galois, 1832

Boolean and pseudo-Boolean functions, i.e., mappings of finite binary vectors (strings) of $\{0,1\}^n$ to binary and real values, respectively, are two of the most fundamental mathematical objects and occur widely in fields as diverse as artificial intelligence [3, 99], combinatorics [95], computational complexity [8, 96, 122], operations research [79, 50, 51], social choice theory [122], among many others. This omnipresence may be partially explained by the fact that they are extremely versatile in modeling and expressing both structured and unstructured discrete properties, as those can be seen as subsets (also called classes) of the set of Boolean or pseudo-Boolean functions whose elements satisfy some logical relations.

Given a Boolean or pseudo-Boolean function f in n variables, on a first level, one is typically interested in answering questions as whether f has some specific property (i.e., belongs to a certain class), or what is the value of $f(x)$ for some $x \in \{0,1\}^n$, or what is the smallest/largest value attained by f over $\{0,1\}^n$. On a second level, one gets concerned about the necessary amount of computational resources to provide those answers, where a resource may mean sequential or parallel time, space, communication between two or more parties, random tosses of a coin, etc. The computation of any of these quantities depends on the problem whose solution is being sought and on the form of accessing the function f, which can be implicit through the use of an *oracle* (that is, an algorithm accessed in a black-box fashion), or explicit through the availability of some table with the function's values, or an algebraic expression or circuit succinctly encoding f — and these last two options open a wide array of questions about the representations of f themselves. Most times, that precise computation is difficult, and one settles for lower and upper bounds on the amount of those resources. Naturally, the smaller the gap between such bounds, the better.

In this dissertation, we contribute in finding and improving lower and upper bounds on time and other more specific resources for two problems: one involving a class of Boolean functions, called Horn functions, and the other involving pseudo-Boolean functions. In both problems, we assume that the functions are specified through some explicit (and problem dependent) algebraic expression.

In the first, the expressions are called *Conjunctive Normal Forms* (CNFs) — logical conjunctions of logical disjunctions of positive or negated Boolean variables — and are not unique, in the sense that a Horn function may admit many different CNF representations. Finding the shortest possible CNF representation of a Horn function is known to be an NP-hard problem (that is, a given candidate solution can be tested to be correct in time polynomially bounded in the size of the problem's input, but finding such a candidate might not be an easy task as it may require exponential time in the input's size — settling the last statement either in a positive or a negative way is the famous P versus NP problem (see Arora and Barak [8])). In fact, even finding approximate short representations is known to be NP-hard beyond a certain approximation factor. We improve that approximation factor under milder complexity-theoretic hypothesis, and also address some more and some less stringent variants: restricting the input to CNFs with at most three positive or negative variables in each disjunction, and allowing sub-exponential time to perform the necessary computations. The net message is that the problem is way harder than what was known and initially thought: no constant factor approximation is possible, even having access to a sub-exponential amount of computational time and dealing with very sparse CNFs, unless a widely believed complexity-theoretic hypothesis, the Exponential Time Hypothesis, turns out to be false. A more informative picture is given below, in Subsection 1.1.

The second problem is related to pseudo-Boolean optimization, or more specifically to minimization of pseudo-Boolean functions represented by multilinear polynomials. It is known that this representation is unique and it is also known that the optimization problem is NP-hard. In recent years, an influx of new activity has been seen, most of it originating from researchers inside the computer vision community. They realized that the results of Hammer, Hansen, and Simeone [78] (*roof duality*, *persistencies*) and of Boros and Hammer [28], and Boros, Hammer, Sun, and Tavares [30] (*max-flow/min-cut* reduction) were specially tailored to handle the very large instances of unconstrained, quadratic binary optimization problems originating from applications such as image restoration, stereo reconstructions, and picture segmentation. As they moved towards more complex models, resulting in higher-degree polynomials, they started experimenting in an ad-hoc fashion with variations of an idea proposed by Rosenberg [127], namely, to introduce auxiliary variables and represent the desired function as a quadratic multilinear polynomial in this higher-dimensional space, so that both forms attain the same values when the minimization targets those auxiliary variables. The literature then saw the addition of a collection of specific "quadratization" procedures, mostly trying to make the resulting quadratized function as close to submodular as possible, for submodular pseudo-Boolean functions are solved exactly by the flow-based algorithm mentioned above. Also, many experimental results, typically testing the quadratic optimization models produced by various quadratization procedures, were published. In spite of all that activity, the understanding of quadratizations and of their structural properties remained extremely limited.

We make some progress towards determining the strengths and weaknesses of quadratizations by exhibiting lower and upper bounds on the number of auxiliary variables necessary to quadratize general and symmetric pseudo-Boolean functions (the latter being dependent only on the number of ones in the input). Our upper bounds are constructive, that is, they generate new quadratization procedures, but they take a global standpoint: they are valid among all possible quadratizations of a given function, and not related to some specific technique. Nonetheless, we also propose some new specific quadratization schemes that proved themselves to be very efficient when experimentally evaluated in some computer vision applications. Our lower bounds are in some cases almost, in some cases essentially tight, and are the first ones ever introduced in the quadratization realm. Combining our lower and upper bounds, it is possible to conclude that short quadratizations, with respect to the number of auxiliary variables, are rare objects in general, but that they can also be easily found in some smaller niches. Similarly, a more informative picture is given below, in Subsection 1.2.

1.1 Hardness of Approximation of Pure Horn CNF Minimization

A Boolean function in n binary variables is a mapping from $\{0,1\}^n$ to $\{0,1\}$. Traditionally, the elements 0 and 1 are associated with the logical concepts of falseness and trueness, as Boolean functions have a long history in modeling logical propositions; see Crama and Hammer [51] for a brief picture, or a book in mathematical logic as Mendelson [114] for a deeper treatment, or even a book in computational complexity theory as Papadimitriou [123] for a coverage with a different viewpoint.

It is not hard to see that for each natural value n, there are 2^{2^n} different Boolean functions in n variables, and that each of those Boolean functions can be completely specified through its true table, i.e., the list of values attained by the function in each of its 2^n possible inputs. Despite being useful in some cases and having some applications, the list quickly becomes too long as n grows and more succinct forms of representation are preferred. One such form, widely used, is the *Conjunctive Normal Form* (CNF): logical conjunctions of logical disjunctions of literals, where the latter is simply a binary variable or its negation. The *satisfiability problem* (SAT) of Boolean functions, the first "natural" problem to be proven NP-complete [45, 73] and most likely, the most famous one in this complexity class, can be stated as: given a CNF representing a Boolean function f, decide whether there is a (binary) valuation to its variables such that the formula evaluates to one, that is, it is satisfiable in the logical sense. In spite of its tremendous importance, its complexity status propelled researchers to investigate simpler classes of Boolean functions in which knowledge representation was still possible, at least in a partial way, and logical inference could be performed in polynomial time in the representation size. One such class is the set of Horn functions.

Horn functions constitute a rich and important class of Boolean functions and have many applications in artificial intelligence, combinatorics, computer science, and operations research. Furthermore, they possess some nice structural and algorithmic properties. An example of this claim resides in the fact that the satisfiability for Horn functions can be solved in linear time in the number of variables plus the length of the Horn CNF formula being considered (Dowling and Gallier [56], Itai and Makowsky [93], and Minoux [117]), where length in this context means the number of literal occurrences in the formula (i.e., multiplicities are taken into account).

As mentioned before, CNF representations are usually not unique and that claim is also applicable in the Horn domain. The problem of finding short Horn CNF representations of Horn functions specified through Horn CNFs has received some considerable attention in the literature since it has an intrinsic appeal stemming from both theoretical and practical standpoints. The same can be said about some special cases, including Horn 3-CNFs, i.e., the ones in which each clause has at most three literals.

Two of the common measures considered are the number of clauses and the number of literal occurrences (henceforth, number of literals). The NP-hardness of minimizing the number of clauses in Horn CNFs was first proved in a slightly different context of directed hypergraphs by Ausiello, D'Atri, and Saccà [13]. Their reduction however, produces clauses in which all variables appear (i.e., clauses of very high degree). The NP-hardness was later shown by Boros, Čepek, and Kučera [23] to also hold for the case of pure Horn 3-CNFs. Regarding the minimization of the number of literals, it was shown to be NP-hard for Horn CNFs by Maier [113] (strictly speaking, he used a slightly different measure, but his proof can be easily modified to work also for the number of literals). A simpler proof, this time really considering the number of literals, was later found by Hammer and Kogan [82]. Both Maier's and Hammer and Kogan's reductions introduce high degree clauses (equal to the number of variables). Čepek [39] improved this result to Horn 7-CNFs and Čepek and Kučera [41] improved it to Horn 5-CNFs. Boros, Čepek, and Kučera [23] later showed that the NP-hardnes status still applies to Horn 3-CNFs.

It is worth to mention that both clause and literal minimization of general 2-CNFs can be accomplished in polynomial time as shown by Hammer and Kogan [83]. This, of course, includes Horn 2-CNFs, implying that the NP-hardness of clause and literal minimization for Horn CNFs is completely solved. It is of independent interest to mention the existence of other subclasses of Horn functions where CNF minimization can be accomplished in polynomial time: the subclasses of bi-dual, acyclic, and quasi-acyclic Horn functions; see Crama and Hammer [51] for details.

The attention then shifted to trying to approximate those values. Hammer and Kogan [82] showed that for a pure Horn function in n variables, it is possible to approximate the minimum number of clauses and the minimum number of literals of a pure Horn CNF formula representing it to within factors of $n-1$ and $\binom{n}{2}$, respectively. For many years, this was the only result regarding approximations. Recently, a super-logarithmic hardness of approximation factor was shown by Bhattacharya, DasGupta, Mubayi, and Turán [18] for the case of minimizing the number of clauses for general Horn CNFs. We provide more details on this shortly.

Another measure for minimum representations of Horn functions concerns minimizing the number of source sides, grouping together all clauses with the same source set. Maier [113] and Ausiello, D'Atri, and Saccá [13] showed that such a minimization can be accomplished in polynomial time (see also Crama and Hammer [51]). While this measure is sometimes used in practice, providing reasonably good results, we consider it an important intelectual quest, following Bhattacharya, DasGupta, Mubayi, and Turán [18], to try to precisely understand the hardness of the other two measures.

Our contribution focus on the hardness of approximating short pure Horn CNF/3-CNF representations of pure Horn functions, where pure means that each clause has exactly one positive literal (definitions are provided in Section 2.1). More specifically, we study the hardness of approximating

1. the minimum number of clauses of pure Horn functions specified through pure Horn CNFs,

2. the minimum number of clauses and the minimum number literals of pure Horn functions specified through pure Horn 3-CNFs,

when either polinomial or sub-exponential computational time is available.

In what follows, we present pointers to previous work on the subject, discuss our results in more details, and mention the main ideas behind them.

1.1.1 Previous Work

The first result on hardness of approximation of shortest pure Horn CNF representations was provided in Bhattacharya et al. [18]. Specifically, it was shown that unless $\mathsf{NP} \subseteq \mathsf{QP} = \mathsf{DTIME}(n^{\text{polylog}(n)})$, that is, unless every problem in NP can be solved in quasi-polynomial deterministic time in the size of the input's representation, the minimum number of clauses of a pure Horn function in n variables specified through a pure Horn CNF formula cannot be approximated in polynomial (depending on n) time to within a factor of $2^{\log^{1-\varepsilon} n}$, for any constant $\varepsilon > 0$ small enough.

This result is based on a gap-preserving reduction from a fairly well known network design problem introduced by Kortsarz [105], namely, MinRep and has two main components: a gadget that associates to every MinRep instance M a pure Horn CNF formula h such that the size of an optimal solution to M is related to the size of a clause minimum pure Horn CNF representation of h, and a gap amplification device that provides the referred gap. Despite being both necessary to accomplish the result, each component works in a rather independent way.

Inspired by the novelty of their result and by some characteristics of their reduction, we are able to further advance the understanding of hardness of approximation of pure Horn functions. We discuss how we strength their result below.

1.1.2 Our Results and Techniques

Our strengthening of the result of Bhattacharya et al. [18] can be summarized as follows: the hardness of approximation factor we present is stronger, the complexity theoretic assumption we use for polynomial time solvability is weaker, and the class of CNF formulae to which our hardness results apply is smaller. We are also able to derive further non-approximability results for sub-exponential time solvability using a different complexity theoretic hypothesis.

In more details, for a pure Horn function h in n variables, we show that unless

$\mathsf{P} = \mathsf{NP}$, the minimum number of clauses in a prime pure Horn CNF representation of h and the minimum number of clauses and literals in a prime pure Horn 3-CNF representation of h cannot be approximated in polynomial (depending on n) time to within factors of $2^{\log^{1-o(1)} n}$ even when the inputs are restricted to pure Horn CNFs and pure Horn 3-CNFs with $O(n^{1+\varepsilon})$ clauses, for some small constant $\varepsilon > 0$. It is worth mentioning that $o(1) \approx (\log\log n)^{-c}$ for some constant $c \in (0, 1/2)$ in this case. Notice that because these results are conditional on the $\mathsf{P} \neq \mathsf{NP}$ hypothesis, they also provide new and different proofs for the NP-hardness of clause minimization of pure Horn CNFs and of clause and literal minimization of pure Horn 3-CNFs.

After that, we show that unless the Exponential Time Hypothesis introduced by Impagliazzo and Paturi [89] is false, it is not possible to approximate the minimum number of clauses and the minimum number of literals of a prime pure Horn 3-CNF representation of h in time $\exp(n^{\delta})$, for some $\delta \in (0, 1)$, to within factors of $O(\log^{\beta} n)$ for some small constant $\beta > 0$. Such results hold even when the inputs are restricted to pure Horn 3-CNFs with $O(n^{1+\varepsilon})$ clauses, for some small constant $\varepsilon > 0$. Furthermore, we also obtain a hardness of approximation factor of $O(\log n)$ under slightly more stringent, but still sub-exponential time constraints. We would like to point out that our techniques leave open the problem of determining hardness of approximation factors when $exp(o(n))$ computational time is available. We conjecture, however, that constant factor approximations are still not possible in that case.

The main technical component of our work is a new gap-preserving reduction[1] from a graph theoretical problem called Label-Cover (see Section 2.2 for its definition) to the problem of determining the minimum number of clauses in a pure Horn CNF representation of a pure Horn function. We show that our reduction has two independent parts: a core piece that forms an exclusive component (Boros et al. [22]) of the function in question and therefore, can be minimized separately; and a gap amplification device which is used to obtain the hardness of approximation factor. It becomes clear that the same principle underlies Bhattacharya et al. [18]. The hardness of approximation

[1]Our reduction is actually approximation-preserving, but we do not need or rely on such characteristic. See Vazirani [146] or Williamson and Shmoys [149] for appropriate definitions.

factor comes (after calculations) from a result of Dinur and Safra [55] on the hardness of approximating certain Label-Cover instances.

We then introduce some local changes into our reduction that allow us to address the case in which the representation is restricted to be a pure Horn 3-CNF. Namely, we introduce extra variables in order to *cubify* clauses whose degree is larger than three, that is, to replace each of these clauses by a collection of degree two or degree three clauses that provide the same logical implications. This is done for two families of high degree clauses and for each family we use a different technique: a *linked-list* inspired transformation that is used on the classical reduction from SAT to 3-SAT instances (Garey and Johnson [73]), and a *complete binary tree* type transformation. The latter type is necessary to prevent certain shapes of prime implicates in minimum clause representations that would render the gap-amplification device innocuos. From this modified reduction, we are also able to derive in a straight-forward fashion a hardness result for determining the minimum number of literals of pure Horn functions represented by pure Horn 3-CNFs.

At this point we should mention that our reduction is somewhat more complicated than the one given in Bhattacharya et al. [18]. While we could adapt their reduction and obtain the same hardness of approximation factor we present in the case of pure Horn CNF representations (as based in our Lemma 2.15 we can argue that the Label-Cover and the MinRep problems are equivalent), the more involved gadget we use is paramount in extending the hardness result for the pure Horn 3-CNF case. Loosely speaking, the simple form of their reduction does not provide enough room to correctly shape those prime implicates in minimum pure Horn 3-CNF representations that we mentioned addressing in ours. In this way, the extra complications are justified.

Finally, using newer and slightly different results on the hardness of approximation of certain Label-Cover instances (Moshkovitz and Raz [119], Dinur and Harsha [54]) in conjunction with the Exponential Time Hypothesis [89], we are able to show (also after calculations) that it might not be possible to obtain constant factor approximations for the minimum number of clauses and literals in pure Horn 3-CNF representations.

The results above appeared in Boros and Gruber [25, 27] and will be presented in

Chapter 2.

1.1.3 Other Related Work

Boros, Čepek, Kogan, and Kučera [22] introduced the concept of *essential sets* of Boolean functions and defined a measure upon it that lower bounds the number of clauses in a clause minimum CNF representation of any Boolean function. As stated by Čepek, Kučera, and Savick [40], for any Boolean function f, thus including Horn functions, let $\text{ess}(f)$ be the largest set of assignments that falsify f, no two of which falsify a common implicate of f.

Čepek, Kučera, and Savick [40] then exhibited a Horn function f in n variables such that the multiplicative gap between $\text{ess}(f)$ and $\text{cnf-size}(f)$, the minimum number of clauses in a CNF representation of f, is

$$\frac{\text{cnf-size}(f)}{\text{ess}(f)} = \Theta(\log n).$$

Hellerstein and Kletenik [86] later showed a Horn function f in n variables such that the gap is $\Theta(\sqrt{n})$, and showed that no gap larger that $\Theta(n)$ is possible. If expressed in terms of $\text{cnf-size}(f)$, the gap becomes

$$\frac{\text{cnf-size}(f)}{ess(f)} \geq \text{cnf-size}(f)^{1/3},$$

and no gap larger than $\text{cnf-size}(f)$ is possible. Furthermore, based on Bhattacharya et al. [18] they showed that unless $\mathsf{NP} \subseteq \mathsf{co\text{-}NTIME}(n^{\text{polylog}(n)})$, i.e., unless every problem in NP can be solved in quasi-polynomial co-nondeterministic time in the size of the input's representation, there is a constant $0 < \varepsilon < 1$ such that there exists an infinite family of pure Horn functions in which f is a member, and such that

$$\frac{\text{cnf-size}(f)}{\text{ess}(f)} \geq 2^{\log^{1-\varepsilon} n},$$

where n is the number of variables of f.

It turns out that we can replace the result of Bhattacharya et al. [18] in Hellerstein and Kletenik [86]'s proof by our result on hardness of approximating the minimum number of clauses in pure Horn CNF representations and obtain that unless $\mathsf{NP} =$

co-NP, there exists a family of pure Horn functions such that

$$\frac{\text{cnf-size}(f)}{\text{ess}(f)} \geq 2^{\log^{1-o(1)} n},$$

where f is a member of such family and n its number of variables.

1.2 Quadratizations of Pseudo-Boolean Functions

A *pseudo-Boolean function* in n variables is a mapping from the set $\{0,1\}^n$ of n-dimensional binary vectors to the set $\mathbb{R}$ of real numbers. It is well known that every pseudo-Boolean function can be represented as a multilinear polynomial of its variables, and that such representation is unique (see Section 3.1.1 for details). This algebraic closed form allows interaction with the function in broader way when compared to a black-box/oracle regime of access, in which the sole operation available is a query of the function's value on its input vectors. Besides also providing queries, the multilinear representation can be used to discover further details of the function as its monomial structure, the spectra of its coefficients, and its degree (all concepts related to polynomial forms, but which are very helpful in understanding and classifying the function as a "simple" or "complex" one), and also to perform some transformations as lifting it to some higher dimensional space through the introduction of new, auxiliary variables, or as projecting it down via removal thereof. So, in a nutshell, multilinear representations can be seen as a white-box type of access and offer a collection of advantages. In this dissertation, we shall benefit from this white-box access fashion and therefore, we shall assume that each and every pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is specified through its unique multilinear polynomial:

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i, \tag{1.1}$$

where $[n] := \{1, 2, \ldots, n\}$, and the $c_S \in \mathbb{R}$ for all $S \subseteq [n]$. Sometimes, for convenience, we shall also consider some expressions involving the Boolean complement $\overline{x} = 1 - x$ of a binary variable x.

The problem of optimizing the multilinear polynomial of a pseudo-Boolean function subject to no constraints is called *Unconstrained nonlinear binary optimization problem* or simply, *Pseudo-Boolean Optimization* (PBO) *problem*. PBO problems exist in both minimization or maximization flavors, and in this dissertation we shall concern ourselves with the former kind. Hence, a PBO problem will have the form:

$$\min\big\{f(x) : x \in \{0,1\}^n\big\}, \tag{1.2}$$

where f is a pseudo-Boolean function in n variables. In case the multilinear polynomial representing f is quadratic, we say that Problem (1.2) is a *Quadratic Pseudo-Boolean Optimization* (QPBO) *problem* — and it is also possible to find it under the name of *Quadratic Unconstrained Binary Optimization* (QUBO) *problem* in the literature.

Pseudo-Boolean optimization problems are notoriously difficult, even in the quadratic case, as they naturally encompass a broad variety of NP-hard problems such as maximum satisfiability, maximum cut, graph coloring, simple plant location, and so on (see the survey of Boros and Hammer [29] for pseudo-Boolean formulations of some of these problems and the book of Garey and Johnson [73] for concepts and results on NP-hardness). Because of such pervasiveness, and also because solving PBO problems is a fundamental question itself, many researchers have devoted them a considerable amount of attention over the past half a century or so.

The first attempts of solving PBO problems can be traced back to the early papers by Fortet [66, 67] and Maghout [111, 112], among others, but it was the monograph of Hammer and Rudeanu [79] that properly establish and popularized pseudo-Boolean optimization as a field — overviews can be found in Boros and Hammer [29], and in Crama and Hammer [49, 51]. In the 60's and 70's, several authors proposed to handle the PBO Problem (1.2) by reformulating it as an integer linear programming problem, as follows:

1. in the objective function (1.1), replace each nonlinear monomial $\prod_{i \in S} x_i$ by a new variable y_S, and
2. set up linear constraints forcing $y_S = \prod_{i \in S} x_i$ in all optimal solutions of the resulting 0/1 LP.

Such *linearization* techniques have given rise to an enormous amount of literature.

For a (partial) overview see, e.g., Burer and Letchford [37], Hansen, Jaumard and Mathon [84], Sherali and Adams [136, 137], and the references therein.

A different approach was proposed by Rosenberg [127], in which he showed that the general PBO Problem (1.2) can also be efficiently reduced to an "equivalent" quadratic case (QBPO). More precisely, Rosenberg's procedure works as follows:

1. select two variables, say x_i, x_j such that the product $x_i x_j$ appears in a monomial of degree at least 3 in the objective function f;
2. let h_{ij} be the function obtained upon replacing each occurence of the product $x_i x_j$ by a new variable y_{ij} in f;
3. let $g_{ij} = h_{ij} + M(x_i x_j - 2x_i y_{ij} - 2x_j y_{ij} + 3y_{ij})$, where M is a large enough positive number.

It is not hard to verify that for each value of x, the minimum of g_{ij} over the auxiliary variable y_{ij} is exactly equal to $f(x)$: indeed, the minimizer is $y^*_{ij} = x_i x_j$, and the penalty term vanishes for this value. The same step can be repeated until the degree of the resulting polynomial is equal to 2, that is, until a *quadratization* of f is obtained.

Rosenberg's result establishes that every pseudo-Boolean function has a quadratization that can be computed in polynomial time. Perhaps for lack of efficient quadratic optimization algorithms, his approach, however, did not lead to practical applications for about 30 years. Meanwhile, much progress has been done on solving QPBO problems: depending on the structure of the objective function (e.g., on its density), instances of QPBO involving about 100–200 variables are now frequently solved to optimality, whereas heuristic algorithms provide solutions of excellent quality for instances with up to a few thousand variables. Among recent contributions on QPBO, we can mention for example the experimental work with exact algorithms by Billionnet and Elloumi [19], Boros, Hammer, Sun and Tavares [30], Hansen and Meyer [85], Helmberg and Rendl [87], and with heuristic algorithms by Boros, Hammer and Tavares [31], Glover, Alidaee, Rego and Kochenberger [74], Glover and Hao [75], Merz and Freisleben [115].

Interestingly, much algorithmic progress on QPBO has been due to the computer vision research community, where quadratic pseudo-Boolean functions (often dubbed "energy functions" in this framework) have proved successful in modeling specific applications such as image restoration or scene reconstruction; see, e.g., Boykov, Veksler and Zabih [33], Kolmogorov and Rother [102], Kolmogorov and Zabik [103], Rother, Kolmogorov, Lempitsky and Szummer [130]. Fast QPBO heuristics building on the roof-duality framework initially introduced by Hammer, Hansen and Simeone [78] have been developed by these researchers (see also Boros et al. [30]) and can efficiently handle the very large-scale, specially structured, sparse instances arising in computer vision applications. Furthermore, these heuristics (often based on maximum-flow/minimum-cut techniques) result in a good quantity of *autarkies* or *persistencies*, i.e., they are able to fix some binary variables to the values they would attain at a certain local minimum point or at every minimum point, respectively, and this is particularly valuable for the computer vision community.

In recent years, the same community has shifted its attention to more complex models, where the "energy function" to be minimized is a higher-degree pseudo-Boolean function. As their binary optimization problems are very large-scale, frequently having 10^5 to 10^6 variables and about 10^7 terms in their polynomial representation, the traditional approach based on integer linear formulations (linearization) described before is not satisfactory, as it would require a similarly high number of variables and constraints, making them practically intractable at today's computing technology. Researchers then tried to extend the roof duality ideas to these higher settings, but only partial success was attained for the cubic case by Wang and Kleinberg [148], and combinatorial methods for higher degrees remains largely elusive.

The computer vision community then turned its focus into Rosenberg's method, but the high number of large, positive coefficients resulting from the penalty-term approach proved to be unsuccessful, and few autarkies and/or persistencies are found as reported, for instance, by Ishikawa [91, 92]. However, the idea that different quadratization techniques could lead to improved results remained and several authors started to investigate it. A convenient and precise definition of quadratization is as follows.

Definition 1.1. *Let $f(x) = f(x_1, x_2, \ldots, x_n)$ be a pseudo-Boolean function on $\{0,1\}^n$. We say that a pseudo-Boolean function $g(x, y)$ is a* quadratization *of $f(x)$ if $g(x, y)$ is* *a quadratic multilinear polynomial depending on x and on m* auxiliary *binary variables* $y_1, y_2, \dots, y_m$*, such that*

$$f(x) = \min\big\{g(x,y) : y \in \{0,1\}^m\big\} \quad \textit{for all} \quad x \in \{0,1\}^n. \tag{1.3}$$

Clearly, if $g(x,y)$ is a quadratization of $f(x)$, then

$$\min\big\{f(x) : x \in \{0,1\}^n\big\} = \min\big\{g(x,y) : x \in \{0,1\}^n, y \in \{0,1\}^m\big\},$$

so that the minimization of $f(x)$ is reduced through this transformation to the QPBO problem of minimizing $g(x,y)$. Of course this QPBO problem remains NP-hard, but as mentioned above, much progress has been made in solving large instances of QPBO, either exactly or heuristically.

Several researchers have met considerable success with approaches based on generating a quadratization $g(x,y)$ of the objective function $f(x)$, and on minimizing g instead of f; see, e.g., Boros and Gruber [26], Boykov, Veksler and Zabih [33], Fix, Gruber, Boros and Zabih [63, 64], Freedman and Drineas [69], Ishikawa [91, 92], Kolmogorov and Zabih [103], Ramalingam, Russell, Ladický and Torr [126], Rother, Kohli, Feng and Jia [129], etc. In particular, quadratization has emerged as one of the most successful approaches to the solution of very large-scale PBO models arising in computer vision applications.

Many quadratization procedures proposed in the above literature (in particular, the procedures of Freedman and Drineas, and of Ishikawa) yield special types of quadratizations, namely, quadratizations without products of auxiliary variables.

Definition 1.2. *A quadratic pseudo-Boolean function $g(x,y)$ on $\{0,1\}^{n+m}$ is called* y-linear *if its polynomial representation does not contain monomials of the form $y_i y_j$ for $i, j \in [m]$, $i \neq j$.*

When $g(x,y)$ is y-linear, it can be written as $g(x,y) = q(x) + \sum_{i=1}^m \ell_i(x) y_i$, where $q(x)$ is quadratic in x and each ℓ_i is a linear function of x. Then, when minimizing g over y, each product $\ell_i(x) y_i$ simply takes the value $\min\{0, \ell_i(x)\}$, that is,

$$f(x) = \min_{y \in \{0,1\}^m} g(x,y) = q(x) + \sum_{i=1}^m \min\{0, \ell_i(x)\}. \tag{1.4}$$

Hence, a y-linear quadratization of $f(x)$ produces an alternative representation of f in the x-variables only. It is also worth noticing that y-linear quadratizations can be viewed as piecewise linear functions of the x-variables, apart from the q part.

Below, we present pointers to previous work on the subject, discuss our results in more details, and mention the main ideas behind them.

1.2.1 Previous Work

After Rosenberg's original approach, the quadratizations that followed can be described as *termwise quadratization procedures* and are based on the following scheme. For a real number c, let $\operatorname{sgn}(c) = +1$ (resp., -1) if $c \geq 0$ (resp., $c < 0$). Then, given $f(x)$ as in Equation (1.1),

1. for each $S \subseteq [n]$, let $g_S(x, y_S)$ be a quadratization of the monomial $\operatorname{sgn}(a_S) \prod_{i \in S} x_i$, where y_S (for $S \subseteq [n]$) are disjoint vectors of auxiliary variables (one vector for each S);
2. let $g(x, y) = \sum_{S \subseteq [n]} |a_S| \, g_S(x, y_S)$.

Then, $g(x, y)$ is a quadratization of $f(x)$. Various choices of the subfunctions $g_S(x, y_S)$ can be found in the literature. When a_S is negative, Kolmogorov and Zabih [103] for cubic monomials, and later Freedman and Drineas [69] for the general degree case, suggest using what we should call the *standard quadratization*

$$g_S(x, y) = (|S| - 1)y - \sum_{i \in S} x_i y, \tag{1.5}$$

where y is a single auxiliary variable.

When a_S is positive, the choice of an appropriate function g_S is less obvious. The first possibility is through complementation, that is, if $j \in S$, let $S_j := S \setminus \{j\}$ in order to obtain

$$\prod_{i \in S} x_i = \prod_{i \in S_j} x_i - \overline{x}_j \prod_{i \in S_j} x_i. \tag{1.6}$$

The second term of this decomposition can be quadratized by the standard quadratization just noticing that both symbols x_j and $\overline{x}_j$ are placeholders for binary values, and the first term can be handled recursively. This results in a quadratization $g_S(x, y)$ using $|S| - 2$ auxiliary variables for each positive monomial. This trick along with some variations was also realized by Boros and Gruber [26], Rother, Kohli, Feng and Jia [129], and others.

Ishikawa [91, 92] then introduced a new transformation, dubbed Higher-Order Clique Reduction (HOCR), and showed that positive monomials can actually be quadratized using $\lfloor(|S| - 1)/2\rfloor$ auxiliary variables, and this is currently the best available bound for positive monomials.

Fix [62] was the first to study quadratizations of pseudo-Boolean functions other than monomials. Specifically, he studied quadratizations of symmetric pseudo-Boolean functions, whose output values deppend only on the number of ones present in the input. Using some simple, but clever local constructions, Fix showed that every symmetric pseudo-Boolean function in n variables can quadratized with at most $n - 1$ auxiliary variables.

1.2.2 Our Results and Techniques

In a nutshell, our contributions start with a scheme that generalizes some existing termwise quadratizations and also introduces some new ones; then it moves to the first aggregative approach in which collections of terms sharing a common part are quadratized together, resulting in a new algorithm of increased quality for some computer vision problems; it then advances to the study of quadratizations of symmetric pseudo-Boolean functions, where we obtain some new lower and upper bounds on the number of auxiliary variables necessary to quadratize arbitrary and some specific functions of this class, the latter being constructive, i.e., producing new effective quadratizations; afterwards, we investigate quadratizations of a pseudo-Boolean function from a global perspective, showing non termwise procedures that provide sub-exponential upper bounds on the number of auxiliary variables necessary to quadratize any pseudo-Boolean function, and also showing lower bounds that essentially match those upper bounds, thus implying that almost all pseudo-Boolean functions require a large number of auxiliary variables to be quadratized; and as an epilogue, we introduce a polyhedral cone, the *quadratization cone*, that can be used to generate quadratizations of a specific pseudo-Boolean function, and use the information acquired for negative monomials to complete characterize all their quadratizations. We briefly elaborate about these contributions below.

Some termwise quadratizations appearing in the literature works out by complementing one or more of the term's variables, then splitting it into two or more sub-terms, and afterwards recurring in such idea, or applying a different technique, or both. That is the case of Equation (1.6) in the previous subsection. As it turns out, complementation is not the only operation that can be used to split terms. In fact, inspired by the *consensus* operation between terms of a Boolean function in *Disjunctive Normal Form* (DNF) — see Crama and Hammer [51] for definitions, and notice that they are just dual concepts of resolution between clauses in a CNF formula, which we use and define in Chapter 2 — we show that any minimal set of Boolean functions (seen as pseudo-Boolean functions) whose summation when minimized evaluates to one can be used to split a term in as many parts as the cardinality of the set plus one. In spite of not being an earthshaking result, it shed an interesting light in the myriad of possible ways of quadratizing single monomials. This result was published Boros and Gruber [26] and will be presented in Section 3.3.

The more terms the multilinear polynomial representation of a pseudo-Boolean function has, the more likely it is for some of these terms to share a subset of their variables. A simple, yet considerable good idea consists in factoring those terms and replacing the common part by a single auxiliary variable, while taking care of the uncommon parts recursively or through different techniques, depending whether they are positive or negative. Clearly, a multitude of possibilities exist regarding which parts to factor and how to deal with the remainders. It turns out that the simple strategy of listing the variables of the function, factoring the positive terms accordingly to that order, one variable at a time, recurring on that idea for the positive remainders and using Freedman and Drineas' standard quadratization (1.5) for the negative ones produces very good results for a class of pseudo-Boolean functions occurring in some computer vision applications: the number of persistencies found by the QPBO algorithm implemented by Kolmogorov and Rother [102] after this transformation reached 96% versus 80% resulted by Ishikawa's HOCR method. This splitting of common parts scheme was introduced in Boros and Gruber [26], applied to computer vision problems in Fix, Gruber, Boros, and Zabih [63, 64], and will be described in Section 3.4.

Symmetric pseudo-Boolean functions are the ones whose value depends solely on the number of ones in the input. Despite being a small class of functions, it is an important class as it includes monomials and some famous Boolean functions as *parity*, *majority*, *mod-k*, among others. Moreover, it is a natural logical step to investigate quadratizations in such a small and rich class in order to acquire better intuition and understanding of quadratizations, before aiming at larger goals. We improve upon the result of Fix [62] and show that for any symmetric pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, there are $0 < \epsilon_i \leq 1$ with $i \in [n]$ such that f can be represented uniquely in the form

$$f(x) = \sum_{i=0}^{n} \alpha_i \left[i - \epsilon_i - \sum_{r=1}^{n} x_r \right]^{-},$$

where the notation $[a]^{-}$ means $\min\{0, a\}$, for any real value a. This representation theorem, together with a shifting argument, allows us to show that every symmetric pseudo-Boolean function in n variables admits a y-linear quadratization with at most $n-2$ auxiliary variables. By specializing this result, we are able to show that specific functions like the positive monomial, t-out-of-n, exact-t, parity, and co-parity, admit y-linear quadratizations with at most $\lfloor (n-1)/2 \rfloor$, $\lceil n/2 \rceil$, $\lfloor n/2 \rfloor$, $\lfloor n/2 \rfloor$, and $\lfloor (n-1)/2 \rfloor$, respectively. Notice we are able to match Ishikawa's [91, 92] upper bound for the positive monomial. We then show that there are symmetric pseudo-Boolean functions whose quadratizations must involve at least $\Omega(\sqrt{n})$ auxiliary variables and whose y-linear quadratizations must involve at least $\Omega(n/\log n)$ auxiliary variables. Using different techniques, we show that every y-linear quadratization for the parity function in n variables requires at least $\Omega(\sqrt{n})$ auxiliary variables. These results appeared in Anthony, Boros, Crama, and Gruber [5] and will be described in Sections 3.5 and 3.7.

We then initiate a systematic study of quadratizations of pseudo-Boolean functions, adopting a more global perspective and investigating some of the properties of the class of *all* quadratizations of a given function. Through the use of some dimension-based linear algebraic arguments, we are able to show that for almost all pseudo-Boolean functions in n variables, any quadratization and any y-linear quadratization must use at least $\Omega(2^{n/2})$ and $\Omega(2^n/n)$ auxiliary variables, respectively. For bounded degree, we show that for almost all pseudo-Boolean functions of degree d, any quadratization must involve at least $\Omega(n^{d/2})$ auxiliary variables. These lower bounds are of considerable importance since the complexity of minimizing the quadratic function $g(x, y)$ heavily depends on the number of binary variables (x, y) — some other factors as number of terms influence it as well. We also show that these bounds are almost or essentially tight, providing algorithms based on extremal combinatorics constructions over the set $\{0, 1\}^n$, also known as *Boolean hypercube* of dimension n, that produce, for any pseudo-Boolean function in n variables, a quadratization and a y-linear quadratization with at most $O(2^{n/2})$ and $O\big(\frac{2^n}{n} \cdot \log n\big)$ auxiliary variables, respectively — and $O(n^{d/2})$ extra variables for the bounded degree case. The combinatorial construction of the regular quadratization case, in which products of auxiliary variables are allowed, is related to that of 2-*bases* of a set system; in the y-linear case, it is based on *Turán systems.* Both these lower and upper bounds rely on a concept we introduce and call it *universal sets*, which are sets of Boolean functions that allow every pseudo-Boolean function to be expressed as linear combinations of quadratic products of them. Universal sets play a pivotal role in our developments and we believe their properties and applicability to go beyond the quadratization realm, thus potentially spawning new research in the future. These results appeared in Anthony, Boros, Crama, and Gruber [4] and will be presented in Sections 3.6 and 3.7.

Finally, we introduce a family of *polyhedral cones* of quadratizations of pseudo-Boolean functions. While our understanding of these objects is still at an introductory level, we were able to acquire some good intuition of quadratizations of negative monomials, which led us to complete characterize all quadratizations of them. Although this result may appear to be rather narrow, it turns out to be more complex than one may expect at first sight: the proof is very long. It appeared in Anthony, Boros, Crama, and Gruber [4], and will also be presented in Section 3.9.

All the aforementioned results along with some extra explanations will be presented in Chapter 3.

1.2.3 Other Related Work

Buchheim and Rinaldi [35, 36] have developed a very different type of reduction of nonlinear binary optimization problems to the quadratic case. Their approach is polyhedral: essentially, they show that the separation problem for the polyhedron defined by the classic linearization of a PBO can be reduced to a separation problem for the quadratic case. The authors use a transformation which replaces each monomial of a pseudo-Boolean function by a product of two lower-degree monomials. They also mention in [35] some connections between their approach and Rosenberg's substitution technique, but the relation to more general quadratization schemes is unclear and remains to be explored.

Kahl and Strandmark [97, 98] have also devised a very different approach to reduce high-degree polynomials to the quadratic case, which they called *Generalized Roof Duality* (GRD). It relies on the characterization of submodular functions expressible as quadratic polynomials of Živný, Cohen and Jeavons [151], and finds the best submodular relaxation of the given function. Hence, it can be partly seen as another global approach to quadratizations (as it does not focus on rewriting individual monomials). In order to find such relaxation, the authors have to solve a linear programming problem, which turns out to be expensive in the setting of computer vision computations — despite the number of persistencies attained being very high. A drawback of this approach is that it can only be used for cubic or degree 4 polynomials, and is doubtful that the method can be generalized, for reasons related to being NP-hard to recognize if a multilinear polynomial of degree 4 or higher represents a submodular function (see Nemhauser, Wolsey, and Fisher [121], Gallo and Simeone [72], and Crama [48]). The authors propose yet some heuristics to be used instead of solving the linear programming problem that also find a good rate of persistencies. Nevertheless, experimental results have shown (cf. Fix, Gruber, Boros, and Zabih [63, 64]) that these heuristics are several times slower than either Ishikawa's HOCR [91, 92] or our splitting of common parts technique (cf. Section 3.4). An experimental comparison with our global methods of Section 3.6, specially in the bounded degree case presents itself as an interesting direction to be addressed in the future.

Quadratizations have also been independently investigated in the constraint satisfaction literature. A pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is *submodular* if

$$f(x \vee y) + f(x \wedge y) \leq f(x) + f(y) \quad \text{for all} \quad x, y \in \{0,1\}^n,$$

where $(x \vee y)_j = x_j \vee y_j$ and $(x \wedge y)_j = x_j \wedge y_j$ for all indices $j \in [n]$. Živný, Cohen and Jeavons [151] proved, among other results, that there are submodular pseudo-Boolean function of degree 4 (and in 4 variables) that do not admit submodular quadratizations. It is worth mentioning that a key step in their proof was obtained through computerized search. Related questions are also investigated in Kolmogorov [101] and in Živný and Jeavons [153].

Another line of research that makes use of the concept of auxiliary variables is that of convex hierarchies of relaxations of polyhedra and spectrahedra. These hierarchies, now commonly referred to as "extended formulations" are routinely applied to combinatorial optimization problems. Its start can be traced back to the work of Balas [15], and the prominent families of hierarchies are those of Lovász and Schrijver [110], Balas, Ceria, and Cornuéjols [16], Sherali and Adams [135], Lasserre [107, 108], and Parrilo [124] (see Laurent [109] and Conforti, Cornuéjols and Zambelli [44] for some comparisons). Also, their strengths and limits have been thoroughly investigated inside the theoretical computer science community; see e.g., Arora, Bollobás, Lovász, and Tourlakis [9], Charikar, Makarychev, and Makarychev [42], Chlamtac and Tulsiani [43], Fiorini, Massar, Pokutta, Tiwary, and de Wolf [61], and Braum, Fiorini, Pokutta, and Steurer [34]. At the moment, despite some superficial resemblance between the ideas behind those formulations and that of quadratization, no formal link has been established and we believe more investigation is needed.

1.3 Prerequisites, Conventions, and Some Definitions

We make an effort and try to introduce all concepts that we use, either directly or indirectly, to obtain our results. Nevertheless, those introductions are usually brief, and sometimes even a bit terse. We opted in favor of such strategy mainly for two reasons:

1. there are very good accounts in the literature about such topics, many of which we cite whenever appropriate; and

2. having decided otherwise, the size of this dissertation could (and most likely would) easily become unmanageable.

In any way, we apologize in advance for any discomfort that this may cause to the reader. That being said, we expect the reader to be somewhat familiar with the fields of algorithm analysis and design, combinatorics, computational complexity, linear algebra, linear programming, and the theory of Boolean and pseudo-Boolean functions. Good references are the books of Cormen, Leiserson, Rivest, and Stein [47] (also, to a certain extent, Vazirani [146], Williamson and Shmoys [149]); Jukna [95], van Lint and Wilson [145]; Arora and Barak [8], Garey and Johnson [73]; Anthony and Harvey [6], Strang [141]; Korte and Vygen [104], Schrijver [132]; Crama and Hammer [51], the survey of Boros and Hammer [29] and the Ph.D. dissertation of Tavares [143], respectively.

Besides mentioning authorship along the text, we provide the names of the authors of a theorem, lemma, proposition, etc., between parenthesis right before the result's statement, every time. Therefore, any unclaimed result is to be considered as part of our contribution.

We decided to use only two sequential counters to number the objects inside each chapter: one for equations, and one for definitions, propositions, lemmas, theorems, corollaries, facts, notations, and remarks. So, for instance, Theorem 2.38 means that such theorem is the 38th named object inside Chapter 2, and not that there are other 37 theorems before. Our decision was made in order to facilitate navigation through the text. As mentioned, equations are numbered independently, but as their references always appear inside parenthesis, that should not create confusion.

We now provide some definitions and notations that will be useful, or that provide some extra insights in what we will do later on.

We denote the set of real numbers by $\mathbb{R}$ and its restriction to the nonnegative side of the real line by $\mathbb{R}_{\geq 0}$. The set of binary or *Boolean* values is denoted simply by $\{0, 1\}$. Depending on the context, the symbols 0 and 1 may be interpreted as the numbers zero and one, or as the logical values *false* and *true*. These interpretations can sometimes be intermixed, and no confusion will arise in our usage. We denote other sets by capital letters and sometimes call them *collections*. A collection of sets (or subsets of a *ground* set) is also called a *family* and sometimes is denoted by a calligraphic or a capital greek letter. For a positive integer n, we denote by $[n] := \{1, 2, \ldots, n\}$ the set of positive integers up to n. Given a set S, its *power set* is defined as $2^S := \{T : T \subseteq S\}$. The *size* or *cardinality* of a finite set S is denoted by $|S|$ and defined as the number of elements it contains. The set of positive integers (or natural numbers) is denoted by $\mathbb{N}$. We assume familiarity with set operations as containment, intersection, union, and so on.

A *graph* G is a pair (V, E), where V is a finite set of elements called *vertices* and $E = \big\{\{u, v\} : u, v \in V \text{ and } u \neq v\big\}$ is a family of pairs of vertices, each pair called an *edge*. For convenience, we denote $\{u, v\}$ by (u, v), keeping in mind that $(u, v) = (v, u)$. That is, our graphs are *undirected*. For an edge $e = (u, v)$, we say that the vertices u and v are the *end-points* of e, that e is incident to both u and v, and that u and v are *neighbors*. For a vertex $u \in V$, its *neighborhood* is the set $N(u) := \{v \in V : (u, v) \in E\}$, and its *degree* is given by $\deg(u) := |N(u)|$. The graph G is *bipartite* with bipartition $A \dot{\cup} B = V$, denoted by $G = (A, B, E)$, if $E = \big\{\{u, v\} : u \in A \text{ and } v \in B\big\}$, i.e., no edge has both its end-points in the same element of the bipartition. A graph $H = (W, F)$ is a *subgraph* of G if $W \subseteq V$ and $F \subseteq E$; H is a *spanning subgraph* of G if $W = V$ and $F \subseteq E$. Let $D = \langle v_0, v_1, \ldots, v_k\rangle$ be a sequence of vertices of G such that $(v_{j-1}, v_j) = e_j \in E$ for all $j \in [k]$, and $v_i \neq v_j$ for all $i, j \in [k]$ with $i \neq j$. Notice that D together with edges e_j is a subgraph of G. If $v_0 = v_k$, we call D a *cycle*. If $v_0 \neq v_j$ for all $j \in [k]$, we call C a *path*. In both cases, the *length* of D is equal to k. The graph G is *connected* if for all distinct vertices $u, v \in V$ there is a path between them. The graph G is a *forest* if it does not contain any cycles, and is a *tree* if in addition it is connected.

Given a finite (ground) set H, a *set system* $\mathcal{H}$ is a family of subsets of H, that is, $\mathcal{H} \subseteq 2^H$. The set system $\mathcal{H}$ is sometimes also referred as the *hypergraph* $\mathscr{H} = (H, \mathcal{H})$, in which the elements of H are its vertices and the sets of $\mathcal{H}$ its *hyperedges*. It is clear that graphs are hypergraphs whose hyperedges have cardinality 2.

The n-fold product of the binary set $\{0,1\}$, namely, $\{0,1\}^n$ is the set of binary n-tuples and can also be seen as a vector of dimension n (with n entries or coordinates). The set $\{0,1\}^n$ is also called *Boolean hypercube* of dimension n as it can be interpreted as the following graph:

$$\mathscr{B} = \Big(\{0,1\}^n, \big\{(x,y) : x, y \in \{0,1\}^n \text{ and } \big||x| - |y|\big| = 1\big\}\Big),$$

where $|z| := \sum_{i=1}^{n} z_i$ is the *Hamming weight* of the vector $z \in \{0,1\}^n$, the number of ones in z. That is, $\mathscr{B}$ is a graph on 2^n vertices and whose edges are between pairs of vertices differing in exactly one coordinate. A subgraph of $\mathscr{B}$ obtained by fixing the value of some coordinates of its vertices together with all the existing edges between those vertices is a *subcube* of $\mathscr{B}$. By convention, $\mathscr{B}$ is a subcube of itself.

We make use of the standard asymptotic notations: $O(\cdot)$, $\Theta(\cdot)$, $\Omega(\cdot)$, $o(\cdot)$, and $\omega(\cdot)$. Namely, *big-oh*, *big-theta*, *big-omega*, *little-oh* or *omicron*, and *little-omega*, respectively. For definitions and examples of use, consult e.g. the book by Cormen et al. [47].

We say a function $f : \mathbb{N} \to \mathbb{N}$ (or $f : \mathbb{N} \to \mathbb{R}$) is *quasi-linear* if $f \in O\big(n^{1+o(1)}\big)$ and it is *nearly linear* if $f \in O\big(n^{1+\varepsilon}\big)$ for some constant $\varepsilon > 0$ small enough; it is *quasi-polynomial* or *super-polynomial* if $f \in O\big(n^{\mathrm{polylog}(n)}\big)$; and it is *sub-exponential* if $f \in O\big(2^{o(n)}\big)$.

Following tradition (and literature), we denote by $\mathsf{DTIME}(f(n))$ and $\mathsf{NTIME}(f(n))$ the classes of *decision problems*, i.e., problems whose yes-or-no solutions can be computed in deterministic and nondeterministic time, respectively, upper bounded by $O(f(n))$, where n is the length of the input's description and $f : \mathbb{N} \to \mathbb{N}$ (or $f : \mathbb{N} \to \mathbb{R}$) is some constructible function. The standard complexity classes of deterministic and nondeterministic polynomial time are then $\mathsf{P} := \mathsf{DTIME}(\mathrm{poly}(n))$ and $\mathsf{NP} := \mathsf{NTIME}(\mathrm{poly}(n))$, the complement of NP is $\mathsf{co\text{-}NP} := \mathsf{co\text{-}NTIME}(\mathrm{poly}(n))$, and quasi-polynomial deterministic time is $\mathsf{QP} := \mathsf{DTIME}(n^{\mathrm{polylog}(n)})$. A multitude of other complexity classes can be defined in a similar way.

The class NP has a nice alternative interpretation: it is the class of decision problems in which every instance whose answer is yes has a polynomial sized *certificate* (also called *witness*) that can be verified in polynomial time and confirm the correctness of the claim, i.e., that the answer for such instance is indeed a yes. Clearly $\mathsf{P} \subseteq \mathsf{NP}$ as one can just compute the solution in polynomial time and in case of a yes answer, use an empty certificate. Notice that it can be very hard, computationally speaking, to find a certificate. In other words, it may be the case that more than polynomial time is needed to find certificates to problems in NP. This is just one way of stating the most famous open question in computational complexity: is $\mathsf{P} \overset{?}{=} \mathsf{NP}$.

Given two decision problems Π and Λ, a *polynomial (time) reduction* from Π to Λ is an algorithm that receives an instance $\mathscr{I}_\Pi$ to the former, runs in polynomial time in the length of $\mathscr{I}_\Pi$'s description (size of $\mathscr{I}_\Pi$), and outputs an instance $\mathscr{I}_\Lambda$ to the latter such that $\mathscr{I}_\Pi$ has a yes answer in Π if and only if $\mathscr{I}_\Lambda$ has a yes answer in Λ.

A decision problem is NP-hard if every problem in NP can be polynomially reduced to it, and it is NP-complete if in addition it belongs to NP itself.

Finally, for a problem $\Pi \in \mathsf{NP}$, a *Probabilistic Checkable Proof* (PCP) system for Π is a polynomial time randomized algorithm, called *verifier*, that receives an input $\mathscr{I}_\Pi$ for Π of size n, has access to $O(\log n)$ random bits, and has a constant number of oracle access to a *proof* π (a relaxed notion of witness, in a different encoding scheme). That is, the verifier does not have complete access to π, but can query it a constant number of times, using the random bits to decide which positions of π to query. The crucial issue is that the proof might not be trustworthy, in the sense that the answer to $\mathscr{I}_\Pi$ may be a yes and π is correctly certifying it, or the answer to $\mathscr{I}_\Pi$ may be a no and π is maliciously trying to fool the verifier in believing the answer is a yes. The verifier is required to accept correct proofs and rejects incorrect proofs with probability at least 2/3. Perhaps surprisingly, such verifiers do exist for every $\Pi \in \mathsf{NP}$, as stated by the celebrated PCP Theorem (cf. Arora and Safra [12]), and they have many applications in proving hardness of approximation results for NP-hard optimization problems.

2
Hardness of Approximation of Pure Horn CNF Minimization

In this chapter, we study the hardness of approximation of clause minimum and literal minimum representations of pure Horn functions in n Boolean variables. We show that unless $\mathsf{P} = \mathsf{NP}$, it is not possible to approximate in polynomial time the minimum number of clauses and the minimum number of literals of pure Horn CNF representations to within a factor of $2^{\log^{1-o(1)} n}$. This is the case even when the inputs are restricted to pure Horn 3-CNFs with $O(n^{1+\varepsilon})$ clauses, for some small positive constant ε. Furthermore, we show that even allowing sub-exponential time computation, it is still not possible to obtain constant factor approximations for such problems unless the Exponential Time Hypothesis is false.

The chapter is organized as follows. We introduce some basic concepts about pure Horn functions, mostly to fix notation and nomenclature, together with some recent theoretical tools regarding Boolean functions that we shall use in Section 2.1. We then present the problem on which our reduction is based, the Label-Cover problem, in Section 2.2. The reduction to pure Horn CNFs, its proof of correctness, and the polynomial time hardness of approximation result are shown in Section 2.3. In Section 2.4, we extend that result to pure Horn 3-CNF formulae and address the case of minimizing the number of literals. In Section 2.5 we show that sub-exponential time availability gives smaller but still super-constant hardness of approximation factors. We then offer some final thoughts in Section 2.6.

The results of this chapter have appeared in Boros and Gruber [25, 27].

2.1 Preliminaries

In this section we succinctly define the main concepts and notations we shall use later on. For an almost comprehensive exposition, consult the book on Boolean Functions by Crama and Hammer [51].

Definition 2.1. *A Boolean function $h(x) = h(x_1, x_2, \ldots, x_n)$ in n propositional variables is a mapping $\{0,1\}^n \mapsto \{0,1\}$, that is, it is a mapping that associates n-dimensional binary vectors in $\{0,1\}^n$ to binary values. In this context, the values 0 and 1 are often interpreted as the logical concepts of falsity and truth.*

The set of variables of h is denoted by $V_h := \{v_1, \ldots, v_n\}$. It is not hard to see that there are 2^{2^n} different Boolean functions in n variables, and that each Boolean function can be specified by a table with 2^n entries, called its *true table.* In some (most, actually) circumstances however, more succinct representations are preferable, and specific kinds of formulae or expressions are used.

A *literal* is a propositional variable v_i (positive literal) or its negation $\bar{v}_i$ (negative literal). An elementary disjunction of literals

$$C = \bigvee_{i \in I} \bar{v}_i \vee \bigvee_{j \in J} v_j, \tag{2.1}$$

with $I, J \subseteq V_h$ is a *clause* if $I \cap J = \emptyset$. The set of variables it depends upon is $\mathsf{Vars}(C) := I \cup J$ and its *degree* or *size* is given by $\mathsf{deg}(C) := |I \cup J|$. It is customary to identify a clause C with its set of literals.

Definition 2.2. *A clause C as in (2.1) is called* pure *or* definite Horn *if $|J| = 1$. For a pure Horn clause C, the positive literal $v \in J$ is called its* head *and $S = C \setminus J$ is called its* subgoal *or* body. *To simplify notation, we sometimes write C simply as $\bar{S} \vee v$ or as the implication $S \longrightarrow v$.*

Definition 2.3. *A conjunction Φ of pure Horn clauses is a* pure Horn formula in Conjunctive Normal Form *(for short, pure Horn CNF). In case every clause in Φ has degree at most three, the CNF is a 3-CNF. A Boolean function h is called* pure Horn *if there is a pure Horn CNF formula $\Phi \equiv h$, that is, if $\Phi(v) = h(v)$ for all $v \in \{0,1\}^n$.*

Let $\Phi = \bigwedge_{i=1}^{m} C_i$ be a pure Horn CNF representing a pure Horn function h. We denote by

$$|\Phi|_c := m \qquad \text{and} \qquad |\Phi|_l := \sum_{i=1}^{m} \mathsf{deg}(C)$$

the numbers of clauses and literals of Φ, respectively. We say that Φ is a clause (literal) minimum representation of h if $|\Phi|_c \le |\Psi|_c$ ($|\Phi|_l \le |\Psi|_l$) for every other pure Horn CNF representation Ψ of h. With this in mind, we define

$$\tau(h) := \min\{|\Phi|_c : \Phi \text{ is a pure Horn CNF representing } h\}, \text{ and}$$
$$\lambda(h) := \min\{|\Phi|_l : \Phi \text{ is a pure Horn CNF representing } h\}.$$

Problem 2.4. *The clause (literal) pure Horn CNF minimization problem consists in determining $\tau(h)$ ($\lambda(h)$) when h is given as a pure Horn CNF. Similar definitions hold for the pure Horn 3-CNF case.*

A clause C as in (2.1) is an *implicate* of a Boolean function h if for all $v \in \{0,1\}^n$ it holds that $h(v) = 0$ implies $C(v) = 0$. An implicate is *prime* if it is inclusion-wise minimal with respect to its set of literals. The set of prime implicates of h is denoted by $\mathcal{I}^p(h)$. It is known (cf. Hammer and Kogan [81]) that prime implicates of pure Horn functions are pure Horn clauses. A pure Horn CNF Φ representing h is *prime* if its clauses are prime and is *irredundant* if the pure Horn CNF obtained after removing any of its clauses does not represent h anymore. Let us note that a clause minimun representation may involve non prime implicates, though it is always irredundant. As Hammer and Kogan [81] pointed out any Horn CNF can be reduced in polynomial time to an equivalent prime and irredundant CNF. In the sequel we shall assume all CNFs considered, including the clause minimum ones, to be prime and irredundant.

Let C_1 and C_2 be two clauses and v be a variable such that $v \in C_1$, $\bar{v} \in C_2$, and C_1 and C_2 have no other complemented literals. The *resolvent* of C_1 and C_2 is the clause

$$R(C_1, C_2) := (C_1 \setminus \{v\}) \cup (C_2 \setminus \{\bar{v}\})$$

and C_1 and C_2 are said to be *resolvable.* It is known (e.g. Crama and Hammer [51]) that if C_1 and C_2 are resolvable implicates of a Boolean function h, then $R(C_1, C_2)$ is also an implicate of h. Naturally, the resolvent of pure Horn clauses is also pure Horn.

A set of clauses $\mathcal{C}$ is *closed under resolution* if for all $C_1, C_2 \in \mathcal{C}$, $R(C_1, C_2) \in \mathcal{C}$. The *resolution closure* of $\mathcal{C}$, $R(\mathcal{C})$, is the smallest set $\mathcal{X} \supseteq \mathcal{C}$ closed under resolution. Clearly, for two sets of clauses $\mathcal{C}_1 \subseteq \mathcal{C}_2$ it holds that $R(\mathcal{C}_1) \subseteq R(\mathcal{C}_2)$ and that $R(R(\mathcal{C}_1)) = R(\mathcal{C}_1)$. Also, it is not hard to see that if $R(\mathcal{C}_1) = R(\mathcal{C}_2)$, then $\mathcal{C}_1 \equiv \mathcal{C}_2$, that is, if two sets have the same resolution closure, then they represent the same function.

For a Boolean function h, let $\mathcal{I}(h) := R(\mathcal{I}^p(h))$. Let us note that the set of all implicates of a Horn function h may, in principle, contain clauses involving arbitrary other variables, not relevant for h. To formulate proper statements one would need to make sure that such redundancies are also handled, which complicates the formulations. To avoid such complications, we focus on $\mathcal{I}(h)$ in the sequel, which is completely enough to describe all relevant representations of h.

Definition 2.5. *Let Φ be a pure Horn CNF representing a pure Horn function h and let $Q \subseteq V_h$. The* Forward Chaining *of Q in Φ, denoted by $F_\Phi(Q)$, is defined by the following algorithm. Initially, $F_\Phi(Q) = Q$. As long as there is a pure Horn clause $\bar{S} \vee v$ in Φ such that $S \subseteq F_\Phi(Q)$ and $v \notin F_\Phi(Q)$, add v to $F_\Phi(Q)$. Whenever a variable v is added to $F_\Phi(Q)$, we say that the corresponding clause $\bar{S} \vee v$ was* triggered.

The result below is pivotal in our work. It tells us that we can make inferences about a pure Horn function h using any of its pure Horn CNF representations.

Lemma 2.6 (Hammer and Kogan [82]). *Two distinct pure Horn CNFs Φ and Ψ represent the same pure Horn function h if and only if $F_\Phi(U) = F_\Psi(U)$, for all $U \subseteq V_h$. Consequently, we can do Forward Chaining in h, which we denote by $F_h(\cdot)$, through the use of any of h's representations.*

The following definitions and lemma concerning exclusive sets of clauses are useful when decomposing and studying structural properties of Boolean functions.

Definition 2.7 (Boros et al. [22]). *Let h be a Boolean function and $\mathcal{X} \subseteq \mathcal{I}(h)$ be a set of clauses. $\mathcal{X}$ is an* exclusive set of clauses of h *if for all resolvable clauses $C_1, C_2 \in \mathcal{I}(h)$ it holds that: $R(C_1, C_2) \in \mathcal{X}$ implies $C_1 \in \mathcal{X}$ and $C_2 \in \mathcal{X}$.*

An example of an exclusive set of clauses is given by the set of pure Horn implicates of a Horn function: it is not hard to see that if a resolvent is a pure Horn clause, then both the resolvable clauses must also be pure Horn.

Definition 2.8 (Boros et al. [22]). *Let $\mathcal{X} \subseteq \mathcal{I}(h)$ be an exclusive set of clauses for a Boolean function h and let $\mathcal{C} \subseteq \mathcal{I}(h)$ be such that $\mathcal{C} \equiv h$. The Boolean function $h_{\mathcal{X}} = \mathcal{C} \cap \mathcal{X}$ is called the $\mathcal{X}$-component of h.*

The following claim justifies the use of "the" in the previous definition.

Lemma 2.9 (Boros et al. [22]). *Let $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{I}(h)$, $\mathcal{C}_1 \neq \mathcal{C}_2$, such that $\mathcal{C}_1 \equiv \mathcal{C}_2 \equiv h$ and let $\mathcal{X} \subseteq \mathcal{I}(h)$ be an exclusive set of clauses. Then $\mathcal{C}_1 \cap \mathcal{X} \equiv \mathcal{C}_2 \cap \mathcal{X}$ and in particular $(\mathcal{C}_1 \setminus \mathcal{X}) \cup (\mathcal{C}_2 \cap \mathcal{X})$ also represents h.*

Proof. First, notice that since $\mathcal{C}_i$ represents h for $i \in [2]$, we have that $\mathcal{I}^p(h) \subseteq R(\mathcal{C}_i)$. Hence, assuming $\mathcal{C}_i \subseteq \mathcal{I}(h)$, gives

$$\mathcal{I}(h) = R(\mathcal{I}^p(h)) \subseteq R(R(\mathcal{C}_i)) = R(\mathcal{C}_i) \subseteq R(\mathcal{I}(h)) = \mathcal{I}(h),$$

which then implies that $\mathcal{X} \subseteq R(\mathcal{C}_1) = R(\mathcal{C}_2)$. These relations plus the fact that $\mathcal{X}$ is exclusive, i.e., no clause in $\mathcal{C}_i \setminus \mathcal{X}$ can appear as a parent clause in a resolution leading to a resolvent in $\mathcal{X}$, leads to $\mathcal{X} \subseteq R(\mathcal{C}_i \setminus (\mathcal{C}_i \setminus \mathcal{X})) = R(\mathcal{C}_i \cap \mathcal{X})$, which in turn implies that $R(\mathcal{X}) \subseteq R(R(\mathcal{C}_i \cap \mathcal{X})) = R(\mathcal{C}_i \cap \mathcal{X})$. As $\mathcal{C}_i \cap \mathcal{X} \subseteq \mathcal{X}$, we have that $R(\mathcal{X}) = R(\mathcal{C}_i \cap \mathcal{X})$ and therefore, it holds that $\mathcal{X} \equiv \mathcal{C}_1 \cap \mathcal{X} \equiv \mathcal{C}_2 \cap \mathcal{X}$. □

The above lemma is a particularly useful and important tool in our work. Loosely speaking, it means that once we are able to identify an exclusive component g of a function h, we can separately study g. Moreover, we can draw conclusions about g using any of its representations (even alternate between distinct representations as convenient) and then reintegrate the acquired knowledge into the analysis of h.

The Forward Chaining procedure provides us with a convenient way of identifying exclusive families for pure Horn functions, as stated in the next lemma. This result appeared recently in Boros et al. [23]. As we make explicit use of it in the analysis of our construction, its proof is included for completeness.

Lemma 2.10 (Boros et al. [23]). *Let Φ be a prime pure Horn CNF representing the function h, let $W \subseteq V_h$ be such that $F_\Phi(W) = W$, and define the set*

$$\mathcal{X}(W) := \{C \in \mathcal{I}(h) : \mathsf{Vars}(C) \subseteq W\}.$$

Then $\mathcal{X}(W)$ is an exclusive family for h.

Proof. Let W be as specified in the lemma's statement and suppose there are clauses $C_1, C_2 \in \mathcal{I}(h)$ such that $R(C_1, C_2) \in \mathcal{X}(W)$ but $\{C_1, C_2\} \not\subseteq \mathcal{X}(W)$. By the definitions of $\mathcal{X}(W)$ and of resolution, all but one of the variables in $C = C_1 \cup C_2$ must belong to W and that variable, say $v \in C \setminus W$, is precisely the variable upon which C_1 and C_2 are resolvable, that is, v occurs as head in one of those clauses. Now, since for the same input the forward chaining outcome is independent of the pure Horn representation of h, it follows that $v \in F_\Phi(W) \neq W$, a contradiction. $\square$

2.2 The Label-Cover Problem

The Label-Cover problem is a graph labeling promise problem formally introduced in Arora et al. [7] as a combinatorial abstraction of interactive proof systems (two-prover one-round in Feige et al. [58] and Feige and Lovász [60], and probabilistically checkable in Arora and Motwani [11] and Arora and Safra [12]). It comes in maximization and minimization flavors (linked by a "weak duality" relation) and is probably the most popular starting point for hardness of approximation reductions. In this section, we introduce a minimization version that is best suited for our polynomial time results. Later in Section 2.5, when dealing with sub-exponential time results, we shall mention its maximization counterpart.

Definition 2.11. *A* Label-Cover instance *is a quadruple $\mathcal{L}_0 = (G, L_0, L'_0, \Pi_0)$, where $G = (X, Y, E)$ is a bipartite graph, L_0 and L'_0 are disjoint sets of labels for the vertices in X and Y, respectively, and $\Pi_0 = (\Pi^0_e)_{e \in E}$ is a set of constraints with each $\Pi^0_e \subseteq L_0 \times L'_0$ being a non-empty relation of admissible pairs of labels for the edge e. The* size *of $\mathcal{L}_0$ is equal to $|X| + |Y| + |E| + |L_0| + |L'_0| + |\Pi_0|$.*

Definition 2.12. *A* labeling *for $\mathcal{L}_0$ is any function $f_0 : X \to 2^{L_0}, Y \to 2^{L'_0} \setminus \{\emptyset\}$ assigning subsets of labels to vertices. A labeling f_0* covers *an edge (x, y) if for every label $\ell'_0 \in f_0(y)$ there is a label $\ell_0 \in f_0(x)$ such that $(\ell_0, \ell'_0) \in \Pi^0_{(x,y)}$. A* total-cover *for $\mathcal{L}_0$ is a labeling that covers every edge in E. $\mathcal{L}_0$ is said to be* feasible *if it admits a total-cover.*

Following Arora and Lund [10], a way to guarantee that a Label-Cover instance is feasible is by imposing an extra condition on it, namely, that there is a label $\ell'_0 \in L'_0$ such that for each edge $e \in E$, there is a label $\ell_0 \in L_0$ with $(\ell_0, \ell'_0) \in \Pi^0_e$. In this way, a labeling assigning ℓ'_0 to each vertex in Y and the set L_0 to each vertex in X is clearly a total-cover. However, all Label-Cover instances that we shall use are, by construction, guaranteed to be feasible. Therefore, we shall not dwell on such imposition and shall consider only feasible Label-Cover instances in the sequel.

Definition 2.13. *For a total-cover f_0 of $\mathcal{L}_0$, let $f_0(Z) := \sum_{z \in Z} |f_0(z)|$ with $Z \subseteq X \cup Y$. The* cost *of f_0 is given by $\kappa(f_0) := f_0(X)/|X|$ and f_0 is said to be* optimal *if $\kappa(f_0)$ is minimum among the costs of all total-covers for $\mathcal{L}_0$. This minimum value we denote by $\kappa(\mathcal{L}_0)$.*

Observe that the feasibility of $\mathcal{L}_0$ implies that $1 \leq \kappa(f_0) \leq |L_0|$, for any total-cover f_0. Also, without loss of generality, we can assume that G has no isolated vertices as they do not influence the cost of any labeling.

We now give an example of a Label-Cover instance $\mathcal{L}_0$. Let $U := \{u_1, \ldots, u_n\}$ be a set of Boolean variables and let $\Phi := \bigwedge_{i=1}^{s} \phi_i$ be a formula in CNF such that each clause of Φ depends on k variables of U (as in a variation of the satisfiability problem in which each clause has exactly k literals). For a clause $\phi \in \Phi$ and a variable $u \in U$, we write $u \in \phi$ whenever ϕ depends on u.

The bipartite graph $G = (X, Y, E)$ is constructed from Φ as follows. Let $X := \{x_1, \ldots, x_{ks}\}$ have a vertex for every occurrence of a variable in Φ, and let $Y := \{1, \ldots, s\}$ have a vertex for every clause $\phi \in \Phi$. Let $X(u) \subseteq X$ denotes the set of vertices corresponding to the variable u, and define

$$E := \Big\{(x, j) \in X \times Y : x \in X(u) \text{ and } u \in \phi_j\Big\},$$

that is, each vertex $j \in Y$ is connected to all occurrences of all variables in the clause ϕ_j.

Define the label-sets as $L_0 := \{0, 1\}$ and $L'_0 := \{0, 1\}^k$. For an edge $(x, j) \in E$, assume that $x \in X(u)$ and that u is the i-th variable in ϕ_j, and define

$$\Pi^0_{(x,j)} := \Big\{(a_i, (a_1, \ldots, a_k)) : \phi_j(a_1, \ldots, a_k) = \mathsf{True}\Big\},$$

where $a_i \in L_0$ and $(a_1, \ldots, a_k) \in L'_0$.

Now, it is not hard to see that in this case, there is a total-cover f_0 with $\kappa(f_0) = 1$ if and only if Φ is satisfiable. Notice that choosing $k \geq 3$ establishes the NP-completeness of the problem of deciding if an optimal total-cover for a given Label-Cover instance has cost equal to one.

The above example was adapted from Dinur and Safra [55]. Their original version is used in the proof of Theorem 2.18. In that context however, Φ is a non-Boolean satisfiability instance produced by a probabilistic checkable proof system and the label-sets involved are larger (see Remark 2.19 below).

Definition 2.14. *A total-cover f_0 is* tight *if $f_0(Y) := \sum_{y \in Y} |f_0(y)| = |Y|$, i.e., if for every $y \in Y$, it holds that $|f_0(y)| = 1$.*

Lemma 2.15. *Every* Label-Cover *instance $\mathcal{L}_0$ admits a tight, optimal total-cover.*

Proof. Suppose f_0 as in Definition 2.12 is a minimally non-tight, optimal total-cover for $\mathcal{L}_0$. Hence, there is a $y \in Y$ such that $|f_0(y)| > 1$. Let $\ell'_0 \in f_0(y)$ and define a new labeling g where $g(z) = f_0(z)$ for all $z \in X \cup (Y \setminus \{y\})$ and $g(y) = f_0(y) \setminus \{\ell'_0\}$. Note that $g(y) \neq \emptyset$ and that every edge (x, y) for $x \in N(y) := \{z \in X : (z, y) \in E\}$ is covered (for f_0 is a total-cover). Moreover, clearly $\kappa(f_0) = \kappa(g)$. Hence, g is an optimal total-cover for $\mathcal{L}_0$ in which $g(Y) = \sum_{y \in Y} |g(y)| < f_0(Y)$, contradicting the minimality of f_0. The result thus follows. □

Notation 2.16. *For $\mathcal{L}_0$ being a* Label-Cover *instance as in Definition 2.11, define $r := |X|$, $s := |Y|$, $m := |E|$, $\lambda := |L_0|$, $\lambda' := |L'_0|$, $\pi_e := |\Pi^0_e|$ for $e \in E$, and set $\pi := \sum_{e \in E} \pi_e$.*

Problem 2.17. *For any $\rho > 1$, a* Label-Cover *instance $\mathcal{L}_0$ has covering promise ρ if it falls in one of two cases: either there is a tight, optimal total-cover for $\mathcal{L}_0$ of cost* 1, *or every tight, optimal total-cover for $\mathcal{L}_0$ has cost at least ρ. The* Label-Cover$_\rho$ *problem is a promise problem which receives a* Label-Cover *instance with covering promise ρ (also known as a ρ-promise instance) as input and correctly classify it in one of those two cases.*

Notice the behavior of Label-Cover$_\rho$ is left unspecified for non-promise instances. Therefore, any answer is acceptable in such case. Due to this characteristic, the Label-Cover$_\rho$ problem is also referred as a *gap-problem* with gap ρ in the literature.

The result below is the basis for the polynomial time hardness we shall exhibit.

Theorem 2.18 (Dinur and Safra [55])**.** *Let c be any constant in $(0, 1/2)$ and $\rho_c(s) := 2^{(\log s)^{1-1/\delta_c(s)}}$ with $\delta_c(s) := (\log \log s)^c$. There are* Label-Cover *instances $\mathcal{L}_0$ with covering-promise $\rho_c(s)$ such that it is* NP*-hard to distinguish between the cases in which $\kappa(\mathcal{L}_0)$ is equal to* 1 *or at least $\rho_c(s)$.*

The closer to $1/2$ the above constant c gets, the larger the hardness of approximation factor becomes. Therefore, from now on we shall consider that c is fixed to a value close to $1/2$.

Remark 2.19. *Every* Label-Cover *instance produced by Dinur and Safra's reduction is feasible, has covering-promise $\rho_c(s)$, and satisfies the following relations: $r = s\lfloor \delta_c(s) \rfloor$, $\lambda = \Theta(\rho_c(s))$, $\lambda' = \Theta(\rho_c(s)^{\delta_c(s)}) = o(s)$, $s\lfloor \delta_c(s) \rfloor \leq m \leq s^2 \lfloor \delta_c(s) \rfloor$, and $\pi \leq m\lambda\lambda' = O(s^2 \delta_c(s) \rho_c(s)^{\delta_c(s)+1}) = o(s^3)$, for s as specified in Notation 2.16. It is then immediate that each such instance has size $o(s^3)$.*

We now introduce a refined version of the Label-Cover definitions, in which the vertices in the sets X and Y have their own copies of the label-sets L_0 and L'_0, respectively. We then show that all structural and approximation properties are preserved in this new version.

Definition 2.20. *Let $\mathcal{L}_0 = (G, L_0, L'_0, \Pi_0)$ be a feasible* Label-Cover *instance and consider the sets $L_x := \{(x, \ell_0) : \ell_0 \in L_0\}$ for each vertex $x \in X$, and $L'_y := \{(y, \ell'_0) :$

$\ell'_0 \in L'_0\}$ for each vertex $y \in Y$. Also, define the sets $L := \bigcup_{x\in X} L_x$, $L' := \cup_{y\in Y} L'_y$, and $\Pi := \bigcup_{(x,y)\in E} \Pi_{(x,y)}$, with

$$\Pi_{(x,y)} := \left\{ \big((x,\ell_0),(y,\ell'_0)\big) : (\ell_0,\ell'_0) \in \Pi^0_{(x,y)} \right\}.$$

The quadruple $\mathcal{L} = (G, L, L', \Pi)$ is called a refinement *of $\mathcal{L}_0$.*

It is clear that $|L_x| = \lambda$ for each vertex $x \in X$, $|L_y| = \lambda'$ for each vertex $y \in Y$, $|\Pi_e| = \pi_e$ for each edge $e \in E$, $|\Pi| = \pi$, and that a labeling for $\mathcal{L}$ is a mapping f such that $x \mapsto f(x) \subseteq L_x$ for each vertex $x \in X$, and $y \mapsto f(y) \subseteq L'_y$, $f(y) \neq \emptyset$ for each vertex $y \in Y$. Furthermore, the remaining definitions and concepts can be adapted in a straight forward fashion, and the size of a refined instance is also $o(s^3)$.

Lemma 2.21. *For any $\rho > 0$, there is a one-to-one cost preserving correspondence between solutions to the* Label-Cover$_\rho$ *problem and to its refined version.*

Proof. It is easy to see that f_0 is a (tight) total-cover for the Label-Cover$_\rho$ problem if and only if f is a (tight) total-cover for its refinement, where $f(x) = \{(x,\ell_0) : \ell_0 \in f_0(x)\}$ for every $x \in X$, and $f(y) = \{(y,\ell'_0) \in L'_y : \ell'_0 \in f_0(y)\}$ for every $y \in Y$ (or $f(y) = (y, f_0(y))$ if the total-covers are tight). Furthermore it is clear that $\kappa(f_0) = \kappa(f)$, in any case. $\square$

Henceforth, all the Label-Cover instances used are assumed to be of the refined kind. For more information on the Label-Cover problem and its applications, consult the survey by Arora and Lund [10], the article by Moshkovitz and Raz [119], and the book by Arora and Barak [8].

2.3 Reduction to pure Horn CNFs and a Polynomial Time Hardness Result

Our first reduction starts with a Label-Cover instance $\mathcal{L}$ as input and produces a pure Horn CNF formula Φ, which defines a pure Horn function h. The driving idea behind this reduction is that of tying the cost of tight, optimal total-covers of $\mathcal{L}$ to the size of clause minimum prime pure Horn CNF representations of h.

With this in mind, let $\mathcal{L} = (G = (X, Y, E), L, L', \Pi)$ be a Label-Cover instance (in compliance with Theorem 2.18 and Definition 2.20), and let d and t be positive integers to be specified later. Both d and t will be used as (gap) amplification devices. For nonnegative integers n, define $[n] := \{1, \ldots, n\}$.

Associate propositional variables $u(\ell)$ with every label $\ell \in L \cup L'$, $e(x, y, i)$ and $e(x, y, \ell', i)$ with every edge $(x, y) \in E$, every label $\ell' \in L'_y$ and every index $i \in [d]$. Let $v(j)$, for indices $j \in [t]$, be extra variables, and consider the following families of clauses:

$$\text{(a)} \quad u(\ell) \wedge u(\ell') \longrightarrow e(x, y, \ell', i) \qquad \forall\ (x, y) \in E,\ (\ell, \ell') \in \Pi_{(x,y)},\ i \in [d];$$

$$\text{(b)} \quad \bigwedge_{z \in N(y)} e(z, y, \ell', i) \longrightarrow e(x, y, i) \qquad \forall\ (x, y) \in E,\ \ell' \in L'_y,\ i \in [d];$$

$$\text{(c)} \quad e(x, y, i) \longrightarrow e(x, y, \ell', i) \qquad \forall\ (x, y) \in E,\ \ell' \in L'_y,\ i \in [d];$$

$$\text{(d)} \quad \bigwedge_{i \in [d]} \bigwedge_{(x,y) \in E} e(x, y, i) \longrightarrow u(\ell) \qquad \forall\ \ell \in L \cup L';$$

$$\text{(e)} \quad v(j) \longrightarrow u(\ell) \qquad \forall\ j \in [t],\ \ell \in L \cup L';$$

where as before, $N(y) := \{x \in X : (x, y) \in E\}$ is the open neighborhood of the vertex $y \in Y$.

Definition 2.22. *Let us call Ψ and Φ the* canonical *pure Horn CNF formulae defined, respectively, by the families of clauses (a) through (d) and by all the families of clauses above. Let g and h be, in that order, the pure Horn functions they represent.*

The construction presented above can be divided into two parts. The families of clauses appearing in Ψ, namely, clauses of type (a) through (d), form an independent core since the function g is an exclusive component of the function h (as we shall show). This core can be analysed and minimized separately from the remainder, and its role is to reproduce the structural properties of the Label-Cover instance. In more details, clauses of type

(a) correspond to the constraints on the pairs of labels that can be assigned to each edge;

(b) will assure that edges $(x, y) \in E$ are covered, enforcing the matching of the labels assigned to all the neighbors of vertex y;

(c) assure that if an edge can be covered in a certain way, then it can be covered in all legal ways — thus implying that it is not necessary to keep track of more than one covering possibility for each edge in clause minimum prime representations;

(d) translate the total-cover requirement and reintroduce all the labels available ensuring that if a total-cover is achievable, so are all the others; this reintroduction of labels is paramount to the proper functioning of the reduction as explained below.

The family of clauses occurring in $\Phi \setminus \Psi$, namely, the clauses of type (e), constitutes the second part of the construction. These clauses have the role of introducing an initial collection of labels, which sole purpose is to help achieve the claimed hardness of approximation result. The intended behavior is as follows.

Consider initially that $d = t = 1$. It is known that given any subset of the variables of h as input, the Forward Chaining procedure in any pure Horn CNF representation of h will produce the same output (cf. Lemma 2.6). In particular, for the singleton $\{v(1)\}$, the output in Φ will be the set with all the variables of h, and so will be the output obtained in any clause minimum prime pure Horn CNF representing h.

The reintroduction of labels performed by the family of clauses (d) may allow for some clauses of type (e) to be dropped without incurring in any loss. In slightly more details, as long as a subset of the family of clauses (e) introduces enough labels so that the Forward Chaining procedure in Φ is able to eventually trigger the family of clauses (d), the remaining clauses of type (e) can be dismissed. All the missing labels will be available by the end of the procedure's execution. It is not hard to see at this point that subsets of retained clauses of type (e) and total-covers of the Label-Cover instance in which the reduction is based are in one-to-one correspondence.

Now, supposing that a clause minimum prime pure Horn CNF representation of h resembles the canonical form Φ, we just have to compensate for the number of clauses in

Ψ to obtain a distinguishable gap that mimics the one exhibited by the Label-Cover (as a promise) problem. This is achieved by making the gap amplification parameter t which the clauses of type (e) depend upon large enough.

However, in principle, there is no guarantee that a clause minimum prime pure Horn CNF representation of h, say Υ, resembles Φ or that Υ has any clause of type (e) whatsoever. It may be advantageous to Υ to have prime implicates where $v(j)$, for $j \in [t]$, occurs in their subgoals or prime implicates with variables other than $u(\ell)$, for $\ell \in L \cup L'$, occurring as heads. Furthermore, the number of prime implicates in Υ involving $v(j)$ might simply not depend on the number of labels. Indeed, if $d = 1$ as we are supposing, whenever the number of edges $|E|$ turns out to be strictly smaller than $\kappa(\mathcal{L})$, the cost of an optimal total-cover for $\mathcal{L}$, it would be advantageous for Υ to have prime implicates of the form $v(j) \longrightarrow e(x, y, 1)$, for $(x, y) \in E$. This not only breaks the correspondence mentioned above, but it renders the gap amplification device t innocuous and the whole construction useless.

We manage to overcome the above difficulty throughout a second amplification device, the parameter d which the clauses of type (a) through (d) depend upon. As we shall prove in Lemma 2.28, setting $d = 1 + r\lambda + s\lambda'$ (which is strictly larger than the total number of labels available in $\mathcal{L}$ — cf. Notation 2.16 and Definition 2.20) allows us to control the shape of the prime implicates involving variables $v(j)$ in prime pure Horn clause minimum representations of h: they will be precisely some of the clauses of type (e). Moreover, after showing that the function g is an exclusive component of the function h, we shall see that we do not need to concern ourselves with the actual form of clause minimum prime pure Horn CNF representations of g. Therefore, in a sense, the canonical form Φ has indeed a good resemblance to a clause minimum prime pure Horn CNF representing h, and the intended behavior is achieved in the end.

2.3.1 Correctness of the CNF Reduction

In this subsection, we formalize the discussion presented above. We will constantly use the canonical representations Φ and Ψ to make inferences about the functions h and g they respectively define, and such inferences will most of the time be made throughout

Forward Chaining. We start with some basic facts about Φ and Ψ.

Lemma 2.23. *Let d and t be as above and let r, s, m, λ, λ', and π be as in Notation 2.16. It holds that the number of clauses and variables in Φ are, respectively,*

$$|\Phi|_c = (t+1)(r\lambda + s\lambda') + d(\pi + 2m\lambda') \quad \textit{and} \quad |\Phi|_v = t + dm(\lambda'+1) + r\lambda + s\lambda'.$$

In Ψ, those numbers are, respectively,

$$|\Psi|_c = r\lambda + s\lambda' + d(\pi + 2m\lambda') \quad \textit{and} \quad |\Psi|_v = dm(\lambda'+1) + r\lambda + s\lambda'.$$

Proof. For $\#(\alpha)$ denoting the number of clauses of type (α) in Φ, simple counting arguments show that the equalities

$$\begin{array}{lll} \#(a) = d\pi & \#(c) = dm\lambda' & \#(e) = t(r\lambda + s\lambda') \\ \#(b) = dm\lambda' & \#(d) = r\lambda + s\lambda' & \end{array}$$

hold. For the number of variables, just notice there are $r\lambda + s\lambda'$ variables $u(\ell)$, dm variables $e(x,y,i)$, $dm\lambda'$ variables $e(x,y,\ell',i)$, and t variables $v(j)$. To conclude the proof, just remember that the only difference between Φ and Ψ is the absence of the family of clauses of type (e) in the latter. $\square$

Considering the bounds for r, m, λ, λ', and π provided in Remark 2.19, the above result immediately implies that as long as the quantities d and t are polynomial in s, namely, the number of vertices in Y, the construction of Φ from $\mathcal{L}$ can be carried out in polynomial time in s. More meaningfully, it can be carried out in polynomial time in $n = |\Phi|_v$, the number of variables of h.

We now establish the pure Horn function g as an exclusive component of h. This structural result allows us to handle g in a somewhat black-box fashion. Specifically, as we shall see briefly, it is not required of us to precisely know all the properties and details of a clause minimum representation of g. We can mainly concentrate on the study of the prime implicates that might involve the variables in h that are not in g, namely, the variables $v(j)$, for $j \in [d]$.

Lemma 2.24. *The function g is an exclusive component of the function h. Consequently, g can be analysed and minimized separately.*

Proof. Let V_g be the set of variables occurring in Ψ. By definition, these are the variables the function g depends upon. Since no clause in Ψ has head outside V_g, it is immediate that V_g is closed under Forward Chaining in Φ. As Φ represents h, Lemma 2.10 then implies that $\mathcal{X}(V_g) := \{C \in \mathcal{I}(h) : \mathsf{Vars}(C) \subseteq V_g\}$ is an exclusive family for h. This gives that $\Psi = \Phi \cap \mathcal{X}(V_g)$ is an $\mathcal{X}(V_g)$-exclusive component of h (cf. Definition 2.8) and therefore, that g is an exclusive component of h. Now, using Lemma 2.9, we obtain a proof of the second claim as wished. □

With some effort, it is possible to prove that Ψ is a clause minimum prime pure Horn CNF representation of g. For our proofs however, a weaker result suffices.

Lemma 2.25. *Let Θ be a clause minimum prime pure Horn CNF representation of g. We have $|\Psi|_c/(\lambda + \lambda') \leq |\Theta|_c \leq |\Psi|_c$.*

Proof. The upper bound is by construction. For the lower bound, observe that each variable of Ψ appears no more than $\lambda + \lambda'$ times as a head. As in any clause minimum representation of g they must appear as head at least once, the claim follows. □

The next lemma is a useful tool in showing whether two different representations of the pure Horn function h are equivalent.

Lemma 2.26. *For all indices $j \in [t]$, it holds that $F_h(\{v(j)\}) = V_g \cup \{v(j)\}$.*

Proof. It is enough to show that $\{e(x, y, i) : (x, y) \in E, i \in [d]\} \subseteq F_\Phi(\{v(j)\})$, for a fixed $j \in [t]$. The inclusion would be false if there existed a label $\ell'' \in L \cup L'$ such that $u(\ell'') \notin F_\Phi(\{v(j)\})$. As for every label $\ell \in L \cup L'$, $v(j) \longrightarrow u(\ell)$ is a clause in Φ, this cannot happen. Hence, the inclusion holds, implying the claim. □

The next couple of lemmas deal with the structure of prime implicates involving the variables $v(j)$, for $j \in [t]$. The first states a simple, but useful fact which is valid in any representation of h. The second proves how the amplification device depending on the parameter d shapes those prime implicates in clause minimum representations of h to the desired form: $v(j) \longrightarrow u(\ell)$, with $\ell \in L \cup L'$.

Lemma 2.27. *A variable $v(j)$, for some index $j \in [t]$, is never the head of an implicate of h. Moreover, every prime implicate of h involving $v(j)$ is quadratic.*

Proof. The first claim is straight forward as all implicates of h can be derived from Φ by resolution, and $v(j)$ is not the head of any clause of Φ. By Lemma 2.26, $v(j) \longrightarrow z$ is an implicate of h for all $z \in V_g$. Since h is a pure Horn function, the claim follows. $\square$

Lemma 2.28. *Let $d = 1 + r\lambda + s\lambda'$. In any clause minimum prime pure Horn CNF representation of h, the prime implicates involving the variables $v(j)$ have the form $v(j) \longrightarrow u(\ell)$, for all indices $j \in [t]$, and for some labels $\ell \in L \cup L'$.*

Proof. Let $\Upsilon = \Theta \wedge \Gamma$ be a clause minimum prime pure Horn CNF representation of h, with Θ being a clause minimum pure Horn CNF representation of g. According to Lemma 2.27, all prime implicates of h involving the variables $v(j)$ are quadratic. So, for all indices $j \in [t]$ and all indices $i \in [d]$ define the sets

$$\Gamma_0^j := \Gamma \cap \{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\},$$
$$\Gamma_i^j := \Gamma \cap \{v(j) \longrightarrow e(x, y, i),\, v(j) \longrightarrow e(x, y, \ell', i) : (x, y) \in E, \ell' \in L'_y\}.$$

Our goal is to show that the chosen value for the parameter d forces all the sets Γ_i^j to be simultaneously empty and consequently, that all the prime implicates involving the variables $v(j)$ in clause minimum pure Horn CNF representations of h have the claimed form. We shall accomplish this in two steps.

Let $j \in [t]$. We first show that if a set $\Gamma_i^j \neq \emptyset$ for some index $i \in [d]$, then $\Gamma_i^j \neq \emptyset$ for all indices $i \in [d]$, simultaneously.

Claim 1. *All clauses of type (d) have the same body. Therefore, during the execution of the Forward Chaining procedure from $\{v(j)\}$, either they all trigger simultaneously or none of them do. The reason for them not to trigger is the absence of some variable $e(x, y, i)$, with $(x, y) \in E$ and $i \in [d]$, in the Forward Chaining closure from $\{v(j)\}$, i.e, $e(x, y, i) \notin F_\Upsilon(\{v(j)\})$.*

Proof. A simple inspection of the families of clauses shows that Claim (1) holds. $\square$

Now, for each index $i \in [d]$, let Ξ_i be the collection of clauses of types (a), (b), and (c) that depend on i.

Claim 2. *It holds that*

$$e(x, y, i) \in F_{\Gamma_0^j \cup \Xi_i}(\{v(j)\}) \quad \textit{if and only if} \quad e(x, y, i') \in F_{\Gamma_0^j \cup \Xi_{i'}}(\{v(j)\}),$$

for all indices $i, i' \in [d]$, with $i \neq i'$.

Proof. Notice that the families of clauses (a), (b), and (c) are completely symmetric with respect to the indexing variable i. Moreover, for $i_1 \neq i_2$, the clauses indexed by i_1 do not interfere with the clauses indexed by i_2 during an execution of the Forward Chaining procedure. In other words, variables depending upon i_1 do not trigger clauses indexed by i_2, and vice-versa. These two properties, symmetry and non interference, proves Claim (2). □

Claim 3. *If there is a variable $e(x, y, i)$, with $(x, y) \in E$ and $i \in [d]$, such that*

$$e(x, y, i) \notin F_{\Gamma_0^j \cup (\bigcup_{i \in [d]} \Xi_i)}(\{v(j)\})$$

then

$$e(x, y, i) \notin F_{\Gamma_0^j \cup (\bigcup_{i \in [d]} \Xi_i) \cup (\bigcup_{i' \neq i} \Gamma_{i'}^j)}(\{v(j)\}).$$

Moreover, this implies that $\Gamma_i^j \neq \emptyset$.

Proof. The symmetry and non interference properties of families of clauses (a), (b), and (c) also justifies the first part of Claim (3). To see it, just notice that were the claim to be false, the prime implicates in $\Gamma_{i'}^j$ would be trigging clauses involving the variable $e(x, y, i)$ in an execution of the Forward Chaining procedure. Since $i' \neq i$, this cannot happen. The second part follows immediately from the validity of the first part together with the fact that Υ represents h. □

To finish the first step, notice that since Claim (3) is valid for any $i \in [d]$, Claim (2) implies that if $\Gamma_i^j \neq \emptyset$ for some index $i \in [d]$, then $\Gamma_i^j \neq \emptyset$ for all indices $i \in [d]$, simultaneously.

For the second step, suppose that $\Gamma_i^j \neq \emptyset$ for all indices $i \in [d]$. We then have that

$$\gamma := \sum_{i\in[d]} |\Gamma_i^j| \geq d = 1 + r\lambda + s\lambda' = 1 + |L \cup L'|,$$

that is, γ is strictly larger than the number of all available labels in $\mathcal{L}$. This implies that the following pure Horn CNF

$$\begin{aligned} \Delta_j &:= \left(\Upsilon \setminus \bigcup_{i\in d} \Gamma_i^j\right) \cup \left\{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\right\} \\ &= \Theta \cup \left(\left(\Gamma \setminus \bigcup_{i\in d} \Gamma_i^j\right) \cup \left\{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\right\}\right), \end{aligned}$$

has fewer clauses than Υ (or more precisely, it implies that $|\Delta_j|_c <= |\Upsilon|_c - 1$).

Now, since Θ is a (clause minimum) representation of the exclusive component g, and since the set of clauses $\{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\}$ makes all available labels reachable by Forward Chaining from $\{v(j)\}$, it follows that $F_{\Delta_j}(\{v(j)\}) = V_g \cup \{v(j)\}$. Furthermore, the change in clauses did not influence the Forward Chaining procedure from any other variable (other than $v(j)$), and thus $F_{\Delta_j}(\{w\}) = F_\Upsilon(\{w\})$ for all variables $w \neq v(j)$. Thus, Lemma 2.26 implies that Δ_j is a representation of h.

We then have that Δ_j is a shorter representation for h, contradicting the optimality of Υ. Therefore, the sets $\Gamma_i^j = \emptyset$ for all indices $i \in [d]$. As the above arguments do not depend on any particular value of j, they can be repeated for all of them. $\square$

The next property we can show more generally for any prime and irredundant CNF of h.

Definition 2.29. *Let $d = 1 + r\lambda + s\lambda'$ and let Υ be a prime and irredundant pure Horn CNF representation of h. For each $j \in [t]$, consider the set*

$$S_j = \left\{\ell \in L \cup L' : v(j) \longrightarrow u(\ell) \in \Upsilon\right\}$$

and define the function $f_j : X \to L, Y \to L'$ given by $f_j(x) = S_j \cap L_x$ for vertices $x \in X$ and $f_j(y) = S_j \cap L'_y$ for vertices $y \in Y$.

The next three lemmas provide important properties of the functions f_j above.

Lemma 2.30. *Let Υ be as in the above Definition. For all indices $j \in [t]$ and vertices $y \in Y$, it holds that $|f_j(y)| \leq 1$.*

Proof. Let Υ be as in Definition 2.29 and suppose indirectly that the claim is false, that is, there is an index $j \in [t]$ and a vertex $y \in Y$ such that $|f_j(y)| > 1$.

During the proof, recall that the chosen value for the parameter d implies, according to Lemma 2.28, that all prime implicates of Υ involving the variable $v(j)$ must have the form $v(j) \longrightarrow u(\ell)$, with $\ell \in L \cup L'$.

Let $\ell' \in f_j(y)$ and define the expression

$$\Upsilon' := \Upsilon \setminus \{v(j) \longrightarrow u(\ell')\}.$$

It is enough to show that $F_{\Upsilon'}(\{v(j)\}) = V_g \cup \{v(j)\}$, that is, that Υ' is also a representation of h (cf. Lemma 2.26). Suppose that is not the case. Since Υ and Υ' differ only in the clause $v(j) \longrightarrow u(\ell')$, it must be the case that $u(\ell') \notin F_{\Upsilon'}(\{v(j)\})$. This happens if the clause of type (d) associated to $u(\ell')$ is not triggered. For this to occur, there must be an edge $(x, y) \in E$ and an index $i \in [d]$ such that $e(x, y, i) \notin F_{\Upsilon'}(\{v(j)\})$.

Now, for y and i as above, notice that: (i) variable $e(x, y, i)$ would be included in $F_{\Upsilon'}(\{v(j)\})$ as long as there were a label in L'_y such that the corresponding clause of type (b) were triggered; and (ii) once such clause of type (b) were triggered, the appropriated clauses of type (c) would trigger, thus making the other clauses of type (b) associated to y and i to also trigger.

Therefore, for $e(x, y, i)$ to not belong to $F_{\Upsilon'}(\{v(j)\})$, it must be the case that for every label $\ell'' \in f_j(y) \setminus \{\ell'\}$ there exists a vertex $z(\ell'') \in N(y)$ for which

$$e(z(\ell''), y, \ell'', i) \notin F_{\Upsilon'}(\{v(j)\}).$$

For this latter relation to be true, we must have that the clauses

$$u(\ell) \wedge u(\ell'') \longrightarrow e(z(\ell''), y, \ell'', i) \tag{2.2}$$

are not triggered in the Forward Chaining procedure on Υ' starting with $\{v(j)\}$, for every label $\ell \in f_j(z(\ell''))$ with $(\ell, \ell'') \in \Pi_{(z(\ell''), y)}$.

However, according to Definition 2.29, for each label $\ell'' \in f_j(y) \setminus \{\ell'\}$ and each label $\ell \in f_j(z(\ell''))$, there are clauses $v(j) \longrightarrow u(\ell'')$ and $v(j) \longrightarrow u(\ell)$, respectively, in Υ and, consequently, in Υ'. This implies that the clauses (2.2) are triggered, which implies that $u(\ell') \in F_{\Upsilon'}(\{v(j)\})$, which then implies that Υ' is also a representation of h. Since this contradicts the irredundancy of Υ, it follows that $|f_j(y)| \leq 1$, thus concluding the proof. □

Lemma 2.31. *Let Υ be a clause minimum prime pure Horn CNF of h. Then it is prime and irredundant, so Definition 2.29 applies. We claim that for all indices $j \in [t]$ and vertices $y \in Y$, it holds that $|f_j(y)| \geq 1$.*

Proof. Suppose that the claim is false, that is, there is an index $j \in [t]$ and a vertex $y \in Y$ such that $|f_j(y)| = 0$.

Then clauses $v(j) \longrightarrow u(\ell')$, for all labels $\ell' \in L'_y$, are absent from Υ. Recall that the chosen value for the parameter d implies that all prime implicates of Υ involving $v(j)$ are quadratic (Lemma 2.28).

Thus, no clause of type (a) dependent on the vertex y is triggered during a Forward Chaining from $\{v(j)\}$ and hence, no clauses of type (b) and of type (c) dependent on y are triggered either. This gives that the variables $e(x, y, i)$, for all vertices $x \in N(y)$ and all indices $i \in [d]$, do not belong to the Forward Chaining closure (from $\{v(j)\}$). Therefore, no clause of type (d) is triggered and no label $\ell' \in L'_y$ is reintroduced. In other words, it is the case that $u(\ell') \notin F_\Upsilon(\{v(j)\})$ and hence, that $F_\Upsilon(\{v(j)\}) \neq F_h(\{v(j)\})$. By Lemma 2.26, Υ does not represent h, a contradiction. So, it must be the case that $|f_j(y)| \geq 1$. □

Combining the two lemmas above, we have the following tight result.

Corollary 2.32. *Let Υ be a clause minimum prime pure Horn CNF of h. Then, for all indices $j \in [t]$ and vertices $y \in Y$, it holds that $|f_j(y)| = 1$.* □

The next lemma shows that the functions f_j are indeed tight total-covers.

Lemma 2.33. *Let Υ be a clause minimum prime pure Horn CNF of h. For each index $j \in [t]$, the function f_j is a tight total-cover for $\mathcal{L}$.*

Proof. Let $j \in [t]$. By construction (cf. Definition 2.29), f_j is a labeling for $\mathcal{L}$. Suppose however, that f_j is not a total-cover. Hence, there exists an edge $(x, y) \in E$ and a label $\ell' \in f_j(y)$ such that for all labels $\ell \in f_j(x)$, it holds that $(\ell, \ell') \notin \Pi_{(x,y)}$. Since Υ is clause minimum, no variable $e(x, y, \ell', i)$ belongs to $F_\Upsilon(\{v(j)\})$, for any index $i \in [d]$. This is so because the variables $e(x, y, \ell', i)$ do not occur as heads in any clause of Υ (cf. Lemma 2.28). Now, using Lemma 2.26, we obtain that Υ does not represent h, a contradiction. To conclude the proof, just notice that Corollary 2.32 implies that f_j is tight. $\square$

In order to relate the size of a clause minimum representation of h to the cost of an optimal solution to $\mathcal{L}$, we need a comparison object. Let f be a tight total-cover for $\mathcal{L}$ and consider the following subfamily of clauses:

$$\text{(e')} \quad v(j) \longrightarrow u(\ell) \qquad \forall\, j \in [t],\ x \in X,\ y \in Y,\ \ell \in f(x) \cup f(y),$$

with $f(x) \subseteq L_x$ and $f(y) \subseteq L'_y$. Let Φ_f be the *refined canonical* (with respect to f) pure Horn CNF formula resulting from the conjunction of Ψ with the clauses of type (e').

Lemma 2.34. Φ_f *represents* h.

Proof. Suppose the opposite. As g is also an exclusive component of Φ_f, that means $u(\ell'') \notin F_{\Phi_f}(\{v(j)\})$ for some label $\ell'' \in L \cup L'$ and index $j \in [t]$. For that to happen, for any index $i \in [d]$ there must be an edge $(x, y) \in E$ such that $e(x, y, i) \notin F_{\Phi_f}(\{v(j)\})$ and for every $\ell' \in L'_y$ there is a vertex $z \in N(y)$ such that $e(z, y, \ell', i) \notin F_{\Phi_f}(\{v(j)\})$ as well. But since f is a tight total-cover, there is a pair of labels $(\ell_z, \ell'_y) \in \Pi_{(z,y)}$ triggering the clause $u(\ell_z) \wedge u(\ell'_y) \longrightarrow e(z, y, \ell'_y, i)$ as both $u(\ell_z)$ and $u(\ell'_y)$ belong to $F_{\Phi_f}(\{v(j)\})$, contradicting $e(z, y, \ell'_y, i) \notin F_{\Phi_f}(\{v(j)\})$. Therefore, $F_{\Phi_f}(\{v(j)\}) = V_g \cup \{v(j)\}$ and the claim follows by Lemma 2.26. $\square$

Lemma 2.35. *Let Υ be a clause minimum prime pure Horn CNF of h, and let us define f_j, for all indices $j \in [t]$, as in Definition 2.29. It holds that each f_j is a tight, minimum cost total-cover for $\mathcal{L}$.*

Proof. Let $\Upsilon = \Theta \wedge \Gamma$ where Θ is an optimal representation of g and Γ consists of clauses of type (e) (cf. Lemma 2.28). As assured by Lemma 2.33, f_j are tight total-covers for $\mathcal{L}$, for each and every index $j \in [t]$. Notice that since Υ is clause minimum, it follows that all these tight total-covers have the same cost, i.e., $\kappa(f_j) = \kappa(f_k)$ for all $j, k \in [t]$.

We then have that

$$\sum_{j\in[t]}(\kappa(f_j)r + s) = t(\kappa(f_j)r + s) = |\Gamma|_c \leq |\Phi_f \setminus \Psi|_c = t(\kappa(f)r + s), \tag{2.3}$$

where f is any tight total-cover for $\mathcal{L}$ and Φ_f is the refined canonical (w.r.t. f) formula as in Lemma 2.34. In particular, Equation (2.3) holds even when f is a tight, minimum cost total-cover, thus implying that f_j is optimal as claimed. □

Remark 2.36. *The tight, optimal total-covers f_j and f_k, for $j, k \in [t]$ and $j \neq k$, might be different. As they have the same optimal cost, any one of them can be exhibited as solution to $\mathcal{L}$.*

The following corollary summarizes the work done so far.

Corollary 2.37. *Let Ψ be as in Definitions 2.22 and Υ be a clause minimum prime pure Horn CNF of h. It holds that*

$$|\Psi|_c/(\lambda + \lambda') \leq |\Upsilon|_c - t(\kappa(f)r + s) \leq \ |\Psi|_c,$$

where $\kappa(f)r + s$ is the total number of labels in a tight optimal total-cover f for $\mathcal{L}$. □

2.3.2 The CNF Hardness Result

We are now able to prove the main result of this section.

Theorem 2.38. *Let c be a fixed constant close to $1/2$. Unless $\mathsf{P} = \mathsf{NP}$, the minimum number of clauses of a pure Horn function on n variables cannot be approximated in polynomial time (depending on n) to within a factor of*

$$\rho_c(n^\varepsilon) \geq 2^{\varepsilon(\log n)^{1-1/\delta_c(n)}} = 2^{\log^{1-o(1)} n},$$

where $\delta_c(n) = (\log\log n)^c$, even when the input is restricted to CNFs with $O(n^{1+2\varepsilon})$ clauses, for some $\varepsilon \in (0, 1/4]$.

Proof. Let $\mathcal{L}_0$ be a Dinur and Safra's LABEL-COVER promise instance (cf. Theorem 2.18) and let $\mathcal{L}$ be its equivalent refined version (cf. Definition 2.20).

Let Φ be the canonical formula constructed from $\mathcal{L}$ with the parameter d set to $d = 1 + r\lambda + s\lambda'$ (cf. Lemma 2.28), and let h be the pure Horn function defined by Φ. Let Υ be a pure Horn CNF representation of h obtained by some exact clause minimization algorithm when Φ is given as input.

Recall Notation 2.16 and for convenience, let $\delta = \delta_c(s)$ and $\rho = \rho_c(s)$. Substituting the quantities established in Lemma 2.23 into the bounds given by Corollary 2.37, and applying the values given in Remark 2.19, we have that

$$\begin{aligned}|\Upsilon|_c &\geq t(\kappa(f)r + s) + \frac{d(\pi + 2m\lambda') + r\lambda + s\lambda'}{\lambda + \lambda'}\\ &\geq st(\kappa(f)(\delta - 1) + 1) + \Omega(s^2\delta\rho^{\delta})\\ &\geq st(\kappa(f)(\delta - 1) + 1) + \omega(s^2),\end{aligned}$$

and

$$\begin{aligned}|\Upsilon|_c &\leq t(\kappa(f)r + s) + d(\pi + 2m\lambda') + r\lambda + s\lambda'\\ &\leq st(\kappa(f)\delta + 1) + O(s^3\delta\rho^{2\delta+1})\\ &\leq st(\kappa(f)\delta + 1) + o(s^4).\end{aligned}$$

Similarly, for the number of variables we have that

$$\begin{aligned}t \leq |\Upsilon|_v &= t + dm(\lambda' + 1) + r\lambda + s\lambda'\\ &\leq t + O(s^3\delta\rho^{2\delta})\\ &\leq t + o(s^4).\end{aligned}$$

Now, choosing $\varepsilon > 0$ such that $t = s^{1/\varepsilon} = \Omega(s^4)$ and supposing that $s \longrightarrow \infty$, we obtain the asymptotic expressions

$$|\Upsilon|_c = s^{(1+1/\varepsilon)}\delta_c(s)\kappa(f)(1 + o(1)) \qquad \text{and} \qquad |\Upsilon|_v = s^{1/\varepsilon}(1 + o(1)).$$

Bringing the existing gaps of the LABEL-COVER promise instances into play, we then obtain the following dichotomy

$$\kappa(\mathcal{L}) = 1 \Longrightarrow |\Upsilon|_c \leq s^{(1+1/\varepsilon)}\delta_c(s)(1+o(1)),$$
$$\kappa(\mathcal{L}) \geq \rho_c(s) \Longrightarrow |\Upsilon|_c \geq s^{(1+1/\varepsilon)}\delta_c(s)\rho_c(s)(1+o(1)).$$

Letting $n = |\Upsilon|_v$ and relating $|\Upsilon|_c$ to the number of variables of h, the above dichotomy reads as

$$\kappa(\mathcal{L}) = 1 \Longrightarrow |\Upsilon|_c \leq n^{(1+\varepsilon)}\delta_c(n^\varepsilon)(1+o(1)),$$
$$\kappa(\mathcal{L}) \geq \rho_c(s) \Longrightarrow |\Upsilon|_c \geq n^{(1+\varepsilon)}\delta_c(n^\varepsilon)\rho_c(n^\varepsilon)(1+o(1)),$$

giving a hardness of approximation factor of $\rho_c(n^\varepsilon)$ for the pure Horn CNF minimization problem (cf. Theorem 2.18). That is, any polynomial time algorithm that approximates the number of clauses of a pure Horn CNF representation of h to within a factor better than $\rho_c(n^\varepsilon)$ can be used to solve the Label-Cover promise problem for $\mathcal{L}$. This in turn would show that $\mathsf{P} = \mathsf{NP}$.

To conclude the proof, notice that since $\log \varepsilon < 0$ and $n \longrightarrow \infty$, the gap

$$\begin{aligned}\rho_c(n^\varepsilon) &= 2^{(\varepsilon \log n)^{1-1/\delta_c(n^\varepsilon)}}\\ &\geq 2^{\varepsilon(\log n)^{1-1/\delta_c(n)}}\\ &= 2^{(\log n)^{1-1/\delta_c(n)+\log\varepsilon/\log\log n}}\\ &= 2^{(\log n)^{1-o(1)}},\end{aligned}$$

where the $\log\varepsilon/\log\log n$ in the exponent is negligible compared to $1/\delta_c(n)$ as $\delta_c(n) = o(\log\log n)$, and also notice that the number of clauses is

$$n^{(1+\varepsilon)}\delta_c(n^\varepsilon) \leq n^{1+2\varepsilon}.$$

□

2.4 Pure Horn 3-CNFs and Minimizing the Number of Literals

In this section, we extend our hardness result in two ways. First, we prove that it still holds when the pure Horn function h is represented through a pure Horn 3-CNF, that is, in which each clause has at most three literals (and at least two, since h is pure Horn). Second, we build upon this first extension and prove that a similar bound holds when trying to determine a literal minimum pure Horn 3-CNF representation of h.

Let again $d = 1 + r\lambda + s\lambda'$ and t be a positive integer to be specified later. As in the previous section, both d and t will be used as (gap) amplification devices.

A brief inspection of our construction in Section 2.3 shows that the clauses of type (b) and (d) may have arbitrarily long subgoals, with long meaning strictly more than three literals. The idea is then to modify the construction locally so that each long clause is replaced by a gadget consisting of a collection of quadratic or cubic new clauses. Each gadget is designed to preserve the original logic implications of the clause it replaces.

Specifically, we replace the clauses of type (b) in a similar way to what is done in the reduction from SAT to 3-SAT (cf. Garey and Johnson [73]), that is, in a *linked-list* fashion. For each vertex $y \in Y$, let $d(y) := |N(y)|$ be its degree and let $\langle z_y^1, z_y^2, \ldots, z_y^{d(y)} \rangle$ be an arbitrary, but fixed ordering of its neighbors. Associate new propositional variables $e(\beta, x, y, \ell', i)$ with all edges $(x, y) \in E$, all labels $\ell' \in L'_y$, all indices $i \in [d]$, and all indices $\beta \in [d(y) - 2]$. As before, we have that $d = 1 + r\lambda + s\lambda'$. Replace the clauses of type (b) by the families of clauses below:

(b$_1$) $\displaystyle\bigwedge_{z \in N(y)} e(z, y, \ell', i) \longrightarrow e(x, y, i) \qquad \forall\ (x, y) \in E,\ \ell' \in L'_y,\ i \in [d],\ d(y) \leq 2;$

(b$_2$) $e(z_y^1, y, \ell', i) \wedge e(z_y^2, y, \ell', i) \longrightarrow e(1, x, y, \ell', i)$

$$\forall\ (x, y) \in E,\ \ell' \in L'_y,\ i \in [d],\ d(y) \geq 3;$$

(b$_3$) $e(z_y^{\beta+2}, y, \ell', i) \wedge e(\beta, x, y, \ell', i) \longrightarrow e(\beta + 1, x, y, \ell', i)$

$$\forall\ (x, y) \in E,\ \ell' \in L'_y,\ i \in [d],\ \beta \in [d(y) - 3];$$

(b$_4$) $e(z_y^{d(y)}, y, \ell', i) \wedge e(d(y) - 2, x, y, \ell', i) \longrightarrow e(x, y, i)$

$$\forall\ (x, y) \in E,\ \ell' \in L'_y,\ i \in [d],\ d(y) \geq 3.$$

The clauses of type (b$_1$) are exactly the quadratic and cubic clauses of type (b) in the original construction. The clauses of types (b$_2$), (b$_3$), and (b$_4$) rely on the new variables $e(\beta, x, y, \ell', i)$ to handle the remaining original clauses of type (b) through a series of split and link operations. It is not hard to see that these new families of clauses retain the symmetry and non interference properties possessed by the original ones they replaced. Furthermore, they will be part of an exclusive component. These characteristics, as we shall see, contributes to the transference of most of the lemmas from the previous section in a rather verbatim fashion.

Trying to apply the same technique to the clauses of type (d) generates the following problem. Recall that $m = |E|$. All the $r\lambda + s\lambda'$ clauses of type (d) have the same long subgoal with dm literals; they only differ in their heads. In a first attempt to emulate what was done above to the clauses of type (b), we may decide to replace this long subgoal with a single linked list and to replicate its last node once for each label $\ell \in L \cup L'$. After ordering these literals in an arbitrary, but fixed order, introducing $dm-2$ new variables, say $e(\zeta)$ for $\zeta \in [dm-2]$, and performing split and link operations, we obtain the linked list whose last node has subgoal $e(x', y', i') \wedge e(dm-2)$, for some edge $(x', y') \in E$ and some index $i' \in [d]$. This subgoal is then replicated, thus spanning the clauses

$$e(x', y', i') \wedge e(dm-2) \longrightarrow u(\ell),$$

for all labels $\ell \in L \cup L'$. Now, it is not hard to see that the prime implicates $v(1) \longrightarrow e(x', y', i')$ and $v(1) \longrightarrow e(dm-2)$ completely bypass the amplification device dependent on the parameter d, reintroducing all the available labels. Similar to what was said before, this renders the gap amplification device dependent on parameter t innocuous and the whole construction useless. It is also not hard to see that introducing a new amplification device and recurring on the whole idea does not solve the problem as we end up in a similar situation.

In a second attempt to emulate what was done to the clauses of type (b), we may decide to introduce different linked lists to different clauses of type (d). Notice that this implies that each of these lists must be indexed by one of the available labels in $L \cup L'$. In order for this new collection of clauses to be reached in an execution of the Forward Chaining procedure, this indexing scheme must be back propagated into the clauses of types (a), (b_1) through (b_4), and (c). That is, every clause of the exclusive component must now be indexed by a label in $L \cup L'$. With some thought, it is possible to realize the following. First, this label indexing of clauses is performing a job similar to the one played by parameter d, in the sense that the latter could in principle be dropped — and replacing one scheme by the other leaves the number of clauses in the exclusive component in the same order of magnitude. Second, this operation significantly changes the meaning of the clauses being used as the labels would then be reintroduced in a somewhat independent fashion. While the high degree of symmetry guarantees that all labels would be reintroduced, the new core is not an extension of the old one: we are not extending the original function by the introduction of auxiliary variables to reduce clause degrees through new local gadgets; we are changing the function being represented, and that qualifies the process as a new construction instead of a *cubification* of the old one. This immediately leads us to our third point: it is not clear how or whether the proofs we presented in Section 2.3 would extend to this new environment. Again with some thought, it is possible to see that the use of the label indexing scheme alone does not guarantee that the prime implicates involving variables $v(j)$ can have forms other than $v(j) \longrightarrow u(\ell)$, for $j \in [t]$ and $\ell \in L \cup L'$. Moreover, the shape of these prime implicates might depend on the ordering chosen for the edges of the graph and reintroducing the amplification device based on parameter d, playing the same role as before, does not ameliorate the situation (actually, it is completely useless). The difficulty in controlling the shape of those prime implicates tarnishes the tie established between clause minimum prime pure Horn representations and tight, optimal total-covers. It might still be possible to replace the clauses of type (d) in a linked-list fashion, but at this point is still not clear how to use such approach in a correct and not overly complicated way.

We shall circumvent the above problem through the use of a different structure: we shall replace the clauses of type (d) by new clauses arranged as *complete binary trees*, i.e., trees in which every level has all the nodes with the possible exception of the last level, where its nodes are flushed to the left; and we shall then link these trees together by their roots. The idea is as follows. For each index $i \in [d]$, we will introduce $m-1$ new variables and will arrange them as internal nodes in a complete binary tree, where the variables $e(x, y, i)$ will appear as tree leaves. Notice that we will have exactly d trees.

We will then associate each label from $L \cup L'$ with the roots of two of those trees, in an orderly fashion: the roots (namely, the variables $e(1,i)$) will be seen as nodes and the labels will be seen as edges of a path of length d. The path will be well defined (i.e., all labels will be reintroduced) if all nodes (root variables) are reachable through Forward Chaining from a variable $v(j)$. It is worth noticing that since $d = 1 + r\lambda + s\lambda' > |L \cup L'|$, all prime implicates involving the variables $v(j)$ will have the form $v(j) \longrightarrow u(\ell)$, for some label $\ell \in L \cup L'$ — similarly to what happend in the pure Horn CNF case, we shall show that it is simply not advantageous for these prime implicates to have any other form in clause minimum prime pure Horn 3-CNF representations. We now formalize this idea.

Let us rename the labels in $L \cup L'$ as ℓ_α, $\alpha \in [d-1]$. Let us also index the m edges in E as e_k, $k \in [m]$. Let us further introduce new propositional variables $e(k,i)$, $k \in [m-1]$, $i \in [d]$, and introduce $e(k,i)$ for $k \in \{m, m+1, \ldots, 2m-1\}$ to be an alias to the variable $e(x,y,i)$, where $(x,y) = e_{k-m+1}$ is an edge in E according to the indexing above. We then create d complete binary trees through the family of clauses:

$$(\mathrm{d}_1) \quad e(2k,i) \wedge e(2k+1,i) \longrightarrow e(k,i) \qquad \forall\, k \in [m-1],\ i \in [d];$$

$$(\mathrm{d}_2) \quad e(1,\alpha) \wedge e(1,\alpha+1) \longrightarrow u(\ell_\alpha) \qquad \forall\, \alpha \in [d-1].$$

Notice that clauses of type (d_1) index the nodes of the tree in a similar way a complete binary tree is stored inside an *array* (cf. Cormen et al. [47]) and that the clauses of type (d_2) do define the path we mentioned.

We shall illustrate the complete binary tree transformation through the following toy example. We start with a Label-Cover instance whose constraint graph is a claw, that is, $G = (\{x_1, x_2, x_3\}, \{y\}, \{(x_1,y), (x_2,y), (x_3,y)\})$, whose label sets are $L_0 = \{\ell_1, \ell_2\}$ and $L'_0 = \{\ell'_1, \ell'_2\}$, and whose constraint set is the union of

$$\Pi^0_{(x_1,y)} = \{(\ell_1,\ell'_1), (\ell_1,\ell'_2)\},\ \Pi^0_{(x_2,y)} = \{(\ell_1,\ell'_2)\},\ \text{and } \Pi^0_{(x_3,y)} = \{(\ell_2,\ell'_1), (\ell_2,\ell'_2)\}.$$

The refined Label-Cover instance will then have 8 labels in total ($L = \{\ell^1_1, \ell^1_2, \ell^2_1, \ell^2_2, \ell^3_1, \ell^3_2\}$ and $L' = \{\ell'_1, \ell'_2\}$)[1] and our CNF construction will introduce $8 \times 9 = 72$

[1]The refined label $(x_i, \ell_j) \in L$, with $x_i \in X$ and $\ell_j \in L_0$, is depicted by ℓ^i_j in this example.

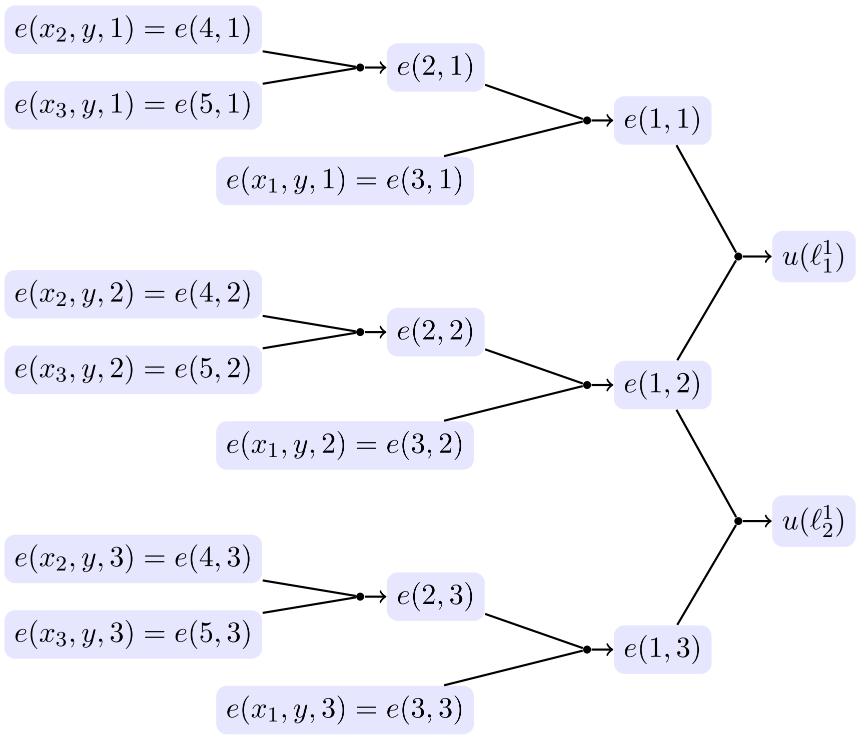

Figure 2.1: Partial depiction of the complete binary tree chain, which is responsible for reintroducing all the available labels of a Label-Cover instance in our 3-CNF construction, of a toy example where the constraint graph is a claw. In such example, the whole chain has nine trees and reintroduces eight different labels. Notice the complete symmetry between each pair of trees.

clause of type (d), nine of which have the following form

$$e(x_1, y, i) \wedge e(x_2, y, i) \wedge e(x_3, y, i) \longrightarrow u(\ell_1^1),$$

as $i \in [d] = [9]$. Clauses of type (d$_1$) replace those subgoals above by

$$e(4, i) \wedge e(5, i) \longrightarrow e(2, i) \qquad \text{and} \qquad e(2, i) \wedge e(3, i) \longrightarrow e(1, i), \tag{2.4}$$

where in this case, $e(x_1, y, i) = e(3, i)$, $e(x_2, y, i) = e(4, i)$, and $e(x_3, y, i) = e(5, i)$.

Finally, clauses of type (d$_2$) link the trees in (2.4) together, as e.g.

$$e(1, 1) \wedge e(1, 2) \longrightarrow u(\ell_1^1), \qquad e(1, 2) \wedge e(1, 3) \longrightarrow u(\ell_2^1), \qquad \text{and so on.}$$

A graphical illustration of part of the above transformation is provided in Figure 2.1.

Now, for clarity purposes, we present this new construction for the pure Horn 3-CNF case in full form below:

(a) $u(\ell) \wedge u(\ell') \longrightarrow e(x,y,\ell',i) \qquad \forall\ (x,y) \in E,\ (\ell,\ell') \in \Pi_{(x,y)},\ i \in [d];$

(b$_1$) $\bigwedge_{z \in N(y)} e(z,y,\ell',i) \longrightarrow e(x,y,i) \qquad \forall\ (x,y) \in E,\ \ell' \in L'_y,\ i \in [d],\ d(y) \leq 2;$

(b$_2$) $e(z_y^1,y,\ell',i) \wedge e(z_y^2,y,\ell',i) \longrightarrow e(1,x,y,\ell',i)$

$\forall\ (x,y) \in E,\ \ell' \in L'_y,\ i \in [d],\ d(y) \geq 3;$

(b$_3$) $e(z_y^{\beta+2},y,\ell',i) \wedge e(\beta,x,y,\ell',i) \longrightarrow e(\beta+1,x,y,\ell',i)$

$\forall\ (x,y) \in E,\ \ell' \in L'_y,\ i \in [d],\ \beta \in [d(y)-3];$

(b$_4$) $e(z_y^{d(y)},y,\ell',i) \wedge e(d(y)-2,x,y,\ell',i) \longrightarrow e(x,y,i)$

$\forall\ (x,y) \in E,\ \ell' \in L'_y,\ i \in [d],\ d(y) \geq 3;$

(c) $e(x,y,i) \longrightarrow e(x,y,\ell',i) \qquad \forall\ (x,y) \in E,\ \ell' \in L'_y,\ i \in [d];$

(d$_1$) $e(2k,i) \wedge e(2k+1,i) \longrightarrow e(k,i) \qquad \forall\ k \in [m-1],\ i \in [d];$

(d$_2$) $e(1,\alpha) \wedge e(1,\alpha+1) \longrightarrow u(\ell_\alpha) \qquad \forall\ \alpha \in [d-1];$

(e) $v(j) \longrightarrow u(\ell) \qquad \forall\ j \in [t],\ \ell \in L \cup L';$

where as before, $N(y) := \{x \in X : (x,y) \in E\}$ is the open neighborhood of the vertex $y \in Y$. Recall that $e(k,i)$ is an alias to a variable $e(x,y,i)$, $(x,y) = e_{k-m+1}$ being an edge in E, if $k \in \{m, m+1, \ldots, 2m-1\}$ and that it is a new variable used to build the i-th tree if $k \in [m-1]$.

Definition 2.39. *Let us call Ψ and Φ the* canonical *pure Horn 3-CNF formulae defined, respectively, by the families of clauses (a) through (d$_2$) and by all the families of clauses above. Let g and h be, in that order, the pure Horn functions they represent.*

2.4.1 Correctness of the 3-CNF Reduction

We now proceed to show the correctness of the ideas discussed above. Once again, we will constantly rely on the canonical representations Φ and Ψ to make inferences about the functions h and g they respectively define. And these inferences will most of the time be made throughout Forward Chaining. First, we present the new estimations for the number of clauses and variables in Φ and Ψ.

Lemma 2.40. *Let d and t be positive integers (amplification parameters) and let r, s, m, λ, λ', and π be as in Notation 2.16. We have the following relations for the number of clauses and variables in Φ, respectively:*

$$d(\pi + 2m\lambda' + 2m - 1) - 1 \leq |\Phi|_c - t(r\lambda + s\lambda') \leq d(\pi + m^2\lambda' + 4m)$$

and

$$t \leq |\Phi|_v \leq t + dm(\lambda' + 2) + m^2\lambda',$$

In Ψ, those numbers are, respectively,

$$d(\pi + 2m\lambda' + 2m - 1) - 1 \leq |\Psi|_c \leq d(\pi + m^2\lambda' + 4m)$$

and

$$0 \leq |\Psi|_v \leq dm(\lambda' + 2) + m^2\lambda'.$$

Proof. Let $\#(\tilde{\mathrm{b}}) := \sum_{i\in[4]} \#(\mathrm{b}_i)$ and $\#(\tilde{\mathrm{d}}) := \sum_{i\in[2]} \#(\mathrm{d}_i)$, where $\#(\alpha)$ denotes the number of clauses of type (α) in Φ.

For each edge $(x, y) \in E$, there was a clause of type (b) whose subgoal had size equal to $d(y)$. Each of those clauses was either maintained in case $d(y) \leq 2$ (originating the new clauses (b_1)) or replaced by the $d(y) - 1$ new clauses (b_2), (b_3), and (b_4) in a linked-list fashion. This procedure results in

$$dm\lambda' \leq \#(\tilde{\mathrm{b}}) = d\lambda' \sum_{(x,y)\in E} (d(y) - 1) \leq dm(m-1)\lambda'.$$

The clauses of type (d) were replaced by clauses of type (d_1) and (d_2). The new clauses of type (d_1) describe d complete binary trees, each of which having $2m-1$ nodes and hence, height $\eta = 1 + \lfloor \log(2m-1) \rfloor \leq \lfloor \log 4m \rfloor$. Considering also the $d - 1$ new clauses of type (d_2), we then obtain that

$$d(2m-1) - 1 \leq \#(\tilde{\mathrm{d}}) = d - 1 + d\sum_{l=1}^{\eta-1} 2^l = d - 1 + d(2^\eta - 1) \leq 4dm.$$

The new gadgets introduces $d(m-1)$ new variables $e(k,i)$ (notice the aliases are not new variables) and at most $dm(m-1)\lambda'$ new variables $e(\beta,x,y,\ell',i)$. Now, the results follow by using the remaining estimates of Lemma 2.23 and by recalling that Φ and Ψ differ only on the clauses of type (e). $\square$

Similarly to the pure Horn CNF case, as long as the quantities d and t are polynomial in s, namely, the number of vertices in Y, the new construction of Φ from $\mathcal{L}$ can also be carried out in polynomial time in s or, in another way, in polynomial time in the number of variables of h (cf. Remark 2.19).

The same arguments used to prove Lemma 2.24 apply in this new setting as the differences introduced by the new clauses are of a local nature. Specifically, it is still immediate that no clause in Ψ has head outside V_g, the set of variables occurring in Ψ, and hence, that V_g is closed under Forward Chaining in Φ. Thus, the set $\mathcal{X}(V_g) := \{C \in \mathcal{I}(h) : \mathsf{Vars}(C) \subseteq V_g\}$ is still an exclusive family for h (cf. Lemma 2.10) and $g \equiv \Psi = \Phi \cap \mathcal{X}(V_g)$ is an $\mathcal{X}(V_g)$-exclusive component of h. We have just proved the following.

Lemma 2.41. *The new function g is an exclusive component of the new function h. Consequently, g can be analysed and minimized separately.* $\square$

It is not hard to see that the bounds provided by Lemma 2.25 are still valid in this new setting.

Lemma 2.42 (Analogue of Lemma 2.25)**.** *Let Θ be a clause minimum prime pure Horn CNF representation of g. We have $|\Psi|_c/(\lambda+\lambda') \leq |\Theta|_c \leq |\Psi|_c$.*

Proof. The upper bound is by construction. For the lower bound, observe that each variable of Ψ appears no more than $\lambda+\lambda'$ times as a head. As in any clause minimum representation of g they must appear as head at least once, the claim follows. $\square$

Furthermore, Lemmas 2.26 and 2.27 transfer in a rather verbatim fashion: the new clauses and variables do not disrupt any of the conclusions obtained. For convenience, we include them below.

Lemma 2.43 (Analogue of Lemma 2.26). *$F_h(\{v(j)\}) = V_g \cup \{v(j)\}$ for all indices $j \in [t]$.*

Proof. It is enough to show that $\{e(1,\alpha) : \alpha \in [d]\} \subseteq F_\Phi(\{v(j)\})$, for a fixed $j \in [t]$. The inclusion would be false if there existed a label $\ell'' \in L \cup L'$ such that $u(\ell'') \notin F_\Phi(\{v(j)\})$. As for every label $\ell \in L \cup L'$, $v(j) \longrightarrow u(\ell)$ is a clause in Φ, this cannot happen. Hence, the inclusion holds, implying the claim. □

Lemma 2.44 (Analogue of Lemma 2.27). *A variable $v(j)$, for some index $j \in [t]$, is never the head of an implicate of h. Moreover, every prime implicate of h involving $v(j)$ is quadratic.*

Proof. The first claim is straight forward as all implicates of h can be derived from Φ by resolution, and $v(j)$ is not the head of any clause of Φ. By Lemma 2.43, $v(j) \longrightarrow z$ is an implicate of h for all $z \in V_g$. Since h is a pure Horn function, the claim follows. □

In the CNF construction presented in the previous section, all clauses of type (d) had the same subgoal. As explained in the beginning of this section, those clauses' subgoals were potentially long and we had to replace them by d gadgets whose structure mimics those of complete binary trees. It is still true that if one label in $L \cup L'$ is reintroduced by a clause of type (d$_2$), then so are all the remaining others. Like before, the reason is the high degree of symmetry occurring inside the exclusive component of g. Nevertheless, as the 3-CNF construction is more involved, we shall still present below complete proofs for the analogues of Lemmas 2.28, 2.30, and 2.31.

Lemma 2.45 (Analogue of Lemma 2.28). *Let $d = 1 + r\lambda + s\lambda'$. In any clause minimum prime pure Horn 3-CNF representation of h, the prime implicates involving the variables $v(j)$ have the form $v(j) \longrightarrow u(\ell)$, for all indices $j \in [t]$, and for some labels $\ell \in L \cup L'$.*

Proof. Let $\Upsilon = \Theta \wedge \Gamma$ be a clause minimum prime pure Horn 3-CNF representation of h, with Θ being a clause minimum pure Horn 3-CNF representation of g. According to an analogue of Lemma 2.27, all prime implicates of h involving the variables $v(j)$ are quadratic. So, for all indices $j \in [t]$ and all indices $i \in [d]$ define the sets

$$\begin{aligned}
\Gamma_0^j &:= \Gamma \cap \{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\}, \\
\Gamma_i^j &:= \Gamma \cap \{v(j) \longrightarrow e(k,i), \\
&\qquad\qquad v(j) \longrightarrow e(x,y,\ell',i), \\
&\qquad\qquad v(j) \longrightarrow e(\beta,x,y,\ell',i) : k \in [2m-1], (x,y) \in E, \ell' \in L'_y, \beta \in [d(y)-2]\}.
\end{aligned}$$

Recall that $e(k,i)$ is an alias to $e(x,y,i)$ if $(x,y) = e_{k-m+1}$ and $k \in \{m, m+1, \ldots, 2m-1\}$. Our goal is to show that the chosen value for the parameter d forces all the sets Γ_i^j to be simultaneously empty and consequently, that all the prime implicates involving the variables $v(j)$ in clause minimum pure Horn CNF representations of h have the claimed form. We shall accomplish this in two steps.

Let $j \in [t]$. We first show that if a set $\Gamma_i^j \neq \emptyset$ for some index $i \in [d]$, then $\Gamma_i^j \neq \emptyset$ for all indices $i \in [d]$, simultaneously.

Claim 0. *It holds that*

$$e(k,i) \in F_\Upsilon(\{v(j)\}) \quad \textit{if and only if} \quad e(k,i') \in F_\Upsilon(\{v(j)\}),$$

for all indices $i, i' \in [d]$, with $i \neq i'$.

Proof. The families of clauses (a), (b$_1$), (b$_2$), (b$_3$), (b$_4$), and (c) are completely symmetric with respect to the indexing variable i and do not interfere with each other. That is, for $i_1 \neq i_2$, variables depending upon i_1 do not trigger clauses indexed by i_2 during Forward Chaining, and vice-versa. □

Claim 1. *All clauses of type (d$_2$) have two variables in their subgoals which are roots of different, but completely symmetric trees (the trees specified by the clauses of type (d$_1$)). Therefore, because of this symmetry and Claim (0), during the execution of the Forward Chaining procedure from $\{v(j)\}$, either all clauses of type (d$_2$) trigger simultaneously or none of them do. The reason for them not to trigger is the absence of variables $e(k,i)$, for some $k \in [2m-1]$ and all $i \in [d]$, in the Forward Chaining closure from $\{v(j)\}$, i.e, $e(k,i) \notin F_\Upsilon(\{v(j)\})$.*

Proof. A simple inspection of the families of clauses shows that Claim (1) holds. □

Now, for each index $i \in [d]$, let Ξ_i be the collection of clauses of types (a), (b$_1$), (b$_2$), (b$_3$), (b$_4$), and (c) that depend on i.

Claim 2. *It holds that*

$$e(k,i) \in F_{\Gamma_0^j \cup \Xi_i}(\{v(j)\}) \quad \textit{if and only if} \quad e(k,i') \in F_{\Gamma_0^j \cup \Xi_{i'}}(\{v(j)\}),$$

for all indices $i, i' \in [d]$, with $i \neq i'$.

Proof. Notice that the families of clauses (a), (b$_1$), (b$_2$), (b$_3$), (b$_4$), and (c) are completely symmetric with respect to the indexing variable i. Moreover, for $i_1 \neq i_2$, the clauses indexed by i_1 do not interfere with the clauses indexed by i_2 during an execution of the Forward Chaining procedure. In other words, variables depending upon i_1 do not trigger clauses indexed by i_2, and vice-versa. These two properties, symmetry and non interference, proves Claim (2). □

Claim 3. *If there is a variable $e(k,i)$, with $k \in [2m-1]$ and $i \in [d]$, such that*

$$e(k,i) \notin F_{\Gamma_0^j \cup (\bigcup_{i\in[d]} \Xi_i)}(\{v(j)\})$$

then

$$e(k,i) \notin F_{\Gamma_0^j \cup (\bigcup_{i\in[d]} \Xi_i) \cup (\bigcup_{i'\neq i} \Gamma_{i'}^j)}(\{v(j)\}).$$

Moreover, this implies that $\Gamma_i^j \neq \emptyset$.

Proof. The symmetry and non interference properties of families of clauses (a), (b), and (c) also justifies the first part of Claim (3). To see it, just notice that were the claim to be false, the prime implicates in $\Gamma_{i'}^j$ would be trigging clauses involving the variable $e(k,i)$ in an execution of the Forward Chaining procedure. Since $i' \neq i$, this cannot happen. The second part follows immediately from the validity of the first part together with the fact that Υ represents h. □

To finish the first step, notice that since Claim (3) is valid for any $i \in [d]$, Claim (2) implies that if $\Gamma_i^j \neq \emptyset$ for some index $i \in [d]$, then $\Gamma_i^j \neq \emptyset$ for all indices $i \in [d]$, simultaneously.

For the second step, suppose that $\Gamma_i^j \neq \emptyset$ for all indices $i \in [d]$. We then have that

$$\gamma := \sum_{i\in[d]} |\Gamma_i^j| \geq d = 1 + r\lambda + s\lambda' = 1 + |L \cup L'|,$$

that is, γ is strictly larger than the number of all available labels in $\mathcal{L}$. This implies that the following pure Horn CNF

$$\begin{aligned}\Delta_j &:= \left(\Upsilon \setminus \bigcup_{i\in d} \Gamma_i^j\right) \cup \left\{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\right\} \\ &= \Theta \cup \left(\left(\Gamma \setminus \bigcup_{i\in d} \Gamma_i^j\right) \cup \left\{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\right\}\right),\end{aligned}$$

has fewer clauses than Υ (or more precisely, it implies that $|\Delta_j|_c <= |\Upsilon|_c - 1$).

Now, since Θ is a (clause minimum) representation of the exclusive component g, and since the set of clauses $\{v(j) \longrightarrow u(\ell) : \ell \in L \cup L'\}$ makes all available labels reachable by Forward Chaining from $\{v(j)\}$, it follows that $F_{\Delta_j}(\{v(j)\}) = V_g \cup \{v(j)\}$. Furthermore, the change in clauses did not influence the Forward Chaining procedure from any other variable (other than $v(j)$), and thus $F_{\Delta_j}(\{w\}) = F_\Upsilon(\{w\})$ for all variables $w \neq v(j)$. Thus, an analogue of Lemma 2.26 implies that Δ_j is a representation of h.

We then have that Δ_j is a shorter representation for h, contradicting the optimality of Υ. Therefore, the sets $\Gamma_i^j = \emptyset$ for all indices $i \in [d]$. As the above arguments do not depend on any particular value of j, they can be repeated for all of them. □

Definition 2.46. *Let $d = 1 + r\lambda + s\lambda'$ and let Υ be a prime and irredundant pure Horn 3-CNF representation of h. For each $j \in [t]$, consider the set*

$$S_j = \left\{\ell \in L \cup L' : v(j) \longrightarrow u(\ell) \in \Upsilon\right\}$$

and define the function $f_j : X \to L, Y \to L'$ given by $f_j(x) = S_j \cap L_x$ for vertices $x \in X$ and $f_j(y) = S_j \cap L'_y$ for vertices $y \in Y$.

Lemma 2.47 (Analogue of Lemma 2.30)**.** *Let Υ be as in the above Definition. For all indices $j \in [t]$ and vertices $y \in Y$, it holds that $|f_j(y)| \leq 1$.*

Proof. Let Υ be as in Definition 2.46 and suppose indirectly that the claim is false, that is, there is an index $j \in [t]$ and a vertex $y \in Y$ such that $|f_j(y)| > 1$.

During the proof, recall that the chosen value for the parameter d implies, according to Lemma 2.45, that all prime implicates of Υ involving the variable $v(j)$ must have the form $v(j) \longrightarrow u(\ell)$, with $\ell \in L \cup L'$.

Let $\ell' \in f_j(y)$ and define the expression

$$\Upsilon' := \Upsilon \setminus \{v(j) \longrightarrow u(\ell')\}.$$

It is enough to show that $F_{\Upsilon'}(\{v(j)\}) = V_g \cup \{v(j)\}$, that is, that Υ' is also a representation of h (by an analogue of Lemma 2.26). Suppose that is not the case. Since Υ and Υ' differ only in the clause $v(j) \longrightarrow u(\ell')$, it must be the case that $u(\ell') \notin F_{\Upsilon'}(\{v(j)\})$. This happens if the clause of type (d_2) associated to $u(\ell')$ is not triggered. For this to occur, there must be an index $k \in [2m-1]$ and an index $i \in [d]$ such that $e(k,i) \notin F_{\Upsilon'}(\{v(j)\})$.

Now, for y and i as above, notice that: (i) the variable $e(k,i)$ would be included in $F_{\Upsilon'}(\{v(j)\})$ as long as there were a label in L'_y such that the corresponding clause of type (b_1) or clauses of types (b_2), (b_3), and (b_4) were triggered; and (ii) once such clauses of type (b_1)–(b_4) were triggered, the appropriated clauses of type (c) would trigger, thus making the other clauses of type (b_1)–(b_4) associated to y and i to also trigger.

Therefore, for $e(k,i) = e(x,y,i)$ to not belong to $F_{\Upsilon'}(\{v(j)\})$, it must be the case that for every label $\ell'' \in f_j(y) \setminus \{\ell'\}$ there exists a vertex $z(\ell'') \in N(y)$ for which

$$e(z(\ell''), y, \ell'', i) \notin F_{\Upsilon'}(\{v(j)\}).$$

For this latter relation to be true, we must have that the clauses

$$u(\ell) \wedge u(\ell'') \longrightarrow e(z(\ell''), y, \ell'', i) \tag{2.5}$$

are not triggered in the Forward Chaining procedure on Υ' starting with $\{v(j)\}$, for every label $\ell \in f_j(z(\ell''))$ with $(\ell, \ell'') \in \Pi_{(z(\ell''),y)}$.

However, according to Definition 2.46, for each label $\ell'' \in f_j(y) \setminus \{\ell'\}$ and each label $\ell \in f_j(z(\ell''))$, there are clauses $v(j) \longrightarrow u(\ell'')$ and $v(j) \longrightarrow u(\ell)$, respectively, in Υ and, consequently, in Υ'. This implies that the clauses (2.5) are triggered, which implies that $u(\ell') \in F_{\Upsilon'}(\{v(j)\})$, which then implies that Υ' is also a representation of h. Since this contradicts the irredundancy of Υ, it follows that $|f_j(y)| \leq 1$, thus concluding the proof. $\square$

Lemma 2.48 (Analogue of Lemma 2.31). *Let Υ be a clause minimum prime pure Horn 3-CNF of h. Then it is prime and irredundant, so Definition 2.46 applies. We claim that for all indices $j \in [t]$ and vertices $y \in Y$, it holds that $|f_j(y)| \geq 1$.*

Proof. Suppose that the claim is false, that is, there is an index $j \in [t]$ and a vertex $y \in Y$ such that $|f_j(y)| = 0$.

Then clauses $v(j) \longrightarrow u(\ell')$, for all labels $\ell' \in L'_y$, are absent from Υ. Recall that the chosen value for the parameter d implies that all prime implicates of Υ involving $v(j)$ are quadratic (Lemma 2.45).

Thus, no clause of type (a) dependent on the vertex y is triggered during a Forward Chaining from $\{v(j)\}$ and hence, no clauses of type (b_1), (b_2), (b_3), (b_4), and (c) dependent on y are triggered either. This gives that the variables $e(k,i)$, for all indices $k \in \{m, m+1, \ldots, 2m-1\}$ such that $e_{k-m+1} = (x,y)$ (recall the indexing of the edges in E) and $x \in N(y)$, and all indices $i \in [d]$, do not belong to the Forward Chaining closure (from $\{v(j)\}$). This implies further that no variables $e(1,i)$ belong to the Forward Chaining closure. Therefore, no clause of type (d_2) is triggered and no label $\ell' \in L'_y$ is reintroduced. In other words, it is the case that $u(\ell') \notin F_\Upsilon(\{v(j)\})$ and hence, that $F_\Upsilon(\{v(j)\}) \neq F_h(\{v(j)\})$. By an analogue of Lemma 2.26, Υ does not represent h, a contradiction. So, it must be the case that $|f_j(y)| \geq 1$. $\square$

Once again, combining the two lemmas above, gives the following tight result.

Corollary 2.49. *Let Υ be a clause minimum prime pure Horn 3-CNF of h. Then, for all indices $j \in [t]$ and vertices $y \in Y$, it holds that $|f_j(y)| = 1$.* $\square$

Analogues of Lemmas 2.33, 2.34, and 2.35 can also be obtained in the same semi-verbatim fashion, and Remark 2.36 remains valid in this context. Hence, we still obtain the following.

Corollary 2.50. *Let Ψ be as in Definitions 2.39 and Υ be a clause minimum prime pure Horn 3-CNF of h. It holds that*

$$|\Psi|_c/(\lambda + \lambda') \le |\Upsilon|_c - t(\kappa(f)r + s) \le \ |\Psi|_c,$$

where $\kappa(f)r + s$ is the total number of labels in a tight optimal total-cover f for $\mathcal{L}$. □

We are also able to claim the following result.

Theorem 2.51. *Let c be a fixed constant close to $1/2$. Unless $\mathsf{P} = \mathsf{NP}$, the minimum number of clauses of a pure Horn function on n variables cannot be approximated in polynomial time (dependent on n) to within a factor of*

$$\rho_c(n^\varepsilon) \ge 2^{\varepsilon(\log n)^{1-1/\delta_c(n)}} = 2^{\log^{1-o(1)} n},$$

where $\delta_c(n) = (\log\log n)^c$, even when the input is restricted to 3-CNFs with $O(n^{1+2\varepsilon})$ clauses, for some $\varepsilon \in (0, 1/6]$.

Proof. The proof follows closely the one given for Theorem 2.38, just using the estimates provided by Lemma 2.40 instead of the ones in Lemma 2.23. □

2.4.2 Number of Literals

With the exception of the variables $v(j)$, with $j \in [t]$, that only appear as subgoals in quadratic prime implicates, every other variable appears in subgoals and heads of mostly cubic pure Horn clauses. The functions we are dealing with are pure Horn and therefore, have no unit clauses. So, it is the case that $2|\Phi|_c \le |\Phi|_l \le 3|\Phi|_c$ or in other words, that $|\Phi|_l = \Theta(|\Phi|_c)$, where Φ is a pure Horn 3-CNF formula obtained from our 3-CNF construction above. We then have the following result.

Corollary 2.52. *Unless $\mathsf{P} = \mathsf{NP}$, the minimum number of literals of a pure Horn function on n variables cannot be approximated in polynomial time (in n) to within a factor of $2^{\log^{1-o(1)} n}$, even when the input is restricted to 3-CNFs with $O(n^{1+2\varepsilon})$ clauses, for some $\varepsilon \in (0, 1/6]$.* □

2.5 Sub-exponential Time Hardness Results

The hardness of approximation results of Sections 2.3 and 2.4 apply to the scenario where the amount of available computational power is polynomial in n, the number of variables of a pure Horn function. In this section, we extend those results by showing that even when the computational power available is sub-exponential in n, it still not likely to be possible to obtain a constant factor approximation for such problems. The main ingredients of this section are: a stronger complexity theoretic hypothesis, a new Label-Cover result, and our pure Horn 3-CNF construction.

Recall that k-SAT is the problem of determining if a k-CNF formula, that is, one in which each and every clause has at most k literals, has a satisfying assignment of Boolean values to it variables. The following conjecture, called *Exponential Time Hypothesis (ETH)*, concerns the time solvability of the k-SAT problem and was introduced by Impagliazzo and Paturi [89].

Conjecture 2.53 (Impagliazzo and Paturi [89]). *For $k \geq 3$, define s_k to be the infimum of the set*

$$\Big\{\delta : \textit{there exists an } O\Big(2^{\delta n}\Big) \textit{ time algorithm for solving the k-SAT problem}\Big\},$$

with n being the number of variables of the k-SAT instance. The Exponential Time Hypothesis (ETH) *states that $s_k > 0$ for $k \geq 3$.*

In other words, if true, the ETH implies that there is no sub-exponential time algorithm for k-SAT with $k \geq 3$, what in turn implies that $\mathsf{P} \neq \mathsf{NP}$. The converse of this last implication however, does not hold and this establish ETH as a stronger hypothesis. It has many implications beyond search and optimization problems, e.g. in communication, proof, and structural complexity, and it is widely believed to be true.

In Section 2.2, we adressed the minimization flavor of the Label-Cover problem and briefly mentioned the existence of a maximization counterpart. Since the Label-Cover results we shall use in this section were originally obtained in the maximization setting, we introduce it below together with a "weak duality" type of result that binds both flavors.

Definition 2.54. *Let $\mathcal{L}_0 = (G, L_0, L'_0, \Pi_0)$ be a* Label-Cover *instance and f_0 be a labeling for it, as in Definitions 2.11 and 2.12, respectively. Let us call f_0* packing *if it assigns exactly one label per vertex of G. A packing labeling is* optimal *if it maximizes the fraction of covered edges of G. We denote this maximum fraction by $\mu(\mathcal{L}_0)$.*

Notice that differently from a total-cover, a packing labeling does not necessarily covers all the edges of the graph G and thus, $0 < \mu(\mathcal{L}_0) \leq 1$. The strict lower bound is due to non-empty relations of admissible pair of labels in Π_0 — in the maximization setting, these relations are also called *projections* as they can be interpreted (or redefined) as mappings $L_0 \longmapsto L'_0$.

In this section, we shall assume that the connected bipartite graph $G = (X, Y, E)$ is regular, that is, each vertex $x \in X$ has degree $d_X \geq 1$ and each vertex $y \in Y$ has degree $d_Y \geq 1$. This assumption is without loss of generality since Label-Cover instances can be regularized without significantly altering their sizes and promises (cf. Dinur and Harsha [54]). Moreover, the instances occurring in the hardness theorem we shall use in this section are regular.

Problem 2.55. *Let $0 < \xi < 1$ be any fixed constant. A* Label-Cover *instance $\mathcal{L}_0$ has* packing-promise ξ *if either $\mu(\mathcal{L}_0) = 1$ or $\mu(\mathcal{L}_0) \leq \xi$. That is, either all edges of $\mathcal{L}_0$ can be covered or at most a ξ fraction of them can. The* Label-Cover-Max$_\xi$ *problem is a promise problem which receives a* Label-Cover *instance with packing-promise ξ as input and correctly classify it in one of those two cases.*

The behavior of Label-Cover-Max$_\xi$ is left unspecified for non-promise instances and any answer is acceptable in that case. As before, the definitions above can be easily extended to refined Label-Cover instances.

The following result appears in Arora and Lund [10] and provides a link between the two flavors of the Label-Cover problem. In a nutshell, it implies that gap producing reductions from NP-complete problems like 3-SAT to the maximization flavor can be viewed as reductions to the minimization flavor as well. Hence, hardness of approximation results can be transfered from one flavor to the other. We decided to include its proof below for clarity reasons.

Lemma 2.56 (Arora and Lund [10]). *"Weak Duality:" for any (refined)* Label-Cover *instance $\mathcal{L}$, we have that*

$$\mu(\mathcal{L}) \geq \frac{1}{\kappa(\mathcal{L})}.$$

That is, the reciprocal of the value of an optimal packing labeling lower bounds the cost of an optimal (tight) total-cover. Furthermore, if $\mathcal{L}$ has packing-promise ξ, then it has covering-promise $\rho \geq 1/\xi$.

Proof. Let f be an optimal total-cover for $\mathcal{L}$ of cost $\kappa(\mathcal{L})$. That is, f covers all the edges of the graph $G = (X, Y, E)$ assigning $|f(z)|$ labels to each vertex of $z \in X \cup Y$. For simplicity reasons and without loss of generality, suppose that f is tight. Recall that by definition, $\kappa(\mathcal{L})$ is the average number of labels assigned by f to the vertices in X, namely,

$$\sum_{x \in X} |f(x)| = \kappa(\mathcal{L}) \cdot |X|. \tag{2.6}$$

Consider the following randomized procedure: for each vertex $x \in X$, pick a label at random in $f(x)$ and delete the remaining ones. Let f' be the resulting labeling. As $|f'(z)| = 1$ for all vertices $z \in X \cup Y$, f' is a packing labeling for $\mathcal{L}$ and the expected fraction of edges covered in f' is a lower bound for $\mu(\mathcal{L})$.

Let $\ell \in f(x)$ be a label used in f to cover an edge (x, y). The probability that $\ell \in f'(x)$, namely, that it survived the deletion process and ended up in f' is $1/|f(x)|$. The expected number of edges of G still covered in f' is then at least

$$\sum_{(x,y) \in E} \frac{1}{|f(x)|} = \sum_{x \in X} \frac{d_X}{|f(x)|} \geq d_X \frac{|X|^2}{\sum_{x \in X} |f(x)|} = d_X \frac{|X|^2}{\kappa(\mathcal{L}) \cdot |X|} = \frac{|E|}{\kappa(\mathcal{L})}, \tag{2.7}$$

where in the first and last equalities, we used the assumption that G is regular and thus, has $|E| = d_X \cdot |X|$ edges; in the inequality, we used the fact that $\sum_x 1/|f(x)|$ is minimized when the values $|f(x)|$ are all equal; and in the second to last equality we used Equation (2.6).

The above randomized procedure thus gives a packing labeling whose expected fraction of edges covered is at least $1/\kappa(\mathcal{L})$. So, there must exist a packing labeling attaining at least this fraction of edges covered and therefore, $\mu(\mathcal{L}) \geq 1/\kappa(\mathcal{L})$ as claimed. The relation on the promise bounds follows in a similar way from Equation (2.7). □

Moshkovitz and Raz [119], in a celebrated breakthrough, introduced a new two-query projection test Probabilistically Checkable Proof system with sub-constant error and quasi-linear size. Essentially, they started with a 3-SAT CNF instance ϕ of size (number of clauses) equal to σ and showed that it is NP-hard to solve $\textsc{Label-Cover-Max}_{1/\rho}$, for some $\rho = \rho(\sigma)$. Moreover, their reduction produces a graph of size at most $\sigma^{1+o(1)}$ and uses fixed label sets L and L' whose sizes depend on the value of the promise ρ. A simpler proof was later found by Dinur and Harsha [54] who started their reduction from a related but slightly different problem (satisfiability of circuits instead of formulas), brought different techniques to the mix, and formally stated the result as follows.

Theorem 2.57 (Moshkovitz and Raz [119], Dinur and Harsha [54])**.** *There exist constants $c > 0$ and $0 < \beta < 1$ such that for every function $1 < \rho(\sigma) \leq 2^{O(\log^\beta \sigma)}$, the following statement holds:*

There exists label sets L and L' of sizes $\exp(\rho(\sigma)^c)$ and $O(\rho(\sigma)^c)$, respectively, such that it is NP-hard to solve $\textsc{Label-Cover-Max}_{1/\rho(\sigma)}$ over these label sets. Furthermore, the size of the constraint graph of the $\textsc{Label-Cover}$ instance produced by this reduction is at most $\sigma \cdot 2^{O(\log^\beta \sigma)} \cdot (\rho(\sigma))^c = \sigma^{1+o(1)}$.

Notice that differently to what happens in the reduction of Theorem 2.18 where the label sets are instance dependent, in the above theorem the label sets are fixed and their sizes depend on the promise value (also called soundness value in this context). Also, the size of label set L is polynomial if $1 < \rho(\sigma) \leq \text{polylog}(\sigma)$ and super-polynomial if $\text{polylog}(\sigma) < \rho(\sigma) \leq 2^{O(\log^\beta \sigma)}$. It is worth mentioning that these differences do not affect our constructions (as they scale up appropriately), but do require changes in the calculations when showing our hardness results.

Using Lemma 2.56, we can apply the above hardness result to the minimization flavor of the $\textsc{Label-Cover}$ problem with promise $\rho(\sigma)$. While the resulting hardness factor is smaller than the one given by Dinur and Safra [55] in Theorem 2.18, the quasi-linear size of the constraint graph (and hence of the instance, as the label sets are fixed) allows the following:

Corollary 2.58 (Moshkovitz and Raz [119]). *Assuming the Exponential Time Hypothesis (cf. Conjecture 2.53, i.e., 3-SAT requires* $\exp(\Omega(\sigma))$ *time to be solved), quasi-polynomially sized instances of* Label-Cover-Max$_{1/\rho(\sigma)}$ *and of* Label-Cover$_{\rho(\sigma)}$ *cannot be solved in less than* $\exp\big(\sigma^{1-o(1)}\big)$ *time.* □

This can be used to rule out better approximations in sub-exponential time for other problems by further reductions from the Label-Cover problem.

Remark 2.59. *The new hardness result above brings along a new parametrization, which is slightly different from the one given in Remark 2.19. We now have that:* $s = \sigma^{1+o(1)}$ *with* $o(1) \approx (\log\log\sigma)^{-\Omega(1)}$, $r = \sigma 2^{O(\log^\beta \sigma)} = \sigma^{1+o(1)}$, $m = r\rho(\sigma)^c = \sigma^{1+o(1)}$, $\lambda = O\big(2^{\rho(\sigma)^c}\big)$, *and* $\lambda' = O(\rho(\sigma)^c)$.

Let $\vartheta > 0$ be such that $c\vartheta \leq 1$ and take $\rho(\sigma) = \log^\vartheta \sigma$. We then have that

$$\lambda = O\big(2^{(\log^\vartheta \sigma)^c}\big) = O(\sigma) \qquad \text{and} \qquad \lambda' = O\big((\log^\vartheta \sigma)^c\big) = O(\log\sigma),$$

and since $m \leq \pi \leq m\lambda\lambda'$, we also have that

$$2^{O(\log^\beta \sigma)}\sigma \log^{c\vartheta}\sigma \leq \pi = O\left(2^{O(\log^\beta \sigma)}\sigma^2 \log^2 \sigma\right).$$

Furthermore, it follows that

$$d = 1 + r\lambda + s\lambda' = \Theta\left(2^{O(\log^\beta \sigma)}\sigma^2 + \sigma^{1+o(1)}\log^{c\vartheta}\sigma\right). \tag{2.8}$$

Combining all the above, we obtain that as long as the (gap) amplification device t is polynomial in the number of variables of the canonical pure Horn 3-CNF Φ, the construction of Φ can still be carried out in polynomial time in this setting.

We are now ready to show a hardness of approximation result for the case when sub-exponential time is allowed. The proof closely follows the one presented for the polynomial time setting.

Theorem 2.60. *Assuming the ETH, the minimum number of clauses and literals of pure Horn functions in* n *variables cannot be approximated in* $\exp(n^\delta)$ *time, for some* $\delta \in (0,1)$, *to within factors of* $O(\log^\vartheta n)$ *for some* $\vartheta > 0$, *even when the input is restricted to 3-CNFs with* $O(n^{1+\varepsilon})$ *clauses and* $\varepsilon > 0$ *some small constant.*

Proof. Let $\mathcal{L}_0$ be a Label-Cover promise instances in compliance to Theorem 2.57 and let $\mathcal{L}$ be its refinement. Let d be as in Equation (2.8), Φ be the canonical pure Horn 3-CNF formula constructed from $\mathcal{L}$, and h be the pure Horn function it defines. Let Υ be a pure Horn 3-CNF representation of h obtained by some exact clause minimization algorithm when Φ is given as input.

For convenience, let $\rho = \rho(\sigma)$ and $\zeta = \zeta(\sigma) = r/s = 2^{O(\log^{\beta}\sigma)}/\sigma^{o(1)}$ and recall Notation 2.16. Substituting the quantities established in Lemma 2.40 into the bounds given by Corollary 2.50 and using and the new parametrization above (cf. Remark 2.59), we obtain that

$$\begin{aligned}|\Upsilon|_c &\geq t(\kappa(f)r + s) + \frac{d(\pi + 2m\lambda' + 2m - 1) - 1}{\lambda + \lambda'}\\ &\geq st(\kappa(f)r/s + 1) + \Omega\big(\sigma^{1+o(1)}2^{\log^{\tau}\sigma}\log^{2c\vartheta}\sigma\big)\\ &\geq st(\kappa(f)\zeta + 1) + \omega(s),\end{aligned}$$

and

$$\begin{aligned}|\Upsilon|_c &\leq t(\kappa(f)r + s) + d(\pi + m^2\lambda' + 4m)\\ &\leq st(\kappa(f)r/s + 1) + O\big(\sigma^4 2^{\log^{\tau}\sigma}\log^3\sigma\big)\\ &\leq st(\kappa(f)\zeta + 1) + o(s^5),\end{aligned}$$

where $\tau > 0$ is some constant. Similarly, for the number of variables we have that

$$\begin{aligned}t \leq |\Upsilon|_v &\leq t + dm(\lambda' + 2) + m^2\lambda'\\ &\leq t + O\big(\sigma^2 2^{\log^{2\tau}\sigma}\log^2\sigma\big)\\ &\leq t + o(s^3).\end{aligned}$$

Now, choosing $\varepsilon' > 0$ such that $t = s^{1/\varepsilon'} = \Omega(s^4)$ and supposing that $s \longrightarrow \infty$, we obtain the asymptotic expressions

$$|\Upsilon|_c = s^{(1+1/\varepsilon')}\zeta(\sigma)\kappa(f)(1 + o(1)) \qquad \text{and} \qquad |\Upsilon|_v = s^{1/\varepsilon'}(1 + o(1)).$$

Bringing the existing gaps of the Label-Cover promise instances into play, we then obtain the following dichotomy

$$\kappa(\mathcal{L}) = 1 \Longrightarrow |\Upsilon|_c \leq s^{(1+1/\varepsilon')}\zeta(\sigma)(1+o(1)),$$
$$\kappa(\mathcal{L}) \geq \rho(\sigma) \Longrightarrow |\Upsilon|_c \geq s^{(1+1/\varepsilon')}\zeta(\sigma)\rho(\sigma)(1+o(1)),$$

with $\rho(\sigma) = \log^{\vartheta}\sigma$ in this case. Let $n = |\Upsilon|_v$ and let $\varepsilon = \varepsilon'/(1+o(1))$. Relating the number of clauses of Υ to the number of variables of h, the above dichotomy reads as

$$\kappa(\mathcal{L}) = 1 \Longrightarrow |\Upsilon|_c \leq n^{(1+\varepsilon')}\zeta(n^{\varepsilon})(1+o(1)),$$
$$\kappa(\mathcal{L}) \geq \rho(\sigma) \Longrightarrow |\Upsilon|_c \geq n^{(1+\varepsilon')}\zeta(n^{\varepsilon})\rho(n^{\varepsilon})(1+o(1)),$$

giving a hardness of approximation factor of $\rho(n^{\varepsilon})$ for the pure Horn 3-CNF clause minimization problem (cf. Theorem 2.57). As ε^{ϑ} is a constant, it follows that the gap

$$\rho(n^{\varepsilon}) = \log^{\vartheta} n^{\epsilon} = O(\log^{\vartheta} n).$$

Also, the number of clauses $n^{(1+\varepsilon')}\zeta(n^{\varepsilon}) \leq n^{1+2\varepsilon'}$.

To conclude the clause minimization part, observe that sub-exponential time in σ, namely, $2^{o(\sigma)}$ is equivalent to $2^{o\left(n^{1/4-o(1)}\right)}$ time in n and since $|\Upsilon|_c = O(n^{1+2\varepsilon'})$, the time bound follows for $\delta < (1-2\varepsilon')/4$.

Regarding literal minimization, the structure of Φ implies its numbers of clauses and literals differ by only a constant (cf. Section 2.4) and therefore, similar results hold in this case as well. $\square$

A natural next step consists in trying to push the hardness of approximation factor further by allowing super-polynomially sized constructions, that is, canonical formulae Φ whose number of clauses is super-polynomial in σ.

So let $b > 1$ be such that $\alpha = bc > 1$ and take $\rho(\sigma) = \log^{\alpha}\sigma$. We then have that

$$\lambda = O\big(2^{\log^{\alpha}\sigma}\big) = O\big(\sigma^{\log^{\alpha-1}\sigma}\big) \qquad \text{and} \qquad \lambda' = O(\log^{\alpha}\sigma).$$

Also, we have that

$$\pi = O\Big(2^{\log^{\alpha}\sigma + O(\log^{\beta}\sigma)}\sigma\log^{\alpha}\sigma\Big),$$

and that we should choose the value of the parameter d such that

$$d = \Theta\left(\sigma 2^{\log^{\alpha}\sigma + O(\log^{\beta}\sigma)}\right).$$

Notice the above values imply that Φ is now super-polynomially sized in σ.

Following the steps of the proof of Theorem 2.60, we obtain that

$$\omega(s^2) \leq |\Upsilon'|_c - t(\kappa(f)r + s) \leq o\left(\sigma^{4+\log^{\alpha-1}\sigma}\right)$$

and that

$$t \leq |\Upsilon'|_v \leq t + o\left(\sigma^{3+\log^{\alpha-1}\sigma}\right).$$

Now, choosing $t = (8\sigma)^{\log^{\alpha-1} 8\sigma}$ gives a hardness of approximation factor for pure Horn clause minimization equal to $\rho(\sigma)$. Writing σ as a function of n, we obtain that $\sigma = (2^{\log^{1/\alpha} n})/8$ and hence, a hardness of approximation factor

$$\rho(n) = \log^{\alpha}\left(\frac{2^{\sqrt[\alpha]{\log n}}}{8}\right) = \left(\sqrt[\alpha]{\log n} - 3\right)^{\alpha} = O(\log n).$$

As $|\Upsilon'|_c = O(n^2)$, sub-exponential time $2^{o(\sigma)}$ means

$$\mu(n) := 2^{o\left(\left(2^{\sqrt[\alpha]{\log n}}/8\right)^{1/2}\right)} = o\left(2^{n^{1/(\log\log n)^C}}\right)$$

time, for any (possibly large) constant $C \geq 1$. We then just proved the following theorem.

Theorem 2.61. *Assuming the ETH, the minimum number of clauses and literals of pure Horn functions in n variables cannot be approximated in $\mu(n)$ time to within factors of $O(\log n)$, even when the input is restricted to 3-CNFs with $O(n^2)$ clauses.*

Pushing the approach further, if we take $\rho(\sigma) = 2^{O(\log^{\beta}\sigma)}$ for some $0 < \beta < 1$, we have that

$$\lambda = 2^{2^{O(\log^{\beta}\sigma)}} \qquad \text{and} \qquad \lambda' = 2^{O(\log^{\beta}\sigma)},$$

that $m = \sigma 2^{O(\log^{\beta}\sigma)}$, and that

$$\pi = d = \sigma 2^{O(\log^{\beta}\sigma)} 2^{2^{O(\log^{\beta}\sigma)}},$$

implying that Φ is now sub-exponentially sized in σ. We then have that

$$\Omega\left(2^{2^{O(\log^{\beta}\sigma)}}\right) \leq \frac{|\Upsilon'|_c - t(\kappa(f)r + s)}{\sigma^2} \leq O\left(\sigma 2^{2^{O(\log^{\beta}\sigma)}}\right)$$

and that

$$t \leq |\Upsilon'|_v \leq t + O\Big(\sigma^2 2^{2^{O(\log^\beta \sigma)}}\Big).$$

Now, letting $n := |\Upsilon'|_v = t = 2^{2^{1+\gamma \log^\beta \sigma}}$, for some constant γ, it gives

$$\sigma = 2^{\left(\frac{1}{\gamma} \log \frac{\log n}{2}\right)^{1/\beta}}$$

and a hardness of approximation factor $\rho(n) = (\log n)/2$, i.e., a similar $O(\log n)$ hardness factor under far more stringent time constraints. This last result is then rendered obsolete by Theorems 2.51 and 2.61.

2.6 Concluding Remarks

In the last three sections, we showed improved hardness of approximation results for the problems of determining the minimum number of clauses and the minimum number of literals in prime pure Horn CNF and 3-CNF representations of pure Horn functions. In the polynomial time setting, we obtained a hardness of approximation factor of $2^{\log^{1-o(1)} n}$ and when sub-exponential computational time is available, we showed a factor of $O(\log^\beta n)$, where $\beta > 0$ is some small constant and n is the number of variables of the pure Horn function. All these results hold even when the input CNF or 3-CNF formula is nearly linear, namely, when its size is $O(n^{1+\varepsilon})$ for some small constant $\varepsilon > 0$. We also managed to obtain a factor of $O(\log n)$ under more stringent, albeit sub-exponential, time constraints even when the input formula has size $O(n^2)$. In the polynomial time setting, our results are conditional on the $\mathsf{P} \neq \mathsf{NP}$ hypothesis, and are conditional on the Exponential Time Hypothesis (ETH) in the sub-exponential time scenario.

A natural question at this point concerns the tightness of our results. As mentioned in the introduction, Hammer and Kogan [82] showed that for a pure Horn function in n variables, it is possible to approximate the minimum number of clauses and the minimum number of literals of a prime pure Horn CNF formula representing it to within factors of $n - 1$ and $\binom{n}{2}$, respectively.

Our results then leave a sub-exponential gap in the polynomial time setting, and we are not aware of the existence of any sub-exponential time approximation algorithm for those problems. Naturally, narrowing or closing these gaps is highly desirable. One direction consists in designing new approximation algorithms with improved approximation guarantees. This does not seem to be an easy task to accomplish, nevertheless.

Another direction consists in further strengthening our hardness results. Replacing our constructions by smaller (e.g. quasi-linear) ones will allow us to extend our sub-exponential hardness result to the scenario where $\exp(o(n))$ computational time is available — recall that our result is valid for $\exp(n^\delta)$ time, for some constant $0 < \delta < 1$, and that we conjecture that no constant approximation factor is possible in the $\exp(o(n))$ time scenario. Besides barely improving the constants in our polynomial time proofs, we believe this option has little, if anything else, to offer.

Improvements on two-query projection test, sub-constant error Probabilistically Checkable Proof (PCP) systems (cf. Arora and Safra [12], Dinur and Safra [55], Moshkovitz and Raz [119], and Dinur and Harsha [54]) might lead to larger gaps for the Label-Cover (as a promise) problem, and as long as the Label-Cover instances are polynomially sized, our constructions would immediately imply larger hardness of approximation results for those problem. Moreover, improvements on Label-Cover might also allow one to obtain hardness results when super-polynomial time is allowed, a case we left untreated.

As a third possibility, notice that it might be possible to start from a different (promise) problem and provide different constructions and different proofs. At the moment, it is not clear how to pursue this venue. However, we conjecture that it is possible to improve the hardness of approximation factor for clause and literal minimization of a Horn function in n variables to at least $O(n^\varepsilon)$, for some small $\varepsilon > 0$. This conjecture concerns the polynomial time setting. In favor of it, we point out the fact that Umans [144] showed that for general Boolean functions, the decision versions of clause and literal minimization problems are Σ_2^p-complete (namely, $\mathsf{NP}^{\mathsf{NP}}$-complete) and that it is Σ_2^p-hard to approximate such quantities to within factors of N^ε, where N is the size of the input (a low-degree polynomial in the number of variables of the function in question) and $\varepsilon > 0$ is some small constant.

3

Quadratizations of Pseudo-Boolean Functions

In this chapter, we study quadratizations of pseudo-Boolean functions: given two pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$ and $g : \{0,1\}^{n+m} \to \mathbb{R}$ in n and $n+m$ variables, respectively, we say that g is a *quadratization* of f in m auxiliary variables if

$$f(x) = \min_{y \in \{0,1\}^m} g(x,y) \quad \text{for all} \quad x \in \{0,1\}^n, \text{ and } g \text{ is quadratic.}$$

In addition, in case all quadratic terms of g involves at most one auxiliary variable, we say that the quadratization g is *y-linear.* Our main interest is in understanding some local and global properties of quadratizations, and in doing so, we shall exhibit some new procedures to quadratize a pseudo-Boolean function, lower and upper bounds on the number of auxiliary variables, and some preliminary characterizations of quadratizations of negative monomials.

In the recent past, several exact and heuristic techniques have been developed to address and solve very large unconstrained quadratic binary optimization problems, techniques which have proven themselves quite successful (see Section 1.2 for references and more details). However, no similarly efficient methods are available for the higher degree case, and the increasing usage and demand for solutions for these more complex models (for instance, researcher from the computer vision community have been experimenting with very large high-degree formulations of nonlinear unconstrained binary problems) makes quadratization a very attractive alternative, thus more than justifying the efforts in understanding the strengths, limits, and applicability of this class of transformations.

The chapter is organized as follows. We introduce some basic concepts and facts about pseudo-Boolean functions, pseudo-Boolean optimization, and quadratizations in Section 3.1. The main intuit being that of fixing notation and nomenclature. We then review the existing quadratization techniques in Section 3.2, while also providing some extra comments and consequences of our own. In Section 3.3, we provide a multiple split scheme, thus generalizing some of the termwise quadratizations presented in the previous section. In Section 3.4, we introduce the first aggregative approach in which common parts of multiple terms are quadratized together, all at once. We also describe a practical algorithm based on this technique and mention some good experimental results obtained in some computer vision problems. Section 3.5 deals with quadratizations of symmetric pseudo-Boolean functions, where we introduce our representation theorems and show upper bounds on the number of auxiliary variables to quadratize first any symmetric pseudo-Boolean function, and then some specific (and particularly) popular ones.

In Section 3.6, we keep in track with the global approach of previous section, studying procedures that quadratize the whole pseudo-Boolean function instead of doing it by pieces. We first present two approaches based on the minterm structure of the function that are able to improve the upper bound on the number of auxiliary variables needed from $O(n\, 2^n)$ to at most 2^n and $\frac{3}{8}2^n$, respectively. Then, we introduce the concept of *universal sets*, which are sets of Boolean functions such that every pseudo-Boolean function can be expressed in terms of quadratic products of such Boolean functions. We then show the existence of universal sets such that every pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ in n variables can be quadratized with at most $O(2^{n/2})$ auxiliary variables, and if f is a degree-d function, i.e., its multilinear polynomial representation has degree equal to d, then it can be quadratized with at most $O(n^{d/2})$ auxiliary variables. Afterwards, we introduce another combinatorial structure, based on Turán systems, that allow us to produce a y-linear quadratization of any pseudo-Boolean function in n variables with at most $O\big(\frac{2^n}{n} \cdot \log n\big)$ auxiliary variables.

In Section 3.7, we take the opposite approach and concern ourselves with the question: how many auxiliary variables are necessary to quadratize a pseudo-Boolean function in n variables, if we consider all possible quadratization schemes? In other words, we study lower bounds on the number of auxiliary variables. We are able to show that our upper bounds of the previous section based on universal sets are essentially tight, as we obtain $\Omega(2^{n/2})$ and $\Omega(n^{d/2})$ for general and degree-d pseudo-Boolean functions, respectively. We also show a lower bound of $\Omega\left(\frac{2^n}{n}\right)$ for y-linear quadratizations, leaving a small gap of $O(\log n)$ to our upper bound. We close this section presenting lower bounds for symmetric pseudo-Boolean functions. Specifically, we show that there exists symmetric pseudo-Boolean functions whose quadratizations and y-linear quadratizations require $\Omega(\sqrt{n})$ and $\Omega(n/\log n)$ auxiliary variables, respectively, and through different techniques, we also show a lower bound of $\Omega(\sqrt{n})$ for the parity function.

Section 3.8 introduces the polyhedral cone of quadratizations, a combinatorial / geometric object that we use to generate and study quadratizations of monomials. While our understanding of its structure (vertices, extremal rays, faces, and facets) is still limited at the present, we present arguments that hints on its central importance in better understanding quadratizations. In Section 3.9, we build upon the knowledge amassed by the experiments of the previous section and present a full characterization of quadratizations of negative monomials involving only one auxiliary variable. While, at first sight, this might seen slightly disappointing, a cautious inspection of the rather long proof reveals the full intricacy one must face in order to fully understand the operation of quadratizing pseudo-Boolean functions.

We present some final considerations and some open problems in Section 3.10, and a list of quadratizations for low-degree monomials, obtained through computer generation is presented in Section 3.11 as an appendix.

The results of this chapter have appeared in publications by Boros and Gruber [26], Fix, Gruber, Boros, and Zabih [63, 64], and in a pair of papers by Anthony, Boros, Crama, and Gruber [4, 5].

3.1 Preliminaries

The goal of this section is to briefly introduce some of the main concepts and notations we shall use latter on. More thorough coverage can be found in Boros and Hammer [29], Crama and Hammer [50], and in the references given along the way.

3.1.1 Pseudo-Boolean Functions

Let S be a finite set and let $2^S := \{T : T \subseteq S\}$ be the family of subsets of S. A *set function* is a mapping of the form $2^S \mapsto \mathbb{R}$, i.e., a mapping that associates a real number to each and every subset of S. The set S is called *ground set* in this context. Set functions have been present in the mathematical literature for more than a century, and in particular have attracted substantial attention and experienced great development in the last 60 years. A related class of mappings was introduced in the works of Peter L. Hammer in the 1960s (see the seminal book of Hammer and Rudeanu [79]).

Definition 3.1. *A pseudo-Boolean function $f(x) = f(x_1, x_2, \ldots, x_n)$ in n binary variables is a mapping $\{0,1\}^n \mapsto \mathbb{R}$, that is, it is a mapping that associates n-dimensional binary vectors in $\{0,1\}^n$ to real numbers.*

It is not hard to see that if $|S| = n$ (the set S has n elements) and one replaces the subsets of S by their characteristic vectors, then the set function on 2^S can be interpreted as a pseudo-Boolean function on $\{0,1\}^n$, where the *characteristic vector* of a subset $T \subseteq S$ is the vector $\mathbf{1}^T \in \{0,1\}^n$ defined as $\mathbf{1}^T_i = 1$ if $i \in T$, and $\mathbf{1}^T_i = 0$ otherwise.

Set functions are often considered as being specified by an oracle, or more specifically, by an algorithm capable of delivering their values for any subset of the given finite ground set. As is well known (see for instance, Hammer and Rudeanu [79], Boros and Hammer [29], and Crama and Hammer [51]), pseudo-Boolean functions can be represented as polynomials over their variables, and since $x_i^2 = x_i$ whenever $x_i \in \{0,1\}$, such polynomial is *multilinear*, i.e., all occurrences of variable x_i have exponent equal to one. Moreover, this representation is unique. Therefore, in contrast, pseudo-Boolean functions are frequently (in our case, always) specified through this closed algebraic representation, and our work will take advantage of such.

To see the existence of the representation, let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function in n variables and for each binary vector $a \in \{0,1\}^n$, define the *characteristic* *function* of a as

$$\chi_a(x) := \prod_{i:a_i=1} x_i \prod_{j:a_j=0} \overline{x}_j = \prod_{i:a_i=1} x_i \prod_{j:a_j=0} (1-x_j),$$

where $\overline{x} = 1-x$. It is not hard to see that $\chi_a(x) = 1$ if $x = a$ and $\chi_a(x) = 0$ otherwise. Hence, f can be written as

$$\begin{aligned} f(x) &= \sum_{a\in\{0,1\}^n} f(a)\chi_a(x) \\ &= \sum_{a\in\{0,1\}^n} f(a)\left(\prod_{i:a_i=1} x_i \prod_{j:a_j=0} \overline{x}_j\right) \qquad (3.1) \\ &= \sum_{a\in\{0,1\}^n} f(a)\left(\prod_{i:a_i=1} x_i \prod_{j:a_j=0} (1-x_j)\right). \qquad (3.2) \end{aligned}$$

Equation (3.1) is called the *minterm normal form* of f (cf. Crama and Hammer [51]). Applying distributivity in Equation (3.2) and aggregating common terms, we obtain the multilinear representation of f:

$$f(x) = \sum_{S\subseteq[n]} c_S \prod_{i\in S} x_i, \qquad (3.3)$$

where $[n] := \{1,2,\dots,n\}$, $\prod_{i\in\emptyset} x_i = 1$ as usual, and the coefficients $c_S \in \mathbb{R}$ can also be computed through the *Möbius inversion formula* (cf. Jukna [95, 96], van Lint and Wilson [145]):

$$c_S = \sum_{T\subseteq S} (-1)^{|S|-|T|} f\big(\mathbf{1}^T\big),$$

where as before, $\mathbf{1}^T \in \{0,1\}^n$ is the characteristic vector of T.

In order to show the uniqueness, let $\mathbb{R}^{2^n}$ denote de vector space over $\mathbb{R}$ of functions from $\{0,1\}^n$ to $\mathbb{R}$, i.e., of all pseudo-Boolean functions, and notice that for each $S \subseteq [n]$, the multilinear monomial $P_S(x) := \prod_{i\in S} x_i$ can be associated in a one-to-one and onto fashion to an element of $\mathbb{R}^{2^n}$: there are 2^n multilinear monomials and $\mathbb{R}^{2^n}$ has dimension 2^n. Now, Equation (3.3) gives us that any pseudo-Boolean function can be expressed as a linear combination of those monomials or, in other words, they span the whole vector space. Hence, the set of all multilinear monomials $\big\{P_S(x) : S \subseteq [n]\big\}$ is a basis for $\mathbb{R}^{2^n}$, and every $f \in \mathbb{R}^{2^n}$ has a unique representation over such basis (precisely the one given by Equation (3.3)).

The *degree* of a pseudo-Boolean function f is defined as the size of the largest set S with a nonzero coefficient in it multilinear polynomial representation (3.3):

$$\deg(f) := \min\big\{|S| : S \subseteq V,\ a_S \neq 0\big\}.$$

Clearly, the degree of a constant function is zero. We say that f is a *quadratic* (respectively *linear*) pseudo-Boolean function if $\deg(f) \leq 2$ (respectively $\deg(f) \leq 1$).

The set of Boolean functions, i.e., mappings from $\{0,1\}^n \mapsto \{0,1\}$ (as in Definition 2.1) form a subclass of pseudo-Boolean functions whose range is $\{0,1\}$ instead of $\mathbb{R}$. It will be convenient to denote by $\mathscr{F}_n$ and $\mathscr{B}_n$ the sets of pseudo-Boolean functions and Boolean functions (viewed as pseudo-Boolean functions) in n variables, respectively; $\mathscr{B}_n \subset \mathscr{F}_n$.

Sometimes it is convenient to interpret a pseudo-Boolean function as a *hypergraph*, i.e., a family of subsets of a common ground set. More specifically, if $f : \{0,1\}^n \to \mathbb{R}$ is a pseudo-Boolean function in n variables, we can associate to f the hypergraph $\mathscr{H}_f = ([n], \mathcal{H})$ in which

$$\mathcal{H} = \left\{ H \subseteq [n] : \prod_{i \in H} x_i \text{ is a term of } f \text{ with a nonzero coefficient} \right\}.$$

That is, the hyperedges H of $\mathscr{H}_f$ are the indices of the variables that appear on terms of f. When clear from context and no confusion arises, we will refer to $\mathscr{H}_f$ simply by its family of hyperedges $\mathcal{H}$.

3.1.2 Symmetric Pseudo-Boolean Functions

We will now introduce basic algebraic concepts since symmetric functions and its numerous generalizations and extensions are naturally defined using them. A thorough coverage of these concepts can be found, for instance, in the book of Cameron [38].

Let G be a set and $\star : G \times G \to G$ be a binary operation closed over G, that is, an operation that maps pairs of elements of G to an element of G. The pair $(G, \star)$ is called an *algebraic group* (or simply *group*) if: (i) the operation $\star$ is associative; (ii) there is a (unique) element e of G that works as identify for $\star$, that is, $e \star a = a \star e = a$ for all $a \in G$; and (iii) each element of G has an inverse under $\star$, i.e., for each $a \in G$

there is an $b \in G$ such that $a \star b = e$, with a and b not necessarily distinct. Notice that the operation $\star$ does not need to be commutative. A group $(H, \star_H)$ is a *subgroup* of a group $(G, \star_G)$ if $H \subseteq G$ and $\star_H$ is the restriction of $\star_G$ to the elements of H. When the operation is clear from context and there is no room for confusion, the group is usually identified with its set of elements.

A *permutation* over a (finite) set Ω is a bijective mapping from Ω to itself. Clearly, the composition of two permutations over Ω is also a permutation over it, the identity permutation, which takes an element of Ω to itself, is well defined, and each permutation has an inverse. Therefore, the set of permutations over Ω endowed with composition as binary operation forms a group called the (finite) *symmetric group* of Ω. When $\Omega = [n] := \{1, 2, \ldots, n\}$ for some integer $n > 0$, this gives rise to a group called the *symmetric group of degree* n, with $n!$ elements and denoted by:

$$\mathfrak{S}_n := \Big(\big\{\pi : [n] \to [n] : \pi \text{ is a permutation}\big\}, \circ\Big), \tag{3.4}$$

where $\circ$ denotes the composition of permutations over $[n]$.

Each permutation $\pi \in \mathfrak{S}_n$ can be interpreted as a permutation of the input coordinates of a binary vector $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. More formally, each permutation $\pi \in \mathfrak{S}_n$ induces a permutation $\widehat{\pi} : \{0,1\}^n \to \{0,1\}^n$ on the set of possible binary vectors and is given by

$$\widehat{\pi}(x) := \big(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}\big).$$

This gives us the ability, for each pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ and each permutation $\pi \in \mathfrak{S}_n$, to define the permutation of f according to π as $f^\pi(x) := f(\widehat{\pi}(x))$, for all $x \in \{0,1\}^n$.

Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function and let $\mathsf{Aut}(f) \subseteq \mathfrak{S}_n$ be the set permutations that leave the value of f unchanged, i.e.,

$$\mathsf{Aut}(f) := \big\{\pi \in \mathfrak{S}_n : f^\pi(x) = f(x) \text{ for all } x \in \{0,1\}^n\big\}. \tag{3.5}$$

It is not hard to verify that $\mathsf{Aut}(f)$ is also a group. Indeed it is a subgroup of $\mathfrak{S}_n$ and is called the *automorphism group* of f.

Definition 3.2. *A pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is* symmetric *if*

$$\mathsf{Aut}(f) = \mathfrak{S}_n.$$

In other words, a pseudo-Boolean function is symmetric if it is invariant under any permutation of the coordinates of its variables. This implies that the value of a symmetric pseudo-Boolean function depends only on the Hamming weigh (number of ones) of its input and hence, can have only $n+1$ different outputs. Therefore, each symmetric pseudo-Boolean function is a vector in a real-valued linear space of dimension $n+1$ — which is exponentially smaller than the 2^n dimensional space of all pseudo-Boolean functions. This observation can be used to provide a nice characterization of symmetric pseudo-Boolean functions.

Theorem 3.3 (Minsky and Papert [118]; see also Jukna [96])**.** *Let $f : \{0,1\}^n \to \mathbb{R}$ be a symmetric pseudo-Boolean function in n variables. There exists a univariate polynomial $p : \mathbb{R} \to \mathbb{R}$ of degree at most* $\deg(f)$ *such that*

$$f(x_1, x_2, \ldots, x_n) = p(x_1 + x_2 + \cdots + x_n) \quad \textit{for all} \quad x \in \{0,1\}^n.$$

Proof. Let $d = \deg(f)$ be the degree of f and for $k = 0, 1, \ldots, d$, let

$$P_k := \sum_{S \in \binom{[n]}{k}} \prod_{i \in S} x_i,$$

that is, P_k denotes the sum of all $\binom{n}{k}$ products of $|S| = k$ different variables.

Since f is symmetric, it can be show by induction on d that f can be written as

$$f(x) = c_0 + c_1 P_1(x) + c_2 P_2(x) + \cdots + c_d P_d(x),$$

with the coefficients c_i being real numbers.

Now, observe that for $x \in \{0,1\}^n$ with $z := x_1 + x_2 + \cdots + x_n$ ones, P_k is equal to

$$P_k(x) = \binom{z}{k} = \frac{z(z-1)\cdots(z-k+1)}{k!},$$

which is a polynomial in z of degree k. Therefore, the univariate polynomial p given by

$$p(z) := c_0 + c_1\binom{z}{1} + c_2\binom{z}{2} + \cdots + c_d\binom{z}{d}$$

has the desired property. $\square$

Despite being a more economical representation of symmetric pseudo-Boolean functions (consider for instance the *parity* Boolean function, which is one if and only if the Hamming weight of its input is odd: it has only $n+1$ terms in the representation above, while it has $2^n - 1$ terms in

$$\bigoplus\nolimits_n(x) = \sum_{\emptyset \neq S \subseteq [n]} (-2)^{|S|-1} \prod_{i \in S} x_i, \tag{3.6}$$

the multilinear polynomial provided by Equation (3.3)), it can still be of degree higher than 2. Nevertheless, we shall base upon this characterization in Section 3.5.1, where we will prove a quadratic representation theorem of symmetric pseudo-Boolean functions (albeit in slightly more variables).

The notion of symmetric pseudo-Boolean functions can be generalized to that of d-part symmetric, where d is a nonnegative integer.

Definition 3.4. *A pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ in n variables is d-*part symmetric*, with $d \in [n]$, if there are nonnegative integers $n_1 + n_2 + \cdots + n_d = n$ such that*

$$\mathsf{Aut}(f) \cong \mathfrak{S}_{n_1} \times \mathfrak{S}_{n_2} \times \cdots \times \mathfrak{S}_{n_d}.$$

Equivalently, f is a d-part symmetric pseudo-Boolean function if there is a partition $[n] = V_1 \dot\cup V_2 \dot\cup \cdots \dot\cup V_d$ such that f is invariant under any permutation of the coordinates in any part V_i (with $|V_i| = n_i$). Following Theorem 3.3, we still can say that f is d-part symmetric if there are d functions $k_i : \{0,1\}^n \to \big(\{0\} \cup [n_i]\big)$, each k_i depending only on the coordinates in V_i and returning the Hamming weight of those coordinates, i.e.

$$k_i(x) := \sum_{j \in V_i} x_j,$$

and a function $\widetilde{f} : \big(\{0\} \cup [n_i]\big)^d \to \mathbb{R}$ such that

$$f(x) = \widetilde{f}\big(k_1(x), k_2(x), \ldots, k_d(x)\big) \quad \text{for all} \quad x \in \{0,1\}^n.$$

Notice that every symmetric pseudo-Boolean function is 1-part symmetric, and that every pseudo-Boolean function can be seen as n-part symmetric.

3.1.3 Pseudo-Boolean Optimization and Quadratizations

As mentioned in the introduction, the problem of optimizing a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ in n variables, specified as a multilinear polynomial, and subject to no constraints is called *Pseudo-Boolean Optimization* (PBO) problem. Both its maximization and minimization flavors have been the focus of researchers along the years (see Section 1.2 for a (partial) historical account). In this dissertation, we are particularly interested in the latter, i.e.,

$$\min_{x\in\{0,1\}^n} \left\{ f(x) = \sum_{S\subseteq[n]} c_S \prod_{i\in S} x_i \right\}. \tag{3.7}$$

Various exact and heuristic techniques for solving Problem (3.7) were proposed in the literature, specially for the case in which the multilinear polynomial of f is quadratic:

$$\rho(f) := \min_{x\in\{0,1\}^n} \left\{ f(x) = c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{1\le i<j\le n} c_{ij} x_i x_j \right\}. \tag{3.8}$$

Possible reasons for the popularity of the quadratic case are that numerous optimization problems, including the well-known MAX-SAT and MAX-CUT problems, have natural formulations as quadratic pseudo-Boolean (QPBO) problems; Rosenberg's [127] result that the general Problem (3.7) can be recast into Problem (3.8)'s form at the expense of some additional, auxiliary variables (more about it in the next section, hereunder) — in spite of the fact QPBO problems are still NP-hard; and the fact that it has many applications in areas ranging from physics through chip design to computer vision; see e.g., the surveys of Boros and Hammer [29], of Kolmogorov and Rother [103], and of Roth and Black [128].

A pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ in n variables is called *submodular* if

$$f(x \vee y) + f(x \wedge y) \le f(x) + f(y) \quad \text{for all} \quad x, y \in \{0,1\}^n,$$

where $(x \vee y)_j = x_j \vee y_j$ and $(x \wedge y)_j = x_j \wedge y_j$ for all indices $j \in [n]$. Submodular functions play an important role in optimization, since Problem (3.7) which is NP-hard in general, is known to be polynomially solvable if f is submodular; see Grötschel, Lovász, and Schrijver [76], Iwata, Fleisher, and Fujishige [94], and Schrijver [133].

It is known (cf. Nemhauser, Wolsey, and Fischer [121]) that if f is a quadratic pseudo-Boolean function, then it is submodular if and only if all quadratic terms have nonpositive coefficients. Hammer [80] proposed a very simple, network flow based minimization algorithm that optimizes quadratic, submodular pseudo-Boolean functions. A similarly efficient characterization of submodularity for cubic pseudo-Boolean functions was also shown to exist by Billionnet and Minoux [20]. However, Crama [48] and Gallo and Simeone [72] independently showed that the problem of recognizing if a pseudo-Boolean function of degree 4 or higher has the submodularity property is co-NP-complete.

For the case in which f is not submodular, various relaxation schemes were proposed. Among those, for the quadratic case, figured *complementation*, *linearization*, and *minorization*, which were shown by Hammer, Hansen, and Simeone [78] to yield the same lower bound to $\rho(f)$. Such lower bound is since then known as the *roof dual* of f. We briefly describe those approaches below.

Let $f : \{0,1\}^n \to \mathbb{R}$ be a quadratic pseudo-Boolean function in n variables as in Problem (3.8), and let $\mathscr{L} := \{x_i, \overline{x}_i : i \in [n]\}$ be the set of $2n$ literals associated to the x-variables of f. A quadratic *posiform* representation of f is given by

$$\phi_f = a_0 + \sum_{u \in \mathscr{L}} a_u u + \sum_{\substack{u,v \in \mathscr{L} \\ u \neq v}} a_{uv} uv, \tag{3.9}$$

where $a_u \geq 0$ and $a_{uv} \geq 0$ for all $u, v \in \mathscr{L}$. The name posiform comes from the fact that every coefficient of ϕ_f, except the constant term a_0 is nonnegative. It is well known that contrary to multilinear polynomial representations, posiform representations of pseudo-Boolean functions are not unique. In fact, a quadratic pseudo-Boolean function may admit posiform representations whose degree is larger than 2 (cf. Boros and Hammer [29]).

It is not hard to see that the constant term of a posiform ϕ_f for f, say $C(\phi_f)$ is a lower bound on $\min_{x \in \{0,1\}^n} f(x)$. The *complementation* approach seeks, among all the quadratic posiform representations of f, for the best lower bound on the minimum of

f, a problem that can be formulated as

$$C_2(f) := \max\Big\{C(\phi_f) : \phi_f \text{ is a quadratic posiform representing } f\Big\},$$

and can be solved by the following linear programming problem:

$$C_2(f) = \max \quad c_0 - \sum_{j=1}^{n} a_{\overline{x}_j} - \sum_{1\le i<j\le n} a_{\overline{x}_i\overline{x}_j}$$

$$\text{subject to}$$

$$a_{x_j} - a_{\overline{x}_j} + \sum_{\substack{1\le i\le n\\ i\ne j}} (a_{\overline{x}_i x_j} - a_{\overline{x}_i\overline{x}_j}) = c_j \quad \text{for} \quad j = 1, 2, \ldots, n, \tag{3.10}$$

$$a_{x_j x_j} + a_{\overline{x}_j\overline{x}_j} - a_{x_i\overline{x}_j} - a_{\overline{x}_i x_j} = c_{ij} \quad \text{for} \quad 1 \le i < j \le n,$$

$$a_u, a_{uv} \ge 0 \quad \text{for} \quad u, v \in \mathscr{L},\ u \ne v.$$

The *linearization* approach was already discussed in the introduction (cf. Section 1.2) and consists in replacing quadratic products $x_i x_j$ by new, auxiliary variables y_{ij}, and enforcing equality $y_{ij} = x_i x_j$ for binary values through the use of linear constraints $0 \le y_{ij} \le x_1, x_2$ and $y_{ij} \ge x_1 + x_2 - 1$. Relaxing the binary constraints on the x-variables, we obtain the linearization bound $L_2(f)$ for f. In slightly more details, $L_2(f)$ can be computed by the linear program below:

$$L_2(f) = \min \quad c_0 + \sum_{j=1}^{n} c_j x_j + \sum_{1\le i<j\le n} c_{ij} y_{ij}$$

$$\text{subject to}$$

$$\left.\begin{array}{l} y_{ij} \ge x_i + x_j - 1 \\ y_{ij} \ge 0 \end{array}\right\} \quad \text{for} \quad 1 \le i < j \le n,\ c_{ij} > 0, \tag{3.11}$$

$$\left.\begin{array}{l} y_{ij} \le x_i \\ y_{ij} \le x_j \end{array}\right\} \quad \text{for} \quad 1 \le i < j \le n,\ c_{ij} < 0,$$

$$0 \le x_j \le 1 \quad \text{for} \quad j = 1, 2, \ldots, n.$$

The *minorization* approach seeks for "best ℓ_1-norm" linear minorants for each and every quadratic term of the pseudo-Boolean function f. The approach is termwise, that is, for each quadratic product $x_i x_j$ it looks for a linear function of the form $\alpha + \beta x_i + \gamma x_j$

which lower bounds x_ix_j and minimizes the sum of the gaps for all four possible binary substitutions of x_i and x_j. In other words, it looks for the solution of the problem (in variables α, β, and γ):

$$\min \quad \sum_{(x_i,x_j)\in\{0,1\}^2} (x_ix_j - \alpha - \beta x_i - \gamma x_j)$$

$$\text{subject to}$$

$$x_ix_j \geq \alpha + \beta x_i + \gamma x_j \quad \text{for all} \quad (x_i, x_j) \in \{0,1\}^2,$$

which can be seen to be of the form $-\alpha = \beta = \gamma = \lambda$, for any $0 \leq \lambda \leq 1$. Similarly, the best ℓ_1-norm linear minorants of $-x_ix_j$ are of the form $-\lambda x_i - (1-\lambda)x_j$, for any $0 \leq \lambda \leq 1$.

Taking the weighted sum of the best ℓ_1-norm linear minorants of the terms of f, using the coefficients of the terms as weights give us the minorization lower bound, $M_2(f)$, which can be computed by the linear program:

$$M_2(f) = \max \quad c_0 - \sum_{\substack{1\leq i<j\leq n\\ c_{ij}>0}} \lambda_{ij}c_{ij} + \sum_{j=1}^{n} z_j$$

$$\text{subject to}$$

$$c_j + \sum_{\substack{i\neq j\\ c_{ij}>0}} \lambda_{ij}c_{ij} + \sum_{\substack{1\leq i<j\\ c_{ij}<0}} c_{ij}(1-\lambda_{ij}) + \sum_{\substack{j<i\leq n\\ c_{ij}<0}} c_{ij}\lambda_{ij} \geq z_j, \tag{3.12}$$

$$0 \geq z_j \quad \text{for} \quad j = 1, 2, \ldots, n,$$

$$0 \leq \lambda_{ij} \leq 1 \quad \text{for} \quad 1 \leq i < j \leq n,\ c_{ij} \neq 0.$$

Theorem 3.5 (Hammer, Hansen, and Simeone [78])**.** *For any quadratic pseudo-Boolean function $f \in \mathscr{F}_2$, it holds that*

$$C_2(f) = L_2(f) = M_2(f) \leq \min_{x\in\{0,1\}^n} f(x).$$

Moreover, the equality of these lower bounds with the minimum of $f(x)$ can be tested in linear time by solving a 2-SAT problem. □

The proof of the above theorem is based on showing that formulations (3.10) and (3.12) are both equivalent with the dual of (3.11).

Perhaps more important than roof duality per se, is the second result of Hammer, Hansen, and Simeone [78], which provides persistence properties for quadratic pseudo-Boolean functions. There are two types of persistence: weak and strong.

Let $S \subseteq [n]$ be a subset of indices of variables, that is, $S = \{i_1, i_2, \dots, i_s\}$ with $i_1 < i_2 < \cdots < i_s$ and $s = |S|$. The *projection* of $x \in \{0,1\}^n$ into S is given by $x_S = (x_{i_1}, x_{i_2}, \dots, x_{i_s})$, and for $y = (y_{i_1}, y_{i_2}, \dots, y_{i_s}) \in \{0,1\}^s$, the *partial assignment* of y to x, denoted by $x^{\otimes y}$, is the vector given by

$$x_i^{\otimes y} = \begin{cases} x_i & \text{if } i \notin S, \\ y_i & \text{otherwise,} \end{cases}$$

for all $i \in [n]$.

Definition 3.6. *Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function in n variables, let $S \subseteq [n]$ be a subset of indices, and let $y \in \{0,1\}^{|S|}$ be a binary vector. We say that the* strong persistence *property holds for f at y if*

$$\textit{for all } x \in \operatorname{argmin}(f), \textit{ we have that } x_S = y.$$

We say that the weak persistence *property holds for f at y if*

$$\textit{for all } x \in \operatorname{argmin}(f), \textit{ we have that } x^{\otimes y} \in \operatorname{argmin}(f).$$

In other words, in strong persistence, the restriction of all minimizing vectors of f to S coincide with y; in weak persistence, switching the entries in S of any minimum vector of f by y results in a minimum vector of f.

The concept of persistence, weak and strong, has been studied in the literature by Nemhauser and Trotter [120], Picard and Queyranne [125], Adams, Lassiter, and Sherali [1], Bertsimas, Natarajan, and Teo [17], among others. Hammer, Hansen, Simeone [78] showed the following results.

Theorem 3.7 (Strong Persistence; Hammer, Hansen, Simeone [78])**.** *Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function and let ϕ_f be a posiform representing it, such that $C_2(f) = C(\phi_f)$. Then, ϕ_f has the property that if $a_u > 0$ for some literal $u \in \mathscr{L}$, then $u = 0$ in all binary vectors $x \in \operatorname{argmin}(f)$ minimizing f.* □

Theorem 3.8 (Weak Persistence; Hammer, Hansen, Simeone [78]). *Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function and let $\widetilde{x}$ be an optimal solution of the Linear Program* (3.11), *for which $\widetilde{x}_j = 1$ if $j \in O$, and $\widetilde{x}_j = 0$ if $j \in Z$, where O and Z are two disjoint subsets of indices. Then, for any minimizing vector $x^* \in \operatorname{argmin}(f)$ switching the components to $x^*_j = 1$ for $j \in O$ and $x^*_j = 0$ for $j \in Z$ will also yield a minimum of f.* □

Besides fixing some of the variables at their provably optimum value, persistency allows for decomposing the residual problem into variable disjoint smaller subproblems, and that makes those properties even more attractive in very large scale QPBO problems.

Boros and Hammer [28], and Boros, Hammer, Sun, and Tavares [30] proposed a maximum-flow based approach, christened as the *QUBO algorithm*, to compute the roof dual of Problem (3.8), and obtain all the strong persistences guaranteed by Theorem 3.7. A multitude of experimental evaluations were conducted by Tavares [143], who then claimed a good performance of QUBO on sparse instances: it managed to fix the astonishing number of 100% of the variables through strong persistence in various planar MAX-2-SAT instances, a still NP-hard variation of the MAX-2-SAT problem in which the associated graph is planar.

The same approach was later recoded inside the computer vision community by Boykov and Kolmogorov [32] — among other changes, they used a bidirectional variant of Dinic's algorithm (see Ahuja, Magnanti, and Orlin [2], Cook, Cunningham, Pulleyblank, and Schrijver[46], Korte and Vygen [104]) for computing maximum flows, thus obtaining good speed ups. Boykov and Kolmogorov's implementation is called the *QPBO algorithm* and is freely available for download. Further details regarding applicability and performance can be obtained in Kolmogorov and Rother [102] and in Blake, Kohli, and Rother [21]. It is worth mentioning here that both versions of the flow-based algorithm were found very effective in computer vision problems, where they can frequently fix up to 80-90% of the variables at their provably optimum value, and both versions also returns automatically a minimizing solution for submodular inputs.

It is worth mentioning that in general, it is a hard task, even in the approximative sense to maximize the number of variables fixed by persistence. Following on the footsteps of Kumar and Sivakumar [106], Feige, Langberg and Nissim [59] showed that for many of the well known NP-complete problems (like 3-SAT, CLIQUE, 3-COLORING, SET COVER, MAX-CUT) it is NP-hard to produce a solution whose Hamming distance from an optimal solution is substantially closer than the one obtained by just taking a random solution. For instance, they showed that for an instance Ψ of 3-SAT, it is NP-hard to compute an assignment x for Ψ that agrees with any satisfying assignment x^* of Ψ in at least $n/2 + n^{1-\varepsilon}$ of the n variables, for some small,fixed constant $\varepsilon > 0$. Guruswami and Rudra [77] improved that bound to $n/2 + n^{2/3+\varepsilon}$, and Sheldon and Young [134] showed that the bound can be pushed down to $n/2 + n^{\varepsilon}$, that is, no deterministic polynomial-time algorithm achieve Hamming distance less that $n/2 + n^{\varepsilon}$ unless $\mathsf{P} = \mathsf{NP}$. Their bound also holds in the randomized setting: for any positive ε and c, no randomized polynomial-time algorithm achieves Hamming distance less that $n/2 - n^{\varepsilon}$ with probability $1/2 + 1/n^c$ unless $\mathsf{RP} = \mathsf{NP}$. Sheldon and Young [134] also showed the existence of a "universal" problem inside NP such that the deterministic bound is hard to approximate to within $n/2 + O(\sqrt{n \log n})$.

The result of Theorem 3.5 was extended to higher degrees by Boros, Crama, and Hammer [24], who showed:

Theorem 3.9 (Boros, Crama, and Hammer [24]). *Given a quadratic pseudo-Boolean function f in n variables, the equalities*

$$C_k(f) = L_k(f) = M_k(f)$$

hold for all $k = 2, 3, \ldots, n$, providing increasingly tighter lower bounds on f, with

$$\min_{x \in \{0,1\}^n} f(x) = C_n(f) = L_n(f) = M_n(f).$$

$\square$

Contrary to the relative easiness of testing the sharpness of the $C_2(f)$ bound, testing if $C_3(f)$ is sharp turns out to be NP-complete, and the linear programming formulations associated to the bounds of Theorem 3.9 do not provide any persistences: neither weak nor strong. See [24] for details and more info.

Also, extending the flow-based technique to compute these higher degree bounds turned out to be much harder than initially thought. Only recently, Wang and Kleinberg [148] proposed a multi-commodity flow based algorithm to compute $C_3(f)$. Unfortunately, however, their approach also does not provide any persistences.

Many researchers, specially inside the computer vision community decided then to revisit an approach initially proposed by Rosenberg [127]: to quadratize pseudo-Boolean functions. We have already encountered this concept in the introduction (cf. Section 1.2) and at the opening of this chapter. Nevertheless, for convenience, we shall repeat its definition below.

Definition 3.10. *Let $f(x) = f(x_1, x_2, \ldots, x_n)$ be a pseudo-Boolean function on $\{0,1\}^n$. We say that a pseudo-Boolean function $g(x, y)$ is a* quadratization *of $f(x)$ if $g(x, y)$ is a quadratic multilinear polynomial depending on x and on m* auxiliary *binary variables $y_1, y_2, \ldots, y_m$, such that*

$$f(x) = \min\big\{g(x, y) : y \in \{0,1\}^m\big\} \quad \textit{for all} \quad x \in \{0,1\}^n. \tag{3.13}$$

Clearly, if $g(x, y)$ is a quadratization of $f(x)$, then

$$\min\big\{f(x) : x \in \{0,1\}^n\big\} = \min\big\{g(x, y) : x \in \{0,1\}^n, y \in \{0,1\}^m\big\},$$

so that the minimization of $f(x)$ is reduced through this transformation to the QPBO problem of minimizing $g(x, y)$.

Definition 3.11. *A quadratic pseudo-Boolean function $g(x, y)$ on $\{0,1\}^{n+m}$ is called y*-linear *if its polynomial representation does not contain monomials of the form $y_i y_j$ for $i, j \in [m]$, $i \neq j$.*

When $g(x, y)$ is y-linear, it can be written as $g(x, y) = q(x) + \sum_{i=1}^m \ell_i(x) y_i$, where $q(x)$ is quadratic in x and each ℓ_i is a linear function of x. Then, when minimizing g over y, each product $\ell_i(x) y_i$ simply takes the value $\min\{0, \ell_i(x)\}$, that is,

$$f(x) = \min_{y \in \{0,1\}^m} g(x, y) = q(x) + \sum_{i=1}^m \min\{0, \ell_i(x)\}. \tag{3.14}$$

Hence, a y-linear quadratization of $f(x)$ produces an alternative representation of f in the x-variables only. It is also worth noticing that y-linear quadratizations can be viewed as piecewise linear functions of the x-variables, apart from the q part.

While Rosenberg's original method did not quite work in practice, due to the presence of very high penalty-terms in the quadratized function, many different techniques have been developed in the past 10 years, with some of those (including some of ours) meeting considerable success: the fraction of variables fixed by persistence in some very large high-degree problems, originating from computer vision applications reaches an astonishing value of 96% (cf. Fix, Gruber, Boros, and Zabih [63, 64]). We shall cover this story in the next section.

3.2 A Review of Existing Quadratization Techniques

In this section, we recall some quadratization techniques that have appeared in the literature previous to our work. This is due to their historical significance and the influence they have exerted in some of our results. While knitting this historical account, we shall also mention some observations and minor results of our own that would not properly fit elsewhere.

3.2.1 Rosenberg's Substitution

The first quadratization technique was introduced by Rosenberg [127] in 1975, and is based on the traditional idea of penalty functions. This method replaces a product xz of two binary variables by a new binary variable y (and hence decreases the degree by one of all terms involving both x and z) such that

$$p(x, z, y) \begin{cases} = \ 0 & \text{if } y = xz, \\ \geq \ 1 & \text{otherwise.} \end{cases} \tag{3.15}$$

Let $f : \{0, 1\}^n \to \mathbb{R}$ be a pseudo-Boolean function. Since f is multilinear, it can be written as $f = xzA + B$, where A is a multilinear polynomial not involving x and z, and where B is a multilinear polynomial not involving the product xz. Assume now that p is a quadratic function satisfying Equation (3.15), and that M is a positive real with $M > \max |A|$, where the maximization is taken over all binary assignments of the variables of A. Then, the function $\widetilde{f} = yA + B + Mp$ on $n+1$ variables has the same minima as f.

Theorem 3.12 (Rosenberg [127])**.** *The quadratic function* $p : \{0,1\}^3 \to \{0,1,3\}$ *given by*

$$p(x,z,y) = xz - 2xz - 2zy + 3y \tag{3.16}$$

satisfies Equation (3.15)*.*

Proof. If $y = xz$, replacing y in Equation (3.16) gives $p(x,z,y) = xz-2xz-2xz+3xz = 0$. If $y < xz$, then $y = 0$ and $xz = 1$, which readily implies that $p(x,z,y) = 1$. Finally, if $y > xz$, we have that $y = 1$ and $xz = 0$, thus leading to $p(x,z,y) = -2x-2z+3 \geq 1$. □

The above idea can be applied recursively until the resulting function $\widetilde{f}$ becomes quadratic. It is not hard to see that this is a polynomial transformation in the size of the representation of f, that is, its number of variables plus number of terms. If one wants a measure of the number of auxiliary variables in terms of the number of original ones, we have that no more than $O(n^d)$ new, auxiliary variables are needed, where d is the degree of f. Clearly, such quantity is $O(n^n)$ in the worst case. It easily comes to mind the idea of trying to find an appropriate order of substitution of variables so that the multiple applications of Theorem 3.12 resulted in the minimum number of auxiliary variables. Rosenberg [127] also showed that this problem, of finding a quadratization with the minimum number of auxiliary variables with his approach is itself an NP-hard problem. To see this let us consider the cubic pseudo-Boolean function

$$f(x_0, x_1, ..., x_n) = \sum_{(i,j)\in E} x_0 x_i x_j$$

where E is the edge set of a graph G on vertex set $V = [n]$. It is not hard to verify that for any quadratization of f, there is one with no more auxiliary variables, in which we substitute by auxiliary variables only products of the form $x_0 x_i$, $i \in C$ for a subset $C \subseteq V$ and the "optimal" quadratization corresponds to a minimum size vertex cover C of G. As the vertex cover problem on non-bipartite graphs is a classic NP-hard problem (cf. Garey and Johnson [73]), this completes the argument. In a nutshell, we can say that a sharp, general upper bound on the number of auxiliary variables in Rosenberg's approach will remains elusive, unless $\mathsf{P} = \mathsf{NP}$ (but of course, it can be computed exactly for specific functions or even particular classes).

From a practical standpoint, the drawback in Rosenbergs' approach is that the resulting quadratic function has many "large" coefficients, due to the recursive application of the "big M" substitution. It also introduces many positive quadratic terms, even if the input f is a nice submodular function. These two effects make the minimization of the resulting $\widetilde{f}$ a hard problem, even in approximative sense. Some experiments were conducted by Ishikawa [91, 92], who reported that Rosenberg's reduction performs very poorly on the functions appearing in some computer vision applications, as the QBPO method finds almost no persistencies.

In trying to address these drawbacks, one could try to replace the product $x_1 x_2 \cdots x_k$ of several variables by a new variable y and enforce the equality $x_1 x_2 \cdots x_k = y$ by a quadratic penalty function. Unfortunately, it is not hard to see that we have the following:

Remark 3.13. *There is no quadratic pseudo-Boolean function $p(x, z, w, y)$ in four binary variables such that $p(x, z, w, y) = 0$ if $y = xzw$, and $p(x, z, w, y) \geq 1$ whenever $y \neq xzw$.*

In other words, Rosenberg's approach is not possible with more than two variables. Before closing this subsection, let us mention that while investigating quadratizations of monomials, we discovered the following simple, Rosenberg-like transformation.

Proposition 3.14. *Let $f(x) = A(x)x_1x_2$ be a pseudo-Boolean function, where $A(x) = A(x_3, \ldots, x_n)$ does not depend on x_1, x_2, and for some $M_1 \geq 0$, $M_2 \geq 0$, assume that $-M_1 \leq A(x) \leq M_2$ for all $x \in \{0,1\}^n$. If*

$$g(x, y) = (A + M_1)y + (M_1 + M_2)(x_1 - y)(x_2 - y) - M_1 x_1 x_2,$$

then $f(x) = \min_y g(x, y)$ for all $x \in \{0,1\}^n$.

Proof. It is not hard to check, by enumerating the values of x_1, x_2, that the minimum of g is always attained when $y = x_1x_2$. Hence, we have that $g(x, y) = g(x, x_1x_2) = f(x)$, which in turn implies that $f(x) = \min_y g(x, y)$ for all $x \in \{0, 1\}^n$. $\square$

As in Rosenberg's original substitution, this transformation introduces only one additional variable and decreases the degree of f by one unit, as long as f is at least cubic, and it can be used recursively to quadratize any pseudo-Boolean function. Notice that when $A(x)$ is nonnegative or nonpositive, the formula simplifies accordingly by setting either M_1 or M_2 to 0. A nice feature of this transformation is that when $A(x)$ is negative, we get a submodular quadratization of f — and we already observed that this never happens with Rosenberg's substitution.

We can then use this transformation to provide upper bounds in the number of auxiliary variables for the very special cases in which the pseudo-Boolean function has only three or four variables, as we show below.

Proposition 3.15. *Every pseudo-Boolean function in $\mathscr{F}_3$ can be quadratized using at most one additional variable.*

Proof. Write $f(x) = ax_1x_2x_3+q(x)$, where $q(x)$ is quadratic, and apply Proposition 3.14 to the cubic term. $\square$

Proposition 3.16. *Every pseudo-Boolean function in $\mathscr{F}_4$ can be quadratized using at most two additional variables.*

Proof. Write $f(x) = Ax_1x_2 + Bx_1 + Cx_2 + D$, where the functions A, B, C, D do not depend on x_1, x_2. Apply Proposition 3.14 to Ax_1x_2. The resulting function $g(x, y)$ has no terms of degree higher than 3, and every cubic term of g necessarily contains the product x_3x_4. Hence, applying again Proposition 3.14 to the terms containing x_3x_4 results in a quadratization of f. $\square$

Notice that similar results can be obtained with Rosenberg's original substitution.

3.2.2 Freedman and Drineas' Higher-Degree Substitution

Kolmogorov and Zabih [103] introduced a simple quadratization for degree-3 negative monomials, which was extended by Freedman and Drineas [69] for arbitrary degree.

Theorem 3.17 (Freedman and Drineas [69]). *For binary variables $x_1, x_2, \ldots, x_d$ and y, it holds that*

$$-x_1x_2\cdots x_d = \min_{y\in\{0,1\}} y\left((d-1)-\sum_{j=1}^{d} x_j\right). \tag{3.17}$$

Proof. We claim that $y = \prod_{j=1}^{d} x_j$ at a minimum, in which case the left and right hand sides are identical. To see this claim, observe that we have the inequalities

$$-x_1x_2\cdots x_d \le 0,$$

and

$$-x_1x_2\cdots x_d \le (d-1)-\sum_{j=1}^{d} x_j.$$

Furthermore, these two right hand sides are the values of the right hand side of (3.17) corresponding to $y = 0$ and $y = 1$, respectively. Thus, $y = \prod_{j=1}^{d} x_j$ indeed achieves a value not larger than any of those. $\square$

The above proof also shows that such transformation can also be viewed as a substitution of a higher degree product, namely, $y = x_1x_2\cdots x_d$. It does, however, work in a rather different way than Rosenberg's method and so, does not contradict Remark 3.13.

A remarkable point is that it requires only one auxiliary variable to quadratize a negative monomial, i.e., the minimum amount possible. In fact, for reasons that we shall mention in Section 3.9, we also refer to this transformation as the *standard quadratization*, $s_d(x_1, x_2, \ldots, x_d, y)$, of negative monomials. Moreover, all quadratic terms in the right hand side of Equation (3.17) have negative coefficients. In particular, if $f : \{0,1\}^n \to \mathbb{R}$ is a degree-d pseudo-Boolean function involving only negative terms, then the application of this transformation to each of the t terms of f yields a submodular quadratization of it in t auxiliary variables with $O(td)$ negative quadratic terms (and of course no positive quadratic term).

Observe that the equality in Equation (3.17) is based only on the fact that the symbols x_i stand for a binary value. Hence, by introducing *negated literals* $\overline{x}_i = 1 - x_i$, the above transformation can be easily extended for positive monomials as well.

Proposition 3.18. *For binary variables $x_1, x_2, \ldots, x_d$ and $y_1, y_2, \ldots, y_{d-2}$, it holds that*

$$\begin{aligned} x_1 x_2 \cdots x_d - x_{d-1} x_d &= -\sum_{i=1}^{d-2} \overline{x}_i \prod_{j=i+1}^{d} x_j \\ &= \min_{y \in \{0,1\}^{d-2}} \sum_{i=1}^{d-2} y_i \left(d - i - \overline{x}_i - \sum_{j=i+1}^{d} x_j \right). \end{aligned} \tag{3.18}$$

Proof. Similar to that of Theorem 3.17. □

Hence Equation (3.17) also implies a quadratization of positive monomials, in which a degree-d positive monomial can be quadratized with $d-2$ auxiliary variables and results in $d-1$ positive (non-submodular) quadratic terms (and whose coefficients do not suffer from any increasing effect as in Rosenberg's method).

Let us add that if a subset of the variables are negated on the left in Equation (3.17) then the corresponding quadratic terms will have positive coefficients on the right hand side. In particular, if all variables are negated, then all quadratic terms have positive coefficients. Since the new variable y does not appear elsewhere we can replace it with its negation $\overline{y} = 1 - y$, not changing the minimization in this way, and get again a submodular quadratization.

Proposition 3.19. *For binary variables $x_1, x_2, \ldots, x_d$ and y, it holds that*

$$\begin{aligned} -\overline{x}_1 \overline{x}_2 \cdots \overline{x}_d &= \min_{y \in \{0,1\}} y \left((d-1) - \sum_{j=1}^{d} \overline{x}_j \right) \\ &= -1 + \sum_{j=1}^{d} x_j + \min_{\overline{y} \in \{0,1\}} \overline{y} \left(1 - \sum_{j=1}^{d} x_j \right). \end{aligned} \tag{3.19}$$

Proof. Again, similar to that of Theorem 3.17. □

Rother et al. [129] also observed that the transformation given in Equation (3.17) can be extended by using negated variables, and called it *type-II* transformation. They also noticed that one can apply Equation (3.17) to a subproduct (of a monomial), under some conditions. In particular, they quadratized separately the negated and unnegated variables in a monomial and derived a new transformation, which they called it *type-I*.

Theorem 3.20. *(Rother et al. [129]) Let S_0, S_1 be disjoint index sets. For binary variables $x_j \in S_0 \cup S_1$ and y_1, y_2, it holds that*

$$-\prod_{j\in S_0} \overline{x}_j \prod_{j\in S_1} x_j = \min_{y_1,y_2\in\{0,1\}} -y_1y_2 + y_1 \sum_{j\in S_0} x_j + y_2 \sum_{j\in S_1} \overline{x}_j, \tag{3.20}$$

Proof. See Section 4 of Rother et al. [129]. □

Notice that no matter which variation from above techniques we use to quadratize a degree-d positive monomial, we always need at least $d-2$ new variables. The situation will be improved to $\lfloor \frac{d-1}{2} \rfloor$ in the next subsection.

Before closing this subsection, let us mention an interesting consequence of Theorem 3.17 and Proposition 3.19. Let us call a pseudo-Boolean function f a *unary negaform* if it can be represented as a negative combination of terms involving either only unnegated variables, or only negated ones. It is easy to show that unary negaforms are submodular, and in fact Equations (3.17) and (3.19) provide a submodular quadratization for such functions.

Unary negaforms (more precisely, their negations) were considered by Billionnet and Minoux [20] and they showed that all cubic submodular functions can be represented by unary negaforms. They also provided a network flow model for the minimization of a unary negaform. The above submodular quadratizations given by Equations (3.17) and (3.19) also lead to a network flow based minimization by the results of Hammer [80] and these two network flow models are of very similar size (though they are not identical). Thus, the above observations can be viewed as a new simple proof for the results of Billionnet and Minoux [20].

Let us remark finally that higher degree submodular functions cannot typically be represented as unary negaforms. This is implied e.g., by the results of Živný et al. [151, 152] — which state that there are degree-4 submodular pseudo-Boolean functions that do not admit submodular quadratizations — since we just proved that a unary negaform always has a submodular quadratization.

3.2.3 Ishikawa's Higher-Order Clique Reduction

The Higher-Order Clique Reduction (HOCR) was introduced by Ishikawa [91, 92] and was the first practical transformation for high degree pseudo-Boolean functions. Notably, he provided a more compact quadratization for positive monomials, which uses only about half as many auxiliary variables as previous methods.

Theorem 3.21 (Ishikawa [91, 92])**.** *Consider the degree-d positive monomial $x_1 x_2 \cdots x_d$, set $k := \lfloor \frac{d-1}{2} \rfloor$, and consider new (auxiliary) binary variables $y = (y_1, y_2, ..., y_k)$. Define*

$$S_1 := \sum_{i=1}^{d} x_i, \quad S_2 := \sum_{1 \le i < j \le d} x_i x_j, \quad A := \sum_{j=1}^{k} y_j \quad \textit{and} \quad B := \sum_{j=1}^{k} (4j-1) y_j.$$

Then the following equalities hold:

$$\prod_{i=1}^{d} x_i = S_2 + \min_{y \in \{0,1\}^k} B - 2AS_1 \tag{3.21}$$

if $d = 2k+2$, and

$$\prod_{i=1}^{d} x_i = S_2 + \min_{y \in \{0,1\}^k} B - 2AS_1 + y_k (S_1 - d + 1) \tag{3.22}$$

if $d = 2k+1$.

Proof. Ishikawa's original proof can be found in [91, 92]. We shall provide an alternative proof in Section 3.5.3, Theorem 3.34. □

Let us note that S_1 and S_2 are symmetric functions of x, while A is a symmetric function of y. However, B is not a symmetric function of y. It is an interesting question on its own if one could find a quadratization of $\prod_{i=1}^{d} x_i$ which is symmetric in both x and y, and needs substantially fewer new variables than d.

In Fix et al. [63], we introduced a slight variation of the above equations:

$$\prod_{i=1}^{d} x_i = \sum_{1 \le i < j \le d} x_i x_j + \min_{y \in \{0,1\}^k} \sum_{j=1}^{k} y_j \left(c_{j,d} \left(-\sum_{i=1}^{d} x_i + 2j \right) - 1 \right), \tag{3.23}$$

where $c_{j,d} = 1$ if $d = j$ is odd, and $c_{j,d} = 2$ otherwise.

It follows from Equations (3.21) and (3.22), and also from Equation (3.23) that each degree-d positive monomial can be quadratized by these transformations with

$\lfloor \frac{d-1}{2} \rfloor = O(d)$ auxiliary variables. In spite of having linear asymptotic behavior, it is a practical improvement when compared to the methods of quadratizing positive monomials of the previous subsection: it requires roughly half the auxiliary variables. They also introduce $dk = O(d^2)$ negative quadratic terms.

Equations (3.21) and (3.22) introduce $\binom{d}{2}$ or $\binom{d+1}{2}$ positive quadratic terms, depending if d is even or odd, respectively, while Equation (3.23) is a bit more economical introducing always $\binom{d}{2}$ quadratic positive terms. Nevertheless, these quantities are all $O(d^2)$, implying that if W is the total weight of positive terms before the transformation, the quadratized function has a positive total weight of $O(d^2 W)$. To finish the analysis, it is important to notice that the same quadratic positive term can result from applying these transformations to different monomials. As they can be combined without loss (by summing their coefficients), it follows that if each variable occurs in positive monomials with at most ℓ variables, then the number of quadratic negative terms is $O(n\ell)$. This high number of positive quadratic terms makes the resulting quadratization highly non-submodular. According to Szeliski et al. [142], this is problematic as it can result is poor performance for graph-cut based methods like QPBO. However, despite of this negative feature, Ishikawa [91, 92] reported very good computational results, in particular when compared to the quadratization of Rosenberg [127].

3.3 Multiple Splits of Terms

We now show a generalization of some of the results presented in the previous section, introducing a general scheme to split a term into several fragments in order to decrease its maximum degree.

Let p, q be positive integers and assume that $\phi_i : \{0,1\}^p \to \{0,1\}$ are Boolean functions, for $i \in [q]$, satisfying the following conditions:

$$\begin{gathered} \min_{y \in \{0,1\}^p} \sum_{i=1}^{q} \phi_i(y) = 1, \quad \text{and} \\ \forall I \subset [q],\ \exists y_I \in \{0,1\}^p \quad \text{such that} \quad \sum_{i \in I} \phi_i(y_I) = 0. \end{gathered} \tag{3.24}$$

In other words, the sum of the ϕ functions have a positive minimum, but if we leave out any of the summands, the minimum becomes zero. For instance, for $p = 2$, $q = 3$ the functions $\phi_1 = y_1$, $\phi_2 = y_2$ and $\phi_3 = \overline{y}_1\overline{y}_2$ form such a set.

Theorem 3.22. *Let ϕ_i be Boolean functions satisfying condition* (3.24)*, and $P_i \subseteq [d]$ be subsets for $i \in [q]$ covering $[d]$. Then we have*

$$\prod_{j=1}^{d} x_j = \min_{y\in\{0,1\}^p} \sum_{i=1}^{q} \phi_i(y) \prod_{j\in P_i} x_j. \tag{3.25}$$

Proof. If $\prod_{j\in P_i} x_j = 1$ for all $i \in [q]$ then we have

$$1 = \min_{y\in\{0,1\}^p} \sum_{i=1}^{q} \phi_i(y) = 1$$

by (3.24). If there is an index $k \in [q]$ for which $\prod_{j\in P_k} x_j = 0$, then by (3.24) there exists a $y^* \in \{0,1\}^p$ such that $\phi_i(y^*) = 0$ for all $i \neq k$, and consequently we have $\phi_k(y^*) = 1$. Thus,

$$\begin{aligned} 0 &\leq \min_{y\in\{0,1\}^p} \sum_{i=1}^{q} \phi_i(y) \prod_{j\in P_i} x_j \\ &\leq \sum_{i=1}^{q} \phi_i(y^*) \prod_{j\in P_i} x_j = \prod_{j\in P_k} x_j = 0 \end{aligned}$$

follows, proving the claim. □

We now provide some remarks and examples:

- $\phi_1 = y_1$ and $\phi_2 = \overline{y}_1 = 1 - y_1$ provides a 2-split;
- $\phi_1 = y_1$, $\phi_2 = y_2$, and $\phi_3 = \overline{y}_1\overline{y}_2$ provides a 3-split;
- any binary tree of depth p with q leaves defines an appropriate system of ϕ_i functions, however not all systems correspond to such a tree (see e.g., the above 3-split);
- p variables can in general provide a $q \leq 2^p$-split transforming a degree d term to q terms of maximum degree $p + \lceil \frac{d}{q} \rceil$.

- a 2-split combined with Freedman and Drineas (3.17) yields the following quadratization of a cubic term

$$\begin{aligned} xyz &= \min_{u\in\{0,1\}} xu + \overline{u}yz \\ &= \min_{u\in\{0,1\}} xu + yz - uyz \\ &= \min_{u,v\in\{0,1\}} xu + yz + (2-y-u-z)v; \end{aligned}$$

- combining the above with a 2-split yields the following quadratization of a quartic term

$$\begin{aligned} txyz &= \min_{u\in\{0,1\}} txu + \overline{u}yz \\ &= \min_{u,v,w,s\in\{0,1\}} tv + xu + (2-v-x-u)w + yz + (2-u-y-z)s; \end{aligned}$$

another way of doing this is

$$\begin{aligned} txyz &= \min_{u\in\{0,1\}} tu + xyz - uxyz \\ &= \min_{u,v,w,s\in\{0,1\}} tu + xv + yz + (2-v-y-z)w + (3-u-x-y-z)s. \end{aligned}$$

It is worth noticing that in all of the above attempts to quadratize a positive degree d term, we had to include at least $d-1$ positive quadratic terms. We in fact conjecture that this is necessary.

3.4 Splitting of Common Parts

In this section, we introduce a new substitution scheme that has its roots in the idea of factorization of polynomials. Given a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, a subset of f's monomials may share a collection of variables whose indices belong to a set $C \subseteq [n]$. Instead of quadratizing or reducing the degree of each of those monomials individually, we replace the subproduct $\prod_{i\in C} x_i$ by a new binary variable y in all of them, altogether. The net effect is a simultaneous reduction on the degree of those monomials at the cost of only one auxiliary variable.

We show below how to apply this idea in groups of all-positive and all-negative monomials, as they behave differently. In describing these transformations, it will be convenient to interpret f as a hypergraph $\mathcal{H}$ over $[n]$ as described in Section 3.1.1.

For positive monomials, we have the following:

Theorem 3.23. *Let $C \subseteq [n]$, $\mathcal{H} \subseteq 2^{[n]\setminus C}$, and consider a fragment of a pseudo-Boolean function of the form*

$$\phi = \sum_{H\in\mathcal{H}} \alpha_H \prod_{j\in H\cup C} x_j,$$

where $\alpha_H \geq 0$ for all $H \in \mathcal{H}$. Then we have

$$\phi = \min_{y\in\{0,1\}} \left(\sum_{H\in\mathcal{H}} \alpha_H\right) y \prod_{j\in C} x_j + \sum_{H\in\mathcal{H}} \alpha_H \, \overline{y} \prod_{j\in H} x_j. \tag{3.26}$$

Proof. We claim that $y = \prod_{j\in C} x_j$ at a minimum, in which case the left and right hand sides are identical. To see this claim, observe that we have the inequalities

$$\phi \leq \sum_{H\in\mathcal{H}} \alpha_H \prod_{j\in H} x_j,$$

and

$$\phi \leq \sum_{H\in\mathcal{H}} \alpha_H \prod_{j\in C} x_j.$$

Furthermore, these two right hand sides are the values of the right hand side of (3.26) corresponding to $y = 0$ and $y = 1$, respectively. Thus, $y = \prod_{j\in C} x_j$ indeed achieves a value not larger than any of those. $\square$

As $\overline{y} = 1 - y$, the right hand side of Equation (3.26) is a multilinear polynomial replacing each positive monomial $H \in \mathcal{H}$ of degree $|H \cup C|$ by a positive monomial of degree $|H|$ and a negative monomial of degree $|H| + 1$, which involves the auxiliary variable y, and an additional positive monomial of degree $|C| + 1$, also involving y. It is easy to see that we have a decrease in degrees of positive monomials in ϕ whenever $0 < |C| < n - 1$. Also, notice that with exception of the additional positive monomial, whose coefficient is the sum of the coefficients in ϕ, all positive and negative monomials inherit in absolute value the coefficient of the monomial they are replacing, i.e., no large collection of "large" coefficients is created.

An interesting consequence of the above theorem is the following: using it recursively, we can find a quadratization $g(x, y)$ of any pseudo-Boolean function f (even if f is already quadratic) in which we have at most $n - 1$ positive quadratic monomials, where n is the number of variables of f. This shows that the difficulty of minimizing f does not come from an excessive number of non-submodular terms. In fact if we restrict our input to quadratic pseudo-Boolean functions in n variables and with at most $n - 1$ positive monomials, the minimization problems remains as hard as general quadratic minimization.

For negative monomials, we have the following:

Theorem 3.24. *Let $C \subseteq [n]$, $\mathcal{H} \subseteq 2^{[n]\setminus C}$, and consider a fragment of a pseudo-Boolean function of the form*

$$\phi = -\sum_{H\in\mathcal{H}} \alpha_H \prod_{j\in H\cup C} x_j,$$

where $\alpha_H \geq 0$ for all $H \in \mathcal{H}$. Then we have

$$\phi = \min_{y\in\{0,1\}} \sum_{H\in\mathcal{H}} \alpha_H \; y \left(1 - \prod_{j\in C} x_j - \prod_{j\in H} x_j\right). \tag{3.27}$$

Proof. We can prove, similarly to the previous proof that $y = \prod_{j\in C} x_j$ at a minimum. For this let us note that if $y = \prod_{j\in C} x_j$, then the right hand side in (3.27) is identical with ϕ, since $\prod_{j\in C} x_j \left(1 - \prod_{j\in C} x_j\right) = 0$ for all assignments $x \in \{0, 1\}^n$. Furthermore, we have the inequalities

$$\phi \leq 0,$$

and

$$\phi \leq \sum_{H\in\mathcal{H}} \alpha_H \left(1 - \prod_{j\in C} x_j - \prod_{j\in H} x_j\right),$$

where the right hand side values are the right hand side values of (3.27) corresponding to $y = 0$ and $y = 1$, respectively. Thus, again $y = \prod_{j\in C} x_j$ achieves the smallest possible value. □

Observe that all monomials in the right hand side of Equation (3.27) are also negative and have the same coefficient in absolute value of the negative monomial they are replacing. It is not hard to see that in order for this transformation to result in monomials of smaller degrees than the ones being replaced we must have the common part C satisfying $1 < |C| < n-1$. That is, if C is a singleton, Equation (3.27) replaces a negative monomial of degree $|H|$ by a slightly different one — involving the auxiliary variable y and not involving the variable in C — of same degree. Despite correct, this replacement would be of little usability. If $|C| > 1$, then a decrease of degrees is attained as desired.

Theorems 3.23 and 3.24 can be used together or separately (in this case, in conjunction with other quadratization techniques) to quadratize pseudo-Boolean functions in a recursive fashion. The possibilities abound, as in each step one has to decide the size of the common set C, which variables belong to C, and in what order should the theorems be applied.

3.4.1 Application: An Algorithm for Computer Vision Problems

In Fix et al. [63, 64], we propose an algorithm to reduce high-degree multilinear polynomials to quadratic ones through sequential application of Theorem 3.23 with $|C| = 1$ on positive monomials, and then using Freedman and Drineas' Theorem 3.17 to take care of negative monomials. More specifically, for $k = 1, 2, \ldots, x_n$ we let $C_k = \{x_k\}$ and let $\mathcal{H}_k$ be the subset of positive monomials that contain variable x_k. We then use Equation (3.26) specialized to this setting, i.e.,

$$\sum_{H\in\mathcal{H}_k} \alpha_H \prod_{j\in H\cup\{x_k\}} x_j = \min_{y\in\{0,1\}} \Bigg\{ \Bigg(\sum_{H\in\mathcal{H}_k} \alpha_H \Bigg) x_k y \tag{3.28a}$$

$$+ \sum_{H\in\mathcal{H}_k} \alpha_H \prod_{j\in H} x_j \tag{3.28b}$$

$$- \sum_{H\in\mathcal{H}_k} \alpha_H \, y \prod_{j\in H} x_j \Bigg\}, \tag{3.28c}$$

which gives that each positive monomial H in the left hand side of Equation (3.28) is replaced by: a quadratic positive monomial (3.28a) in x_k and y, an auxiliary variable; a positive monomial (3.28b) whose degree is one unit smaller than that of H, as it involves the same variables of H except x_k and does not involve y; and a negative monomial (3.28c) also not involving x_k, but involving y. This implies that at the end of the loop not only all positive monomials are quadratic but they are all y-linear. We then use Freedman and Drineas' transformation (3.17) to reduce one by one, in an arbitrary order, all the remaining high-degree negative monomials, thus obtaining the desired quadratization – however, the end product is not y-linear anymore.

The worst case of our algorithm happens when none of the positive monomials introduced by (3.28b) are already present in the original/current multilinear polynomial. That is, they are not being combined with previous monomials through the addition of their respective coefficients (what corresponds to a "pruning effect"), but are strictly growing the number of monomials in the polynomial in each step. When the pruning effect is observed, its performance improves considerably. We shall analyse its worst case first.

For a single degree-d positive monomial, we have that its complete quadratization through $d-1$ recursive applications of Equation (3.28), is similar to $d-1$ applications of the 2-split procedure (cf. Section 3.3) followed by the usage of $d-2$ Freedman and Drineas (3.17) technique to deal with the negative monomials. In more details, the recursive application of Equation (3.28) generates $d-1$ positive quadratic monomials, one linear positive monomial, $d-1$ negative monomials of degrees $2, 3, \ldots, d$, and introduces $d-1$ auxiliary variables along the way. Then, using Freedman and Drineas (3.17) we replace the $d-2$ of those negative monomials whose degrees are larger than 2 by $\binom{d}{2}-3 = O(d^2)$ new negative quadratic monomials, while introducing $d-2$ new auxiliary variables in doing so. Summarizing, we end up with d positive sub-cubic monomials, $O(d^2)$ negative monomials, and $2d-3 = O(d)$ required auxiliary variables.

For the case in which we have t degree-d positive monomials in n variables, the factorization of the common parts C_k shows some improvements, in the sense that we do not just multiply the quantities above by t. Specifically, the overall quadratization of t positive monomials requires $n + O(td)$ auxiliary variables, introduces $O(td^2)$ new quadratic negative monomials and at most n positive sub-cubic monomials.

Regarding the weights of the coefficients of the positive monomials (the non-submodular edge-weight), each positive monomial contributes α_H (in Equation (3.28a)) each time it is reduced, resulting in a total of $\alpha_H(d-1)$ non-submodular edge-weight. So, if the total weight of positive monomials is W before the transformation, we obtain $O(dW)$ after it.

When compared to Ishikawa's HORC method of Section 3.2.3, the worst case of our algorithm requires slightly more auxiliary variables ($n + O(td)$ instead of $O(td)$), introduces roughly the same amount of quadratic negative monomials ($O(td^2)$ for both), introduces less quadratic positive terms (n instead of $O(nk)$), and has a factor d improvement for non-submodular total edge-weigh ($O(dW)$ instead of $O(d^2W)$).

It was noticed by Alex Fix, one of our coauthors in Fix et al. [63, 64], that the pseudo-Boolean functions occurring in some common computer vision problems (cf. Woodford et al. [150], Roth and Black [128]) have the special structure in which "most" of the positive monomials introduced by (3.28b) were already present in the original multilinear polynomial, because the pseudo-Boolean functions of such problems are likely to have all monomials on all subsets of d variables, thus making the representation of the function essentially large. As it turns out, our algorithm (unlike Ishikawa's HORC or Kahl and Strandmark's Generalized Roof Duality (GRD)) performs asymptotically better on such instances.

The key idea behind the specialized analysis is that of local completeness.

Definition 3.25. *Let $\mathcal{H}$ be the hypergraph of a pseudo-Boolean function f, and let $\mathcal{H}^{\downarrow}$ be its* ideal*, i.e., $\mathcal{H}^{\downarrow} := \bigcup_{H \in \mathcal{H}} 2^H$. We say that f is* c-locally complete *if there exists a constant $0 < c \leq 1$ such that $|\mathcal{H}| \geq c|\mathcal{H}^{\downarrow}|$.*

It is easy to see that local completeness provides a lower bound on the size of the representation of the pseudo-Boolean function in question. The larger this lower bound is, more positive monomials introduced by (3.28b) are combined with existent monomials, and less negative monomials will be introduced by (3.28c) in subsequent steps of the loop, thus requiring less auxiliary variables in the second phase of our algorithm. In more details, it is possible to show the following.

Theorem 3.26 (Fix et al. [63, 64])**.** *Let f be a c-locally complete pseudo-Boolean function, and let $\mathcal{H}$ be its hypergraph interpretation. The first phase (reduction of positive monomials) of our algorithm generates at most $\frac{1}{c}|\mathcal{H}|$ negative monomials.*

Proof. See Fix et al. [63, 64]. □

Therefore, if f is a degree-d, c-locally complete pseudo-Boolean function in n variables and with t terms, our algorithm results in a quadratized pseudo-Boolean function with at most $n + \frac{1}{c}t$ auxiliary variables, $\frac{1}{c}td$ negative quadratic monomials, and n positive quadratic monomials.

A detailed description of the experimental results of the above algorithm can be checked in Fix et al. [63, 64]. In the larger context where its output was optimized by the QPBO algorithm of Kolmogorov and Rother [102], and the whole process was run a considerable number of times (depending on the computer vision problem and the image being processed), some remarkable observations are: (i) our algorithm runs faster than both HOCR and GRD methods; (ii) it produces more compact representations of the pseudo-Boolean functions; (iii) it finds more persistencies; and (iv) altering the ordering of the variables in the first phase loop does not produce significant different results, that is, the algorithm is robust.

Below, we reproduce two tables from our aforementioned paper that illustrates some of these comments. The first shows a comparison of Ishikawa's, Kahl and Strandmark's, and our methods, on benchmarks introduced by Ishikawa [91, 92], and averaged over 30 iterations of "fusion move" (a technique used to solve some computer vision problems). Relative performance is showed inside parenthesis.

	Energy improvement	Percent labeled by QPBO	Time (seconds)
HORC	$1,302$ (-45%)	59.4% (-22%)	14.1 $(+150\%)$
GRD-heur	$3,183$ $(+35\%)$	86.3% $(+13\%)$	40.2 $(+620\%)$
Our algorithm	$2,351$	76.1%	5.6

The second compares the total size of Ishikawa's transformation versus ours, on Ishikawa's benchmark as given in [91, 92]. The relative performance of our method appears inside parenthesis.

	Auxiliary variables	Positive terms	Total terms
HORC	$224,346$	$421,897$	$1,133,811$
Our algorithm	$236,806$ $(+6\%)$	$38,343$ (-90%)	$677,183$ (-40%)

3.5 Quadratizations of Symmetric Pseudo-Boolean Functions

In this section we investigate quadratizations of a certain prominent family of pseudo-Boolean functions: the class of symmetric pseudo-Boolean functions. Recall from Section 3.1.2 that symmetric pseudo-Boolean functions are the ones which are invariant under renaming of its variables or, in an equivalent way (cf. Theorem 3.3) depend exclusively on the Hamming weight (number of ones) of their inputs.

For most, but not all, complexity measures over Boolean and pseudo-Boolean functions one can find in the literature (see e.g. Junka [96]), symmetric functions are usually presented as objects having low complexity values for those measures. We show in this section that the same principle applies to quadratizations.

3.5.1 A Representation Theorem

Let a be any real number and let $[a]^- := \min(a, 0)$ denote the smaller of a and 0. In this subsection, we introduce a 'representation theorem' that expresses a symmetric pseudo-Boolean function on variables $x_1, x_2, \ldots, x_n$ as a linear combination of terms of the form $[a - \sum_{r=1}^n x_r]^-$, for a suitable range of values a. This result will be key in our approach to obtain quadratizations of symmetric pseudo-Boolean functions.

Our representation result, in its most general form, is as follows.

Theorem 3.27. *Let $0 < \epsilon_i \leq 1$, for $i = 0, 1, \ldots, n$. Then every symmetric pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ can be represented uniquely in the form*

$$f(x) = \sum_{i=0}^{n} \alpha_i \left[i - \epsilon_i - \sum_{r=1}^{n} x_r \right]^- .$$

Proof. When $\sum_{r=1}^{n} x_r = j$, we should have $f(x) = k_j$. So, to find a coefficient vector $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n)^T \in \mathbb{R}^{n+1}$ which establishes the required representation, we need to find a solution to the following system of $n+1$ linear equations:

$$k_j = \sum_{i=0}^{n} \alpha_i \left[i - \epsilon_i - j\right]^- = \sum_{i=0}^{j} \alpha_i \left(i - \epsilon_i - j\right), \quad \text{for } j = 0, 1, \ldots, n. \tag{3.29}$$

The matrix underlying this system is the lower-triangular matrix

$$\begin{aligned} A &= \left([q - \epsilon_q - p]^-\right)_{p,q=1,2,\ldots,n+1} \\ &= \begin{pmatrix} -\epsilon_0 & 0 & 0 & 0 & \cdots & 0 \\ -1-\epsilon_0 & -\epsilon_1 & 0 & 0 & \cdots & 0 \\ -2-\epsilon_0 & -1-\epsilon_1 & -\epsilon_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -n-\epsilon_0 & -n+1-\epsilon_1 & -n+2-\epsilon_2 & -n+3-\epsilon_3 & \cdots & -\epsilon_n \end{pmatrix} \end{aligned}$$

Because A is lower-triangular with nonzero diagonal entries $-\epsilon_q$, for $q = 0, 1, \ldots, n$, A is non-singular and this system does indeed have a unique solution. Therefore the representation exists and is unique. □

In case all the ϵ_i are equal, we can be more explicit about the coefficients in the representation. Recall that $k_l = k(l)$ is the value of $f(x)$ in any point x with Hamming weight equal to l. We set $k_{-1} = 0$ by convention.

Theorem 3.28. *Let $0 < \epsilon \leq 1$. Then every symmetric pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ can be represented uniquely in the form*

$$f(x) = \sum_{i=0}^{n} \alpha_i \left[i - \epsilon - \sum_{r=1}^{n} x_r\right]^-$$

where, for $j = 0, 1, \ldots, n$, the value of α_j is

$$\alpha_j = -\sum_{i=0}^{j-2} \frac{(\epsilon - 1)^{j-i-2}}{\epsilon^{j-i+1}} k_i + \left(\frac{1}{\epsilon} + \frac{1}{\epsilon^2}\right) k_{j-1} - \frac{1}{\epsilon} k_j. \tag{3.30}$$

As usual, the first sum above is equal to 0 if $j < 2$.

Proof. We show that Equation (3.30) is a solution (and hence *the* solution) of the System (3.29) whenever $\epsilon_i = \epsilon$, for all $i = 0, 1, \ldots, n$. The proof is by induction on j.

The case $j = 0$ is easily verified, since the first equation in (3.29) immediately yields $\alpha_0 = -\frac{1}{\epsilon}k_0$, in agreement with (3.30). Assume now that (3.30) is satisfied by the values of α_i up to $i = j - 1$. Then, from (3.29) and from the induction hypothesis,

$$\begin{aligned}
-\epsilon\alpha_j &= k_j + \sum_{i=0}^{j-1} \alpha_i (\epsilon + j - i) \\
&= k_j + \sum_{i=0}^{j-1} \alpha_i (\epsilon + j - 1 - i) + \sum_{l=0}^{j-1} \alpha_l \\
&= k_j - k_{j-1} + \sum_{l=0}^{j-1} \alpha_l. \qquad (3.31)
\end{aligned}$$

Substituting (3.30) in the last term of (3.31) yields

$$\begin{aligned}
\sum_{l=0}^{j-1} \alpha_l &= \sum_{l=0}^{j-1} \left[-\sum_{i=0}^{l-2} \frac{(\epsilon-1)^{l-i-2}}{\epsilon^{l-i+1}} k_i + \left(\frac{1}{\epsilon} + \frac{1}{\epsilon^2}\right) k_{l-1} - \frac{1}{\epsilon} k_l \right] \\
&= -\sum_{i=0}^{j-3} k_i \sum_{l=i+2}^{j-1} \frac{(\epsilon-1)^{l-i-2}}{\epsilon^{l-i+1}} + \left(\frac{1}{\epsilon} + \frac{1}{\epsilon^2}\right) \sum_{l=0}^{j-1} k_{l-1} - \frac{1}{\epsilon} \sum_{l=0}^{j-1} k_l \\
&= -\sum_{i=0}^{j-3} k_i \sum_{t=0}^{j-i-3} \frac{(\epsilon-1)^t}{\epsilon^{t+3}} - \frac{1}{\epsilon} k_{j-1} + \frac{1}{\epsilon^2} \sum_{l=0}^{j-1} k_{l-1} \qquad (3.32) \\
&= \sum_{i=0}^{j-3} \frac{(\epsilon-1)^{j-i-2}}{\epsilon^{j-i}} k_i - \frac{1}{\epsilon} k_{j-1} + \frac{1}{\epsilon^2} k_{j-2} \qquad (3.33)
\end{aligned}$$

where the last equality is obtained by summing the geometric series which appears in the first sum of equation (3.32).

Combining (3.31) and (3.33), we find

$$\alpha_j = -\sum_{i=0}^{j-3} \frac{(\epsilon-1)^{j-i-2}}{\epsilon^{j-i+1}} k_i - \frac{1}{\epsilon^3} k_{j-2} + \left(\frac{1}{\epsilon} + \frac{1}{\epsilon^2}\right) k_{j-1} - \frac{1}{\epsilon} k_j,$$

which is equivalent to (3.30). □

Two special cases of Theorem 3.28 deserve special attention, as we shall use them often. When $\epsilon = 1/2$, Theorem 3.28 yields:

Corollary 3.29. *Every symmetric pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ can be represented uniquely in the form*

$$f(x) = \sum_{i=0}^{n} \alpha_i \left[i - \frac{1}{2} - \sum_{r=1}^{n} x_r \right]^-$$

where

$$\alpha_i = -8 \sum_{j=0}^{i} (-1)^{i-j} k_j - 2k_{i-1} + 6k_i$$

for $i = 0, 1, \ldots, n$, and $k_{-1} = 0$. □

Taking $\epsilon = 1$ in Theorem 3.28, we obtain the following.

Corollary 3.30. *Every symmetric pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ can be represented in the form*

$$f(x) = k_0 + (k_1 - k_0) \sum_{r=1}^{n} x_r + \sum_{i=1}^{n-1} (-k_{i-1} + 2k_i - k_{i+1}) \left[i - \sum_{r=1}^{n} x_r \right]^- .$$

□

A simpler, more direct proof of Corollary 3.30 follows from work of Fix [62]. We include it here as it helps furthering understanding on the subject. As Fix observed, if $\sum_{r=1}^{n} x_r = l$, then

$$f(x) = k(l) = \sum_{i=0}^{n} k(i) \delta_i(l)$$

where $\delta_i(l) = 1$ if $i = l$ and $\delta_i(l) = 0$ otherwise. Then, it can be seen that

$$\delta_i(l) = -\left[i - 1 - l\right]^- + 2\left[i - l\right]^- - \left[i + 1 - l\right]^- .$$

From this, it follows that

$$f(x) = \sum_{i=0}^{n} k(i) \left(-\left[i - 1 - l\right]^- + 2\left[i - l\right]^- - \left[i + 1 - l\right]^- \right) .$$

On simplification, this gives

$$f(x) = k_0 + l(k_1 - k_0) + \sum_{i=1}^{n-1} (-k_{i-1} + 2k_i - k_{i+1}) \left[i - l\right]^- ,$$

as required.

3.5.2 From Representations to Quadratizations

In this subsection we explain how a representation of the type presented in the previous section can be used to construct quadratizations of pseudo-Boolean functions. One useful observation is that when a coefficient α_i is non-negative, the corresponding term $\alpha_i \left[i - \epsilon_i - \sum_{r=1}^{n} x_r\right]^-$ in the representation of Theorem 3.27 of f can be quadratized as $\min_{y_i} \alpha_i y_i (i - \epsilon_i - \sum_{r=1}^{n} x_r)$. But this translation simply does not work if α_i is negative. The strategy described in this section is to take an expression as given in Theorem 3.27 (or one of its special cases) and add a quantity that is identically-0 and which will result in a final expression that has no terms with negative coefficients. The following Lemma describes three possible such quantities. The first is going to be useful when working with representations of the form given in Corollary 3.30, and the second and third will be useful when working with the representations from Corollary 3.29.

Lemma 3.31. *Let*

$$E(l) = l(l-1) + 2\sum_{i=1}^{n-1} [i-l]^-,$$

$$E'(l) = \frac{l(l-1)}{2} + 2\sum_{\substack{i=2:\\ i \text{ even}}}^{n} \left[i - \frac{1}{2} - l\right]^-,$$

and

$$E''(l) = \frac{l(l+1)}{2} + 2\sum_{\substack{i=1:\\ i \text{ odd}}}^{n} \left[i - \frac{1}{2} - l\right]^-.$$

Then, for all $l = 0, 1, \ldots, n$, $E(l) = E'(l) = E''(l) = 0$.

Proof. First we show that $E(l)$ is identically-0. We have

$$\begin{aligned}
E(l) &= l(l-1) + 2\sum_{i=1}^{n-1} [i-l]^- \\
&= l(l-1) + 2\sum_{i=1}^{l-1} (i-l) \\
&= l(l-1) - 2\sum_{j=1}^{l-1} j \\
&= l(l-1) - l(l-1) = 0.
\end{aligned}$$

We next show that $E'(l) = 0$ for all values of l. Fix l and note that $i - \frac{1}{2} - l \leq 0$ if and only if $i \leq l$. Hence,

$$\sum_{\substack{i=2:\\ i \text{ even}}}^{n} \left[i - \frac{1}{2} - l\right]^{-} = \sum_{\substack{i=2:\\ i \text{ even}}}^{l} \left(i - \frac{1}{2} - l\right) = \sum_{\substack{i=2:\\ i \text{ even}}}^{l} i - \left(\frac{1}{2} + l\right) \left\lfloor \frac{l}{2} \right\rfloor . \qquad (3.34)$$

By considering separately the cases where l is respectively even or odd, one can conclude that $E'(l) = 0$ for all $l = 0, \ldots, n$. For, if $l = 2r$, then the expression on the right in equation (3.34) is $r/2 - r^2 = -l(l-1)/4$ and, if $l = 2r+1$, it is $-r/2 - r^2 = -l(l-1)/4$. The identity $E''(l) = 0$ (for all l) can be proved similarly, or can be deduced from the previous one by observing that, for all $l = 0, 1, \ldots, n$,

$$\sum_{i=1}^{n} \left[i - \frac{1}{2} - l\right]^{-} = \sum_{i=1}^{l} \left(i - \frac{1}{2} - l\right) = -\frac{1}{2} l^2.$$

We then can note that

$$E'(l) + E''(l) = l^2 + 2 \sum_{i=1}^{n} \left[i - \frac{1}{2} - l\right]^{-} = l^2 - l^2 = 0,$$

so that $E'' = -E' = 0$. □

We gave a direct, self-contained, proof of Lemma 3.31, but in fact these three identities follow from Corollary 3.29 and Corollary 3.30. For, if we apply Corollary 3.30 to the function

$$f(x) = \sum_{r=1}^{n} x_r \left(\sum_{r=1}^{n} x_r - 1\right),$$

we see that

$$f(x) = -2 \sum_{i=1}^{n-1} \left[i - \sum_{r=1}^{n} x_i\right]^{-},$$

which implies the first identity of Lemma 3.31. Applying Corollary 3.29 to $f(x)$ shows (after some calculation) that

$$f(x) = -4 \sum_{\substack{i=2:\\ i \text{ even}}}^{n} \left[i - \frac{1}{2} - l\right]^{-},$$

giving the second identity (that E' is identically-0). Applying Corollary 3.29 to the function

$$g(x) = \sum_{r=1}^{n} x_r \left(\sum_{r=1}^{n} x_r + 1\right)$$

yields the third identity.

3.5.3 Upper Bounds on Number of Auxiliary Variables

We now use the results of the two previous subsections and show an explicit construction assuring that any symmetric pseudo-Boolean function admits a quadratization using no more than $n-2$ auxiliary variables. We then show that some popular symmetric pseudo-Boolean functions admit even more economical quadratizations.

Arbitrary Symmetric Pseudo-Boolean Functions

Theorem 3.32. *Every symmetric function of n variables can be quadratized using $n-2$ auxiliary variables.*

Proof. Using Corollary 3.29, we can write any symmetric function f as

$$f(x) = -\alpha_0 \left(\frac{1}{2} + \sum_{j=1}^{n} x_j \right) + \sum_{i=1}^{n} \alpha_i \left[i - \frac{1}{2} - \sum_{j=1}^{n} x_j \right]^{-} .$$

Let $\alpha_r = \min\{\alpha_i : i \text{ even}, i \geq 2\}$ and $\alpha_s = \min\{\alpha_i : i \text{ odd}\}$. Now add to f the expression

$$-\frac{\alpha_r}{2} E' \left(\sum_{j=1}^{n} x_j \right) - \frac{\alpha_s}{2} E'' \left(\sum_{j=1}^{n} x_j \right),$$

which is identically-0. This results in an expression for f of the form

$$f(x) = a_0 + a_1 \sum_{j=1}^{n} x_j + a_2 \sum_{1 \leq i < j \leq n} x_i x_j + \sum_{i=1}^{n} \beta_i \left[i - \frac{1}{2} - \sum_{j=1}^{n} x_j \right]^{-},$$

where, for each i, if i is even, $\beta_i = \alpha_i - \alpha_r \geq 0$, and if i is odd, $\beta_i = \alpha_i - \alpha_s \geq 0$. So all the coefficients β_i are non-negative. Furthermore, $\beta_r = \beta_s = 0$, so we have an expression for f involving no more than $n-2$ positive coefficients β_i. Then,

$$g(x, y) = a_0 + a_1 \sum_{j=1}^{n} x_j + a_2 \sum_{1 \leq i < j \leq n} x_i x_j + \sum_{\substack{i=1: \\ i \neq r,s}}^{n} \beta_i y_i \left(i - \frac{1}{2} - \sum_{j=1}^{n} x_j \right)$$

is a quadratization of f involving at most $n-2$ auxiliary variables. □

A construction in [62] shows an upper bound of $n-1$. This is obtained by adding a multiple of $E(\sum_{r=1}^{n} x_r)$ to each term in the expression from Corollary 3.30, rather than to the expression as a whole, resulting in more complex quadratizations.

Notice that the quadratization in the proof of Theorem 3.32 is y-linear, so we have in fact shown:

Theorem 3.33. *Every symmetric function of n variables has a y-linear quadratization involving at most $n-2$ auxiliary variables.*

Furthermore, these quadratizations are also symmetric in the x-variables. Not every quadratization of a symmetric function must itself be symmetric in the original variables. For example, consider the negative monomial

$$-\prod_{i=1}^{n} x_i = -x_1 x_2 \cdots x_n.$$

As we have seen, this has the quadratization $y\left(n-1-\sum_{j=1}^{n} x_j\right)$, which is symmetric. However, it also has the quadratization

$$(n-2)x_n y - \sum_{i=1}^{n-1} x_i(y-\overline{x}_n),$$

where $\overline{x}_n = 1 - x_n$, which is not symmetric in the x-variables.

Monomials

The quadratization of monomials (positive and negative) has been fairly well-studied (cf. Sections 3.2.2 and 3.2.3). The *standard quadratization* of the negative monomial

$$f(x) = -\prod_{i=1}^{n} x_i = -x_1 x_2 \cdots x_n,$$

is

$$s_n(x_1, x_2, \ldots, x_n, y) = y\left(n-1-\sum_{j=1}^{n} x_j\right).$$

If we apply Corollary 3.29 to the negative monomial, noting that $k_i = 0$ for $i < n$ and $k_n = -1$, we obtain the representation

$$f(x) = 2\left[n - \frac{1}{2} - \sum_{r=1}^{n} x_r\right]^{-},$$

which immediately leads to the quadratization

$$h = 2y\left(n - \frac{1}{2} - \sum_{r=1}^{n} x_r\right),$$

only slightly different from the standard one. We could, instead, apply Corollary 3.30, which would show that $f(x) = \left[n - 1 - \sum_{r=1}^{n} x_r\right]^-$, from which we immediately obtain the standard quadratization.

As we noted earlier, the best known result (smallest number of auxiliary variables) for positive monomials is that they can be quadratized using $\lfloor \frac{n-1}{2} \rfloor$ auxiliary variables. This was shown by Ishikawa [91, 92]. We can see that this many auxiliary variables suffice by using our representation theorem, Corollary 3.29, together with the argument given in the proof of Theorem 3.32.

Theorem 3.34. *The positive monomial $P = \prod_{i=1}^{n} x_i$ can be quadratized using $\lfloor \frac{n-1}{2} \rfloor$ auxiliary variables.*

Proof. Consider first the case where n is even. By Corollary 3.29, noting that $k_i = 0$ for $i < n$ and $k_n = 1$, we have $P = -2\left[n - \frac{1}{2} - l\right]^-$ where $l = \sum_{r=1}^{n} x_r$. By Lemma 3.31,

$$\begin{aligned}
P &= -2\left[n - \frac{1}{2} - l\right]^- + E'(l) \\
&= \frac{l(l-1)}{2} + \sum_{\substack{i=2:\\ i \text{ even}}}^{n-2} 2\left[i - \frac{1}{2} - l\right]^- \\
&= \sum_{1 \le i < j \le n} x_i x_j + \min_y \sum_{\substack{i=2:\\ i \text{ even}}}^{n-2} 2y_i\left(i - \frac{1}{2} - l\right).
\end{aligned}$$

This provides the required quadratization using $\frac{n}{2} - 1 = \lfloor \frac{n-1}{2} \rfloor$ new variables.

When n is odd, one similarly derives the following from Lemma 3.31:

$$\begin{aligned}
P &= -2\left[n - \frac{1}{2} - l\right]^- + E''(l) \\
&= \sum_{i=1}^{n} x_i + \sum_{1 \le i < j \le n} x_i x_j + \min_y \sum_{\substack{i=1:\\ i \text{ odd}}}^{n-2} 2y_i\left(i - \frac{1}{2} - l\right).
\end{aligned}$$

□

This quadratization of P requires the same number of auxiliary variables as the construction of Ishikawa. Both quadratizations are, in fact, identical when n is even, but appear to be different when n is odd.

Note that an alternative approach to the case of odd n would be as follows. Write

$$P = \prod_{i=1}^{n-1} x_i - \prod_{i=1}^{n-1} x_i \overline{x}_n,$$

where $\overline{x}_n = 1 - x_n$. The first term can now be quadratized using $\frac{n-1}{2} - 1$ new variables (since it contains an even number of variables), and the second term, viewed as a negative monomial in $x_1, \dots, x_{n-1}, \overline{x}_n$, has a standard quadratization requiring one further auxiliary variable. Thus, this leads again to a quadratization of P with $\frac{n-1}{2} = \left\lfloor \frac{n-1}{2} \right\rfloor$ new variables. This quadratization is also different from Ishikawa's.

t-out of n and exact-t Functions

Consider now the *t-out-of-n* function defined by:

$$f_{t,n}(x) = 1 \qquad \text{if and only if} \qquad \sum_{i=1}^{n} x_i \geq t.$$

The basic Boolean functions

$$\mathsf{And}_n(x) := \prod_{i=1}^{n} x_i \qquad \text{and} \qquad \mathsf{Or}_n(x) := 1 - \prod_{i=1}^{n} (1 - x_i)$$

are examples of t-out of n functions with $t = n$ and $t = 1$, respectively. Notice that And_n is also a positive monomial in n variables. Another popular example is the *majority* function given by:

$$\mathsf{Maj}_n(x) := \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq \lceil n/2 \rceil, \\ 0 & \text{otherwise,} \end{cases}$$

which breaks ties in favor of ones when n is even. In this case, $t = \lceil n/2 \rceil$.

Corollary 3.35. *The t-out-of-n function $f_{t,n}$ can be quadratized using $\lceil n/2 \rceil$ auxiliary variables.*

Proof. From Corollary 3.29, $f_{t,n}$ can be represented in the form

$$f_{t,n}(x) = \sum_{i=0}^{n} \alpha_i \left[i - \frac{1}{2} - \sum_{j=1}^{n} x_j \right]^{-} \tag{3.35}$$

where $\alpha_i = 0$ when $i < t$, $\alpha_t = -2$, and $\alpha_i = 4(-1)^{i-t-1}$ when $i > t$.

Since the terms of $f_{t,n}$ alternate in sign when $i \geq t$, we can again use Lemma 3.31 to make all coefficients non-negative by adding either $2E'(l)$ or $2E''(l)$ to (3.35), depending on the parity of t. The resulting expression has $\lceil n/2 \rceil$ positive coefficients, and its remaining coefficients are zero. Thus, it can be quadratized with $\lceil n/2 \rceil$ auxiliary variables. □

A related function is the *exact-t* (out of n) function, defined as $f^{=}_{t,n}(x) = 1$ if and only if the Hamming weight of x equals t. Using Corollary 3.29 again, we have that $f^{=}_{t,n}$ can be represented in the form given in (3.35) with $\alpha_i = 0$ when $i < t$, $\alpha_t = -2$, $\alpha_{t+1} = 6$, and $\alpha_i = 0$ when $i > t+1$. Depending on the parity of t, we add $E'(l)$ or $E''(l)$ to (3.35) to obtain an expression with $\lfloor n/2 \rfloor$ positive coefficients, which can then be quadratized with $\lfloor n/2 \rfloor$ auxiliary variables. We have just proved the following:

Corollary 3.36. *The exact-t function $f^{=}_{t,n}$ can be quadratized using $\lfloor n/2 \rfloor$ auxiliary variables.* □

The positive monomial and the And_n Boolean function are also special cases of exact-t functions, both with $t = n$. It is apparent from the argument leading to Corollary 3.36 that the reason the positive monomial (and hence, the And_n function) requires $\lfloor \frac{n-1}{2} \rfloor$ auxiliary variables instead of $\lfloor n/2 \rfloor$ is precisely because $t = n$.

Parity and its Complement

The *parity* function is the (pseudo-)Boolean function $\bigoplus_n(x)$ such that

$$\bigoplus_n(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i, \text{ the Hamming weight of } x\text{, is odd,} \\ 0 & \text{otherwise.} \end{cases}$$

To derive a quadratization of this function, we will use Corollary 3.30 rather than Corollary 3.29, and will make use of a variant of the argument given to establish Theorem 3.32. By Corollary 3.30, we can see that $\bigoplus_n$ has the representation

$$\bigoplus_n(x) = \sum_{j=1}^n x_j + 2\sum_{i=1}^{n-1}(-1)^{i-1}\left[i - \sum_{j=1}^n x_j\right]^- . \qquad (3.36)$$

Let $E(l)$ be as in Lemma 3.31. By adding $E(\sum_{j=1}^n x_j)$ to this representation of $\bigoplus_n$, we obtain a representation with non-negative coefficients, which leads to a quadratization with $m = \lfloor n/2 \rfloor$ auxiliary variables: $\bigoplus_n(x) = \min_{y \in \{0,1\}^m} g(x,y)$ where

$$g(x,y) = 2\sum_{i<j} x_i x_j + \sum_{j=1}^n x_j + 4 \sum_{\substack{i=1:\\ i \text{ odd}}}^{n-1} y_i \left(i - \sum_{j=1}^n x_j \right).$$

Notice that the terms with coefficient -2 in the expansion (3.36) disappear on the addition of E.

The complement of parity, $\overline{\bigoplus}_n$, can be represented as

$$\overline{\bigoplus}_n(x) = 1 - \sum_{j=1}^n x_j + 2\sum_{i=1}^{n-1} (-1)^i \left[i - \sum_{j=1}^n x_j \right]^- ,$$

so, by adding $E(\sum_{j=1}^n x_j)$, to eliminate negative coefficients, we arrive at the following quadratization involving $m = \left\lfloor \frac{n-1}{2} \right\rfloor$ auxiliary variables:

$$g'(x,y) = 1 + 2\sum_{i<j} x_i x_j - \sum_{j=1}^n x_j + 4 \sum_{\substack{i=2:\\ i \text{ even}}}^{n-1} y_i \left(i - \sum_{j=1}^n x_j \right).$$

So we conclude the following:

Theorem 3.37. *The parity function of n variables has a y-linear quadratization involving $\lfloor n/2 \rfloor$ auxiliary variables, and its complement has a y-linear quadratization involving $\left\lfloor \frac{n-1}{2} \right\rfloor$ auxiliary variables.* □

3.5.4 Lower Bounds on Number of Auxiliary Variables

We have two types of lower bounds on the number of auxiliary variables for quadratizations of symmetric pseudo-Boolean functions to present: an existence result, establishing that there are symmetric functions needing a rather large number of auxiliary variables; and a concrete lower bound on the number of auxiliary variables in any y-linear quadratization of the parity function. For improved clarity and also for organizational reasons, we shall defer the presentation of these results to Section 3.7, right after the exposition of our lower bounds for general pseudo-Boolean functions.

3.5.5 d-part Symmetric Pseudo-Boolean Functions

A natural question at this point is whether we can extend (some of) the results to the d-part symmetric setting. While this line of research is still ongoing, we show below how to extend Theorem 3.27 to 2-part symmetric pseudo-Boolean functions.

Theorem 3.38. *Let $f : \{0,1\}^n \to \mathbb{R}$ be a 2-part symmetric pseudo-Boolean function in n variables, with symmetric parts $V_1 \dot\cup V_2 = [n]$ of sizes $n_1+n_2 = n$, and let $0 < \epsilon_{i,j} \le 1$, for $i = 0,1,\dots,n_1$ and $j = 0,1,\dots,n_2$. Then, f can be uniquely represented in the form*

$$f(x) = \sum_{i=0}^{n_1}\sum_{j=0}^{n_2} \alpha_{i,j}\left[\Big((n_2+1)i + j\Big) - \epsilon_{i,j} - \left((n_2+1)\sum_{r_1=1}^{n_1} x_{r_1} + \sum_{r_2=1}^{n_2} x_{r_2}\right)\right]^- .$$

Proof. When $\sum_{r_1 \in V_1} x_{r_1} = p_1$ and $\sum_{r_2 \in V_2} x_{r_2} = p_2$, we should have $f(x) = k_{p_1,p_2}$. So, to find a coefficient vector

$$\alpha = (\alpha_{0,0}, \alpha_{0,1}, \dots, \alpha_{0,n_2}, \alpha_{1,0}, \alpha_{1,1}, \dots, \alpha_{1,n_2}, \dots, \alpha_{n_1,0}, \alpha_{n_1,2}, \dots, \alpha_{n_1,n_2})^T \in \mathbb{R}^\gamma$$

with $\gamma = (n_1+1)(n_2+1)$, which establishes the required representation, we need to find a solution to the following system of γ linear equations:

$$\begin{aligned} k_{p_1,p_2} &= \sum_{i=0}^{n_1}\sum_{j=0}^{n_2} \alpha_{i,j}\left[\big((n_2+1)i+j\big) - \epsilon_{i,j} - \big((n_2+1)p_1 + p_2\big)\right]^- \\ &= \sum_{i=0}^{p_1}\sum_{j=0}^{p_2} \alpha_{i,j}\Big(\big((n_2+1)i+j\big) - \epsilon_{i,j} - \big((n_2+1)p_1 + p_2\big)\Big), \end{aligned} \tag{3.37}$$

for $p_1 = 0,1,\dots,n_1$ and $p_2 = 0,1,\dots,n_2$.

The matrix underlying this system is

$$A = \Big(\left[\big((n_2+1)i+j\big) - \epsilon_{i,j} - \big((n_2+1)r + s\big)\right]^-\Big)_{i,r=0,1,\dots,n_1}^{j,s=0,1,\dots,n_2}.$$

Iterating the pair (r,s) for the rows and the pair (i,j) for the columns, both in lexicographical order, it is possible to see that A is the lower-triangular matrix

$$A = \begin{pmatrix} -\epsilon_{0,0} & 0 & 0 & 0 & \cdots & 0 \\ -1-\epsilon_{0,0} & -\epsilon_{0,1} & 0 & 0 & \cdots & 0 \\ -2-\epsilon_{0,0} & -1-\epsilon_{0,1} & -\epsilon_{0,2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -\beta-\epsilon_{0,0} & -\beta+1-\epsilon_{0,1} & -\beta+2-\epsilon_{0,2} & -\beta+2-\epsilon_{0,3} & \cdots & -\epsilon_{n_1,n_2} \end{pmatrix}$$

where $\beta = n_1 + n_2 + n_1 n_2$.

Because A is lower-triangular with nonzero diagonal entries, A is non-singular and this system does indeed have a unique solution. Therefore the representation exists and is unique. $\square$

The above proof also shows that such cylindrical mapping can also be done for larger values of d and therefore, it is possible to uniquely represent any d-part symmetric pseudo-Boolean function in a different, but similar way to that of Equation (3.38).

In order to quadratize Equation (3.38), we need to repeat the process of shifting the negative coefficients $\alpha_{i,j}$. That is, we need similar relations to those of Lemma 3.31. It is not hard to see that the larger the value of d gets, more involved those relations become. We are currently looking for different ways of accomplishing that goal.

3.6 Upper Bounds on the Number of Auxiliary Variables

Ishikawa [91, 92] observed that, by relying on his procedure for positive monomials (cf. Theorem 3.21) and on Freedman and Drineas' procedure for negative monomials (cf. Theorem 3.17), any pseudo-Boolean function can be quadratized using at most $t\left\lfloor\frac{d-1}{2}\right\rfloor$ auxiliary variables, where d is the degree of the multilinear polynomial (3.3) and t is the number of terms with nonzero coefficients. Since t can be as large as $\binom{n}{d}$, this yields a (tight) upper bound of $O(n^d)$ on the number of auxiliary variables introduced by Ishikawa's procedure for a polynomial of fixed degree d, and $O(n\,2^n)$ variables for an arbitrary function. Note that these bounds depict worst case scenarios, being attained for functions containing all possible positive monomials of degree d or smaller, for each $d \leq n$.

We present some improvements below, by showing that for any pseudo-Boolean function f in n variables, it is possible to find a general quadratization and a y-linear quadratization with at most $O(2^{n/2})$ and $O\big(\frac{2^n}{n}\cdot\log n\big)$ auxiliary variables, respectively. Similarly to what we accomplished to symmetric pseudo-Boolean functions, our methods in this section take a global standpoint in quadratizing f, that is, they do not act over the terms of the multilinear polynomial representing f. This time, however, our methods are based on more complicated extremal combinatorics constructions instead of the linear algebraic argument we used before.

3.6.1 Minterm Quadratizations

We start first with providing a somewhat simple argument that yields a sort of canonical quadratization for every pseudo-Boolean function. We call it *minterm quadratization* since there is an auxiliary variable associated to each and every non-null value in the function's domain.

Theorem 3.39. *Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function in n variables and let M be an upper bound on f, that is, $f(x) \leq M$ for all $x \in \{0,1\}^n$. Then,*

$$g(x,y) = M + \sum_{\alpha \in \{0,1\}^n} \big(M - f(\alpha)\big) \left(\sum_{i:\alpha_i=1} \overline{x}_i + \sum_{j:\alpha_j=0} x_j - 1 \right) y_\alpha$$

is a quadratization of f.

Proof. Let x be some fixed value in $\{0,1\}^n$ and notice that the term $\sum_{i:\alpha_i=1} \overline{x}_i + \sum_{j:\alpha_j=0} x_j - 1$ is negative if and only if $\alpha = x$. Therefore, the minimum of $g(x,y)$ is attained by $y_\alpha = 0$ if $\alpha \neq x$, and by $y_\alpha = 1$ if $\alpha = x$. Hence, $\min_y g(x,y) = f(x)$. □

The above theorem shows in particular that every pseudo-Boolean function $f \in \mathscr{F}_n$ has a quadratization involving at most 2^n auxiliary variables, an improvement when compared to Ishikawa's $O(n\,2^n)$. Moreover, it also shows that minterm quadratizations are not only y-linear, but *y-full*: all terms involve an auxiliary variable.

Notice that $g(x,y)$ is in reality the quadratization of $M+(f-M)$, where $f-M$ only takes negative values. We can use this fact to provide an alternative and also instructive proof of Theorem 3.39. Consider the "minterm normal form" (cf. Section 3.1) of $M+(f-M)$, given by

$$f(x) = M + \sum_{\alpha \in \{0,1\}^n} \big(f(\alpha) - M\big) \left(\prod_{i:\alpha_i=1} x_i \right) \left(\prod_{j:\alpha_j=0} \overline{x}_j \right).$$

It is not hard to see that all the coefficients of $f(\alpha) - M$ in this expression are negative or null. Therefore, all monomials can be quadratized by the extension of Freedman and

Drineas' transformation given in Equation (3.17), and this yields the expression $g(x, y)$ in the statement of the proposition, concluding the proof.

An easy generalization goes as follows.

Theorem 3.40. *For each $k = 1, 2, \ldots, m$, let (P_k, N_k) be a partition of $[n]$ and assume that the subcubes*

$$B_k = \big\{x \in \{0,1\}^n : x_i = 1 \text{ for } i \in P_k,\ x_j = 0 \text{ for } j \in N_k\big\}$$

define a partition of $\{0,1\}^n$ into m subcubes such that f takes constant value f_k on each B_k. Then,

$$g(x, y) = M + \sum_{k=1}^{m} (M - f_k) \left(\sum_{i \in P_k} \overline{x}_i + \sum_{j \in N_k} x_j - 1 \right) y_k$$

is a quadratization of f.

Proof. Immediate by mimicking the minterm normal form based argument provided above. □

3.6.2 y-full Quadratizations

We now present a quadratization procedure that offers a constant improvement in the number of auxiliary variables in the worst case to that of minterm quadratizations, from 2^n to $\frac{3}{8}2^n$. While the improvement is modest, the techniques used may be of (independent) interest in further developments.

Definition 3.41. *Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function and let $g : \{0,1\}^{n+m} \to \mathbb{R}$ be a quadratization of f. We say that g is y-*full *if every term of the multilinear polynomial expression of g contains an auxiliary y-variable.*

As mentioned in the previous subsection, minterm quadratizations are y-full and that implies that every pseudo-Boolean function f admits a y-full quadratization (actually, $f - M$ admits such a quadratization, for M large enough). In the sequel, for convenience, we will refer to a quadratization with m auxiliary variables as an *m-quadratization.*

We start by showing that when a pseudo-Boolean function has a y-full quadratization, then a related quadratization *involving the same y-variables* can be derived for certain extensions of the function to higher-dimensional subcubes.

Lemma 3.42. *Let $f : \{0,1\}^k \to \mathbb{R}$ be a pseudo-Boolean function in k variables, let $g : \{0,1\}^{k+\ell} \to \mathbb{R}$ be a y-full, ℓ-quadratization of f, and let M be any strict majorant of g, that is, $M > \max_{x,y} g(x,y)$. The pseudo-Boolean function*

$$p(x_1, \dots, x_n, y) = g(x_1, \dots, x_k, y) + M\left(\sum_{i=k+1}^{n} x_i\right)\left(\sum_{j=1}^{\ell} y_j\right)$$

in $n+\ell$ variables is a y-full, ℓ-quadratization of the extension of f to n variables:

$$\widetilde{f}(x_1, \dots, x_n) = f(x_1, \dots, x_k) \prod_{i=k+1}^{n} \overline{x}_i.$$

Proof. Let $x \in \{0,1\}^n$. There are two cases to inspect. First, whenever $x_{k+1} = \cdots = x_n = 0$, we immediately have that $\min_y p(x,y) = \min_y g(x,y) = f(x) = \widetilde{f}(x)$. Second, if at least one of the variables $x_{k+1}, \dots, x_n$ is equal to 1, then $\widetilde{f}(x) = 0$; also, since M is an strict majorant of g, we have that $y_1 = \cdots = y_\ell = 0$ in every minimizer p, and hence since g is y-full, $\min_y p(x,y) = p(x,0) = g(x,0) = 0$, concluding the proof. $\square$

Lemma 3.42 allows us to strengthen the construction used for the minterm quadratization by decomposing the pseudo-Boolean function f into subfunctions defined on subcubes of $\{0,1\}^n$. Notice that the difference of it to Theorem 3.40 is that in the latter the pseudo-Boolean function must be constant inside the subcubes.

Proposition 3.43. *Assume that r and ℓ are constants such that for every pseudo-Boolean function $h : \{0,1\}^r \to \mathbb{R}$ in r variables, there exists a constant C_h such that for all $C \geq C_h$, the function $h - C$ has a y-full, ℓ-quadratization. Then, for every $n \geq r$ and for every pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ in n variables, there exists a constant C_f such that $f - C_f$ has a y-full, $(2^{n-r}\ell)$-quadratization.*

Proof. The idea of the proof is to expand f with respect to all 2^{n-r} complete monomials ("minterms") on the last $n-r$ variables, and then to quadratize the coefficient of each monomial using ℓ additional variables per monomial. More precisely, write f as

$$f(x) = \sum_{\alpha \in \{0,1\}^{n-r}} h_\alpha(x_1, \dots, x_r) \left(\prod_{i>r:\, \alpha_i=1} x_i\right)\left(\prod_{j>r:\, \alpha_j=0} \overline{x}_j\right),$$

where $h_\alpha(x_1,\dots,x_r) = f(x_1,\dots,x_r,\alpha)$ for all $x \in \{0,1\}^r$ and $\alpha \in \{0,1\}^{n-r}$. Note that each h_α is a function of r variables, and let $C_f = \max_\alpha C_{h_\alpha}$. Then, by hypothesis, each function $h_\alpha - C_f$ has a y-full, ℓ-quadratization, say $g_\alpha(x, y^\alpha)$.

By Lemma 3.42, there is an M_α large enough such that

$$(h_\alpha(x_1,\dots,x_r) - C_f)\left(\prod_{i>r:\alpha_i=1} x_i\right)\left(\prod_{j>r:\alpha_j=0} \overline{x}_j\right) = \min_{y^\alpha\in\{0,1\}^\ell} p(x,y^\alpha)$$

with

$$p(x,y^\alpha) = g_\alpha(x,y^\alpha) + M_\alpha\left(\sum_{\alpha_i=0} x_i + \sum_{\alpha_i=1} \overline{x}_i\right)\left(\sum_{j=1}^{\ell} y_j^\alpha\right).$$

It then follows that

$$f(x) - C_f = \sum_{\alpha\in\{0,1\}^{n-r}} (h_\alpha(x_1,\dots,x_r) - C_f)\left(\prod_{i>r:\alpha_i=1} x_i\right)\left(\prod_{j>r:\alpha_j=0} \overline{x}_j\right) \tag{3.38}$$

$$= \sum_{\alpha\in\{0,1\}^{n-r}} \min_{y_\alpha\in\{0,1\}^\ell} p(x,y^\alpha) \tag{3.39}$$

$$= \min_y \sum_{\alpha\in\{0,1\}^{n-r}} p(x,y^\alpha) \tag{3.40}$$

where the last equality results from the fact that all vectors y_α of additional variables are distinct, for all $\alpha \in \{0,1\}^{n-r}$. Thus, the function

$$\sum_{\alpha\in\{0,1\}^{n-r}} p(x,y^\alpha)$$

is a y-full quadratization of $f - C_f$ using $(2^{n-r}\ell)$ additional variables y^α, $\alpha \in \{0,1\}^{n-r}$.

□

It is worth noticing that Theorem 3.39 implies that the main assumption of Proposition 3.43 is satisfied for all r by choosing $\ell = 2^r$. This, however, does not lead to an improved bound. Therefore, we will now show that in some sense, y-full quadratizations do not require many more additional variables than arbitrary quadratizations.

Lemma 3.44. *If a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ in n variables has a k-quadratization, then there is a constant C_f such that for all $C \geq C_f$, the pseudo-Boolean function $f - C$ has a y-full, $(n+k)$-quadratization.*

Proof. Let $g(x,y)$ be a k-quadratization of f. If g is not y-full, then consider all its terms involving the variable x_1 and no y-variable, and write the sum of these terms as

$$p(x) = x_1\left(\sum_{j=1}^{n} a_j x_j + b\right) = x_1\left(\sum_{j=1}^{n} a_j x_j + b + K\right) - Kx_1,$$

where K is such that $\left|\sum_{j=1}^{n} a_j x_j + b\right| \leq K$ for all $x \in \{0,1\}^n$.

The function $p(x)$ can be quadratized as

$$p(x) = \min_{v_1}\left\{2Kx_1v_1 + \overline{v}_1\left(\sum_{j=1}^{n} a_j x_j + b + K\right) - Kx_1\right\},$$

where v_1 is a new, auxiliary binary variable. Replacing $p(x)$ by this quadratization in g yields a quadratization $g_1(x,y,v_1)$ of f without quadratic terms of the form x_1x_j, for $j \in [n]$.

Repeating the same procedure for $x_2, x_3, \ldots, x_{n-1}$, introduces in total $n-1$ new additional variables $v_1, v_2, \ldots, v_{n-1}$ and results in a quadratization $h(x,y,v)$ of f in which every term involves a y-variable or v-variable, except for linear terms in x. Now, write h as

$$h(x,y,v) = Q(x,y,v) + \sum_{j=1}^{n} a'_j x_j + b',$$

where Q is (y,v)-full, and let $C_f = \max_{x\in\{0,1\}^n} \sum_{j=1}^{n} a'_j x_j + b'$. Then for all $C \geq C_f$,

$$f(x) - C = \min_{y,v}\{h(x,y,v) - C\} = \min_{y,v,u}\left\{Q(x,y,v) + u\left(\sum_{j=1}^{n} a'_j x_j + b' - C\right)\right\}$$

where u is a new binary variable (which can be set to 1 in every minimizer). The last equation defines a (y,v,u)-full quadratization of $f - C$ involving at most $n+k$ additional variables. □

We are now able to establish the main result of this subsection.

Theorem 3.45. *Every pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ in n variables has a quadratization involving at most $\frac{3}{8}2^n$ auxiliary variables.*

Proof. Recall that Proposition 3.16 states that every pseudo-Boolean function in 4 variables can be quadratized using at most two additional variables. Together with

Lemma 3.44, they imply that for every pseudo-Boolean function h in 4 variables, there is a constant C_h such that for all $C \geq C_h$, the pseudo-Boolean function $h - C$ has a y-full, 6-quadratization. Thus, we can set $r = 4$ and $\ell = 6$ in Proposition 3.43 to obtain the required conclusion when $n \geq 4$. For $n = 3$, the bound follows from Proposition 3.15. □

Let us close this subsection by mention that using Proposition 3.15 instead of Proposition 3.16 in the previous proof would lead to $p = 3$ and $\ell = 4$, and hence to the weaker bound $\frac{1}{2}2^n$.

3.6.3 Universal Sets and Pairwise Cover Quadratizations

In order to improve the $\frac{3}{8}2^n$ upper bound established above on the number of auxiliary variables required in a quadratization, we need to make some observations and introduce some definitions.

Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function in n variables and let $g : \{0,1\}^{n+m} \to \mathbb{R}$ be a quadratization of f using m auxiliary variables, that is, $g(x,y)$ is such that

$$f(x) = \min\{g(x,y) : y \in \{0,1\}^m\} \quad \text{for all} \quad x \in \{0,1\}^n.$$

A slight change in focus allow us to notice that equivalently, we can write $f(x) = g(x, y^*(x))$, where

$$y^*(x) = \operatorname{argmin}\{g(x,y) : y \in \{0,1\}^m\}, \tag{3.41}$$

thus enabling us to view each component y^*_i of y^* as a Boolean function of x: $y^*_i(x) = h_i(x)$.

This (apparent simple) change in perspective turns out to significantly open some good venues.

Definition 3.46. *Let $\mathcal{F} \subseteq \mathscr{F}_n$ be a subset of pseudo-Boolean functions and let $\mathcal{U} \subseteq \mathscr{B}_n$ be a subset of Boolean functions, both in n variables. We say that $\mathcal{U}$ is a* universal set *for $\mathcal{F}$ if*

(i) for every function $f \in \mathcal{F}$, there is a quadratization $g(x,y)$ of f requiring $m \leq |\mathcal{U}|$ auxiliary variables $y_1, y_2, \ldots, y_m$, and

(ii) there is a subset $\{y_1^*, y_2^*, \dots, y_m^*\} \subseteq \mathcal{U}$ such that $y^*(x) = (y_1^*(x), y_2^*(x), \dots, y_m^*(x))$ is a minimizer of $g(x, y)$ for all $x \in \{0,1\}^n$, as in Equation (3.41) *above.*

The main clause in Definition 3.46 tells us that when $\mathcal{U}$ is a universal set, then all minimizers $(y_1^*(x), y_2^*(x), \dots, y_m^*(x))$ can be chosen in $\mathcal{U}$, for all functions in $\mathcal{F}$. Clearly, $\mathscr{B}_n$ itself is a universal set for every set of functions $\mathcal{F}$, and it is not entirely obvious that there should be smaller ones for $\mathcal{F} = \mathscr{F}_n$. We are going to show, however, that rather small universal sets for $\mathscr{F}_n$ can be constructed by relying on the concept of *pairwise covers.*

Definition 3.47. *Let $\mathcal{F}, \mathcal{H} \subseteq 2^{[n]}$ be two hypergraphs. We say that $\mathcal{H}$ is a* pairwise cover *of $\mathcal{F}$ if, for every set $S \in \mathcal{F} \cup \mathcal{H}$ with $|S| \geq 3$, there are two sets $A(S), B(S) \in \mathcal{H}$ such that $|A(S)| < |S|$, $|B(S)| < |S|$, and $A(S) \cup B(S) = S$.*

The motivation behind the above definition is that a pairwise cover $\mathcal{H}$ can be used to partition each monomial of the form $\prod_{j\in S} x_j$ into a product of two monomials $\left(\prod_{j\in A(S)} x_j\right)\left(\prod_{j\in B(S)} x_j\right)$, which can subsequently be replaced by a product of two auxiliary variables $y_{A(S)}\, y_{B(S)}$.

Pairwise covers are closely related to hypergraphs called 2-*bases* by Füredi and Katona [71], Frein, Lévêque and Sebö [70], and Ellis and Sudakov [57]. The only difference to between them resides on the fact that the subsets $A(S), B(S)$ are not required to be *strict* subsets of S in a 2-base.

As an example of a pairwise cover, consider the following hypergraph $\mathcal{H}^*$ defined as

$$\mathcal{H}^* := \left\{ S \subseteq [n] : |S| \leq \left\lceil \frac{1}{3} n \right\rceil \right\} \cup \left\{ S \subseteq [n] : |S| = \left\lceil \frac{2}{3} n \right\rceil \right\}. \tag{3.42}$$

It is not hard to see that $\mathcal{H}^*$ is a pairwise cover for $\mathcal{F} = 2^{[n]}$ of size

$$|\mathcal{H}^*| = \sum_{i=0}^{\lfloor \frac{1}{3} n \rfloor} \binom{n}{i} + \binom{n}{\lceil \frac{1}{3} n \rceil} \leq 2^{H(1/3)\cdot n} = 2^{(\log 3 - \frac{2}{3})n} \leq 2^{0.92n},$$

where the function $H(p) := -p \log p - (1-p)\log(1-p)$ is the *binary entropy* of p, for $0 < p \leq 1/2$. (For a proof of the entropy bound, see Lemma 16.19 in Flum and Grohe [65].)

We are now ready to show the existence of universal sets.

Proposition 3.48. *If $\mathcal{F}, \mathcal{H} \subseteq 2^{[n]}$ are two hypergraphs such that $\mathcal{H} \subseteq \mathcal{F}$ and $\mathcal{H}$ is a pairwise cover of $\mathcal{F}$, then*

$$\mathcal{U}(\mathcal{H}) = \left\{ \prod_{j \in H} x_j : H \in \mathcal{H} \right\}$$

is a universal set for the set of functions of the form $f(x) = \sum_{S \in \mathcal{F}} a_S \prod_{j \in S} x_j$.

Proof. Let $|\mathcal{H}| = m$ and consider a function $f(x) = \sum_{S \in \mathcal{F}} a_S \prod_{j \in S} x_j$. Note that for every choice of nonnegative coefficients b_S, $S \in \mathcal{F}$, we have

$$f(x) = \min_{y \in \{0,1\}^m} \sum_{S \in \mathcal{F}} a_S \prod_{j \in S} x_j + \sum_{H \in \mathcal{H}} b_H \left(y_H \left(|H| - \frac{1}{2} - \sum_{j \in H} x_j \right) + \frac{1}{2} \prod_{j \in H} x_j \right) \quad (3.43)$$

for all $x \in \{0,1\}^n$. This is because $y^*_H = \prod_{j \in H} x_j$ minimizes the right-hand side of (3.43) for all x, and for this value the second summation in the right-hand side is identically zero. This reflects the fact that $y_H \left(|H| - \frac{1}{2} - \sum_{j \in H} x_j \right)$ is nothing but a variant of Freedman and Drineas' quadratization in Equation (3.17) for the negative monomial $-\frac{1}{2} \prod_{j \in H} x_j$.

We now specify the coefficients b_S as follows: for $S \in \mathcal{F}$, $S \notin \mathcal{H}$, we let $b_S = 0$; for $H \in \mathcal{H}$, we let

$$\frac{1}{2} b_H = \sum_{\substack{S \in \mathcal{F}: \\ H \in \{A(S), B(S)\}}} \left(|a_S| + \frac{1}{2} b_S \right). \quad (3.44)$$

Note that for each $H \in \mathcal{H}$, the right-hand side of Equation (3.44) only involves subsets S with $|S| > |H|$. Thus the system of equations (3.44), for $H \in \mathcal{H}$, is triangular and has a nonnegative feasible solution $b_S \geq 0$ for all $S \in \mathcal{F}$.

Let us substitute this solution in Equation (3.43), and let us finally replace every occurence of a term $\prod_{j \in T} x_j$ in Equation (3.43) by $y_{A(T)} y_{B(T)}$. Note that this construction is well defined since $\mathcal{H} \subseteq \mathcal{F}$. It yields a quadratic function $g(x, y)$. We claim that $g(x, y)$ is a quadratization of $f(x)$.

More precisely, consider a point $x \in \{0,1\}^n$. We are going to show that, here again, $y^*_H = \prod_{j \in H} x_j$, for all $H \in \mathcal{H}$, minimizes $g(x, y)$, which entails that $f(x) = \min_{y \in \{0,1\}^m} g(x, y)$ and that $\mathcal{U}(\mathcal{H})$ is a universal set.

To see this, consider an arbitrary set $H \in \mathcal{H}$ and write $g(x,y) = c(x,y)y_H + d(x,y)$, where $c(x,y)$ and $d(x,y)$ do not depend on y_H. More precisely, when $H \in \{A(S), B(S)\}$, define $R(S)$ to be such that $\{H, R(S)\} = \{A(S), B(S)\}$. Then,

$$\begin{aligned} c(x,y) &= \sum_{\substack{S\in\mathcal{F}:\\ H\in\{A(S),B(S)\}}} a_S\, y_{R(S)} + b_H \left(|H| - \frac{1}{2} - \sum_{j\in H} x_j\right) + \frac{1}{2} \sum_{\substack{S\in\mathcal{H}:\\ H\in\{A(S),B(S)\}}} b_S\, y_{R(S)} \\ &= \sum_{\substack{S\in\mathcal{F}:\\ H\in\{A(S),B(S)\}}} \left(a_S + \frac{1}{2}b_S\right) y_{R(S)} + b_H \left(|H| - \frac{1}{2} - \sum_{j\in H} x_j\right). \qquad (3.45) \end{aligned}$$

If $\prod_{j\in H} x_j = 1$, then we get

$$\begin{aligned} c(x,y) &= \sum_{\substack{S\in\mathcal{F}:\\ H\in\{A(S),B(S)\}}} \left(a_S + \frac{1}{2}b_S\right) y_{R(S)} - \frac{1}{2}b_H \qquad (3.46) \\ &\le \sum_{\substack{S\in\mathcal{F}:\\ H\in\{A(S),B(S)\}}} \left(|a_S| + \frac{1}{2}b_S\right) - \frac{1}{2}b_H \qquad (3.47) \\ &= 0, \end{aligned}$$

where the last equality is implied by Equation (3.44). Thus, $c(x,y) \le 0$ and hence $y_H^* = 1$ minimizes $g(x,y)$.

If $\prod_{j\in H} x_j = 0$, then $\sum_{j\in H} x_j \le |H| - 1$, and thus we get by (3.45)

$$\begin{aligned} c(x,y) &\ge \sum_{\substack{S\in\mathcal{F}:\\ H\in\{A(S),B(S)\}}} \left(a_S + \frac{1}{2}b_S\right) y_{R(S)} + \frac{1}{2}b_H \qquad (3.48) \\ &\ge \frac{1}{2}b_H - \sum_{\substack{S\in\mathcal{F}:\\ H\in\{A(S),B(S)\}}} \left(|a_S| + \frac{1}{2}b_S\right) \qquad (3.49) \\ &= 0. \end{aligned}$$

Here the first inequality is implied by $b_H \ge 0$, the second follows from the inequalities

$$\left(a_S + \frac{1}{2}b_S\right) y_{R(S)} \ge \left(-|a_S| - \frac{1}{2}b_S\right),$$

while the last equality follows by (3.44). Thus, $c(x,y) \ge 0$ implies that $y_H^* = 0$ minimizes $g(x,y)$. □

Using Proposition 3.48 with the pairwise cover given in Equation (3.42) immediately establishes that every pseudo-Boolean function in n variables can be quadratized with at most $2^{0.92n}$ auxiliary variables. It turns out that we can improve on that by choosing a different pairwise cover for $\mathcal{F} = 2^{[n]}$.

Theorem 3.49. *Every pseudo-Boolean function in n variables has a quadratization involving at most*

$$2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor} - 2 = O(2^{n/2})$$

auxiliary variables.

Proof. Let $\mathcal{H}^{oe}$ contain all nonempty subsets of $[n]$ consisting either only of odd integers, or only of even integers. Then, $\mathcal{H}^{oe}$ is a pairwise cover of $\mathcal{F} = 2^{[n]}$ with size $2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor} - 2$. Hence, Proposition 3.48 implies that every pseudo-Boolean function on $\{0,1\}^n$ has a quadratization using at most $|\mathcal{H}^{oe}|$ auxiliary variables. $\square$

In particular, it is also not hard to see that $\mathcal{H}^{oe}$ is a 2-base of $2^{[n]}$. The role of odd and even integers in its construction could be replaced by any partition of $[n]$ into two sets V_1, V_2 of nearly-equal sizes $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$. According to Füredi and Katona [71], Erdös has conjectured that this generic construction yields the smallest possible 2-bases of $2^{[n]}$. This conjecture hints that it may be hard to further improve the upper bound given in Theorem 3.49. In fact, we shall revisit this question in Section 3.7 and show that such upper bound is essentially tight — i.e., tight up to constants.

We now establish an upper bound for the case where $\mathcal{F}$ contains all subsets of size at most d and f is a degree-d pseudo-Boolean function, that is, f can be expressed by a degree-d multilinear polynomial in n variables.

Theorem 3.50. *For every fixed $d \leq n$, every degree-d pseudo-Boolean function in n variables has a quadratization involving at most $O(n^{d/2})$ auxiliary variables.*

Proof. Fix d and let $\mathcal{F} = [n]^d := \{S \subseteq [n] : |S| \leq d\}$. In order to establish the theorem, we just need to produce a small pairwise cover of $[n]^d$. For simplicity of the presentation, assume that d is a power of 2, and let $\mathcal{H}^d$ contain all subsets of $[n]$ of sizes $d/2, d/4, d/8, \ldots, 1$. Then, it is easy to see that $\mathcal{H}^d$ is a pairwise cover of $[n]^d$ with size $O(n^{d/2})$. $\square$

3.6.4 Attractive Partitions and y-linear Quadratizations

In this section, we show that every pseudo-Boolean function in n variables admits a y-linear quadratization involving at most $O\big(\frac{2^n}{n} \cdot \log n\big)$ auxiliary variables. Similarly to what happened in the previous section, we start by introducing some intuition and needed definitions.

Let us first recall that the auxiliary variables in a y-linear quadratization appear in terms like $\ell_i(x)y_i$, where $\ell_i(x)$ is a linear function of the original variables $x = \{x_1, x_2, \dots, x_n\}$. When minimizing over y_i, this term contributes the nonpositive quantity $\min\{0, \ell_i(x)\}$.

Given a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, our plan is to start with a symmetric majorant of f and to use a series of y-linear adjustments to push its values down to those of f, layer by layer (where a layer is a subset of vectors of $\{0,1\}^n$ that share the same Hamming weight). More specifically, we will construct a sequence of y-linear quadratic pseudo-Boolean functions $g_0, g_1, \dots, g_n : \{0,1\}^{n+m} \to \mathbb{R}$ (m to be estimated) such that g_{k+1} results from an "adjustment" of g_k. Each function g_k, when minimized on the auxiliary variables $y \in \{0,1\}^m$, produces a function $\sigma_k(x) = \min_{y\in\{0,1\}^m} g_k(x,y)$ that bounds f in the following way:

$$\begin{aligned} \sigma_k(x) &= f(x) \quad \text{for all } x \in \{0,1\}^n \text{ such that } |x| \le k, \text{ and} \\ \sigma_k(x) &\ge f(x) \quad \text{whenever } |x| > k, \end{aligned}$$

where $|x| := \sum_{i=1}^n x_i$ is the *Hamming weight* of x. In particular, g_n is a y-linear quadratization of f. In order to understand our construction, it helps noticing that when we introduce the new variables to adjust to the values of f in layer k, we may decrease the values of our new approximation for vectors with $|x| > k$ by much more than intended. To make sure that the sequence remains above f, we will start with a symmetric majorant of f, which is increasingly larger than f on higher layers, to preventively compensate for later "accidental" decreases.

Let us denote by $Q_k := \{a \in \{0,1\}^n : |a| = k\}$, for $k = 0, 1, \dots, n$, the set of $\binom{n}{k}$ elements that constitutes the k-th *layer* of $\{0,1\}^n$, that is, those elements having Hamming weight k. For $a \in Q_k$ and $k = 0, 1, \dots, n-1$, let us call the set $N(a) := \{b \in$

$Q_{k+1} : |b-a| = 1\}$ as the *upper neighborhood of* a, with each element in $N(a)$ being an *upper neighbor* of a.

Definition 3.51. *Let* $\mathcal{A}^n := \{A_0, A_1, \dots, A_{n-1}\}$ *be a family of sets such that* $\emptyset \neq A_k \subseteq Q_k$ *for all* k*, and let* $\mathscr{A} = \{\mathbf{0}\} \cup \{\Delta(a) : a \in \bigcup_{k=0}^{n-1} A_k\}$ *be a family of sets such that* $\emptyset \neq \Delta(a) \subseteq N(a)$ *for all* a*. We say that* $\mathscr{A}$ *is an* attractive partition *of* $\{0,1\}^n$ induced by $\mathcal{A}^n$ *if*

$$\bigcup_{a \in A_k} \Delta(a) = Q_{k+1} \quad \textit{and} \quad \Delta(a) \cap \Delta(a') = \emptyset,$$

for all $k = 0, 1, \dots, n-1$ *and for all* $a, a' \in A_k$ *with* $a \neq a'$*. We say that* $\mathcal{A}^n$ *is an* inductor *of the partition. Its size is defined as* $|\mathcal{A}^n| := \sum_{k=0}^{n-1} |A_k|$*.*

Even though $\mathcal{A}^n$ does not uniquely defines $\mathscr{A}$, we frequently find it convenient to identify the partition with its inductor when this does not create confusion. Also, notice that $A_0 = \{\mathbf{0}\}$ by definition, i.e., A_0 is the set whose only element is the all-zero vector.

For each element $a \in \bigcup_{k=0}^{n-1} A_k$ of an inductor, let us define the set

$$\delta(a) := \{i \in [n] : a_i = 0 \text{ and } a_i + \mathbf{e}_i \in \Delta(a)\}.$$

Let also $\tilde{a}$ denote the binary vector

$$\tilde{a}_i := \begin{cases} a_i & \text{if } i \notin \delta(a), \\ 1 & \text{otherwise,} \end{cases}$$

and define the *subcube of* $\{0,1\}^n$ *induced by* a and $\Delta(a)$ as

$$[a, \tilde{a}] := \{b \in \{0,1\}^n : a \leq b \leq \tilde{a}\}.$$

We will now show how to use an attractive partition $\mathcal{A}^n$ to construct a y-linear quadratization of a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$. Our first ingredients in the construction of a y-linear quadratization for f are symmetric pseudo-Boolean functions $s_k : \{0,1\}^n \to \mathbb{R}$, for $k = 0, 1, \dots, n+1$, in the same variables as f:

$$s_k(x) := \begin{cases} 0 & \text{if } |x| < k \\ D_k & \text{if } |x| \geq k, \end{cases} \tag{3.50}$$

where $D_k \geq 0$ are constants to be specified later.

As shown in Section 3.5.3, the k-out-of-n function s_k has a y-linear quadratization, say $\hat{s}_k(x,y)$, requiring only $\lceil \frac{n}{2} \rceil$ auxiliary variables. Let us denote by y_j^k, $j = 1, 2, \ldots, \frac{n}{2}$ the auxiliary variables appearing in $\hat{s}_k$, $k = 0, 1, \ldots, n$. We emphasize that for $k \neq \ell$ the functions $\hat{s}_k$ and $\hat{s}_\ell$ depend on disjoint sets of auxiliary variables, and hence

$$\min_y \left(\hat{s}_k(x,y) + \hat{s}_\ell(x,y)\right) = s_k(x) + s_\ell(x) \tag{3.51}$$

for all $x \in \{0,1\}^n$.

We next define the following sequence of quadratic pseudo-Boolean functions: for $k = 0, 1, \ldots, n-1$,

$$g_0(x,y) := \hat{s}_0(x,y) + \hat{s}_1(x,y), \tag{3.52}$$

$$g_{k+1}(x,y) := \hat{s}_{k+2}(x,y) + g_k(x,y) + \sum_{a \in A_k} y_a h_a(x), \tag{3.53}$$

where $y_a \in \{0,1\}$ is an auxiliary variable for each $a \in A_k$, and

$$h_a(x) := \alpha_a \left(\sum_{i: a_i = 1} \overline{x}_i + \sum_{\substack{i: a_i = 0, \\ i \notin \delta(a)}} x_i \right) - \sum_{i \in \delta(a)} \Big(\sigma_k(a + \mathbf{e}_i) - f(a + \mathbf{e}_i) \Big) x_i, \tag{3.54}$$

with $\alpha_a = 1 + \sum_{i \in \delta(a)} |\sigma_k(a + \mathbf{e}_i) - f(a + \mathbf{e}_i)|$,

$$\sigma_k(x) := \min_{y \in \{0,1\}^m} g_k(x,y) \quad \text{for all} \quad k = 0, 1, \ldots, n, \tag{3.55}$$

and where $\mathbf{e}_i$ denotes the unit vector with i-th entry equal to 1 and all other entries equal to 0.

At this point, we are ready to specify the constants D_k involved in definition (3.50) of the functions $s_k(x)$. We set

$$D_0 := f(\mathbf{0}), \; D_1 := \max_{x \in \{0,1\}^n} f(x) - f(\mathbf{0}), \tag{3.56}$$

and recursively, for $k = 0, 1, \ldots, n-1$,

$$D_{k+2} := (n-k)\, |A_k| \max_{x \in Q_{k+1}} (\sigma_k(x) - f(x)). \tag{3.57}$$

Note that σ_k only depends on $D_0, D_1, \ldots, D_{k+1}$ (through $g_0, g_1, \ldots, g_k$), and hence D_{k+2} is well-defined.

We note some simple consequences of the previous definitions.

Fact 3.52. *For each $a \in \bigcup_{k=0}^{n-1} A_k$,*

(i) $h_a(x) = -\sum_{i\in\delta(a)} \Big(\sigma_k(a+\mathbf{e}_i) - f(a+\mathbf{e}_i)\Big)x_i$ *when* $x \in [a, \tilde{a}]$*;*

(ii) $h_a(a) = 0$*;*

(iii) $h_a(x) > 0$ *for* $x \notin [a, \tilde{a}]$*;*

(iv) $\min_{y_a} y_a h_a(x) = 0$ *for all* $x \notin [a, \tilde{a}]$ *and for* $x = a$*.*

Proof. If $x \in [a, \tilde{a}]$, then by definition of δ, we have that $x_i = a_i$ for all $i \notin \delta(a)$. Hence the summation

$$\sum_{i:a_i=1} \overline{x}_i + \sum_{\substack{i:a_i=0,\\ i\notin\delta(a)}} x_i \tag{3.58}$$

vanishes in the definition of $h_a(x)$, and (i) follows. If $x = a$, then all terms of of $h_a(x)$ vanish since $a_i = 0$ when $i \in \delta(a)$, thus implying (ii). If $x \notin [a, \tilde{a}]$, then Equation (3.58) is positive, and thus (iii) follows by definition of α_a. Finally, (iv) is a direct consequence of (ii) and (iii). □

Fact 3.53. *For each $k = 0, 1, \ldots, n$, the function g_k only depends on the original variables $x_1, \ldots, x_n$, on the $(k+2)\frac{n}{2}$ auxiliary variables y_j^ℓ, $\ell = 0, 1, \ldots, k+1$, $j = 1, 2, \ldots, \frac{n}{2}$ occurring in $\hat{s}_0$,$\hat{s}_1$,…, $\hat{s}_{k+1}$, and on the auxiliary variables y_a for $a \in \bigcup_{j=0}^{k-1} A_j$.*

Proof. Immediately follows from Equations (3.52)–(3.55), by induction. □

In view of Fact 3.53, the three main terms in (3.53) depend on disjoint sets of auxiliary variables. Thus,

$$\begin{aligned}
\sigma_{k+1}(x) &= \min_y g_{k+1}(x, y) \\
&= \left(\min_y \hat{s}_{k+2}(x, y)\right) + \left(\min_y g_k(x, y)\right) + \left(\min_y \sum_{a\in A_k} y_a h_a(x)\right) \\
&= s_{k+2}(x) + \sigma_k(x) + \left(\min_y \sum_{a\in A_k} y_a h_a(x)\right).
\end{aligned}\tag{3.59}$$

We will repeatedly rely on equality (3.59) in the sequel.

We are now ready to establish the main properties of the above construction.

Proposition 3.54. *Let $\mathscr{A}$ be an attractive partition induced by $\mathcal{A}^n$. Then, for all $x \in \{0,1\}^n$ and all $k = 0, 1, \ldots, n$, we have that*

$$f(x) = \sigma_k(x) \quad \text{if } |x| \leq k, \tag{3.60}$$

$$f(x) \leq \sigma_k(x) \quad \text{if } |x| > k. \tag{3.61}$$

In particular, $f(x) = \sigma_n(x)$, and thus $g_n(x, y)$ is a y-linear quadratization of $f(x)$ involving $m = O(n^2) + |\mathcal{A}^n|$ auxiliary variables.

Proof. Let $x \in \{0,1\}^n$ be arbitrary. The proof is by induction on k. In case $k = 0$, (3.50) and (3.56) easily imply that, for all $x \in \{0,1\}^n$,

$$\begin{cases} s_0(x) & = f(\mathbf{0}), \text{ and} \\ s_0(x) + s_1(x) & \geq f(x). \end{cases} \tag{3.62}$$

In view of (3.51), (3.52) and (3.55), it follows that

$$\sigma_0(x) = f(\mathbf{0}) \quad \text{if } |x| = 0,$$

$$\sigma_0(x) \geq f(x) \quad \text{if } |x| > 0.$$

Now suppose the statement is valid for $k < n$ and let us show that it is also valid for $k + 1$. We divide the analysis into three cases:

Case 1: $|x| \leq k$.

Either $x \notin [a, \tilde{a}]$ for all $a \in A_k$, or $x = a' \in A_k$. In both cases we have $\min_y \sum_{a \in A_k} y_a h_a(x) = 0$ by Fact 3.52. Furthermore, $s_{k+2}(x) = 0$ by definition, since $|x| < k + 2$. Thus, by (3.59) we get

$$\sigma_{k+1}(x) = \sigma_k(x) = f(x),$$

where the last equality follows by the induction hypothesis.

Case 2: $|x| = k + 1$.

Since $\mathcal{A}^n$ is an attractive partition, there are unique $a' \in A_k$ and $i \in \delta(a')$ such that $x = a' + \mathbf{e}_i$. Note that for all $a \neq a'$, $a \in A_k$ we have $x \notin [a, \tilde{a}]$, and hence $\min_{y_a} y_a h_a(x) = 0$ by Fact 3.52.

Moreover, $y_{a'}h_{a'}(x) = y_{a'}(-\sigma_k(x) + f(x))$ as $x_{i'} = 0$ for all $i' \in \delta(a')$ with $i' \neq i$. Therefore, since $s_{k+2}(x) = 0$ by definition for vectors with $|x| < k+2$, and since $\sigma_k(x) \geq f(x)$ by our inductive hypothesis, we get

$$\sigma_{k+1}(x) = \sigma_k(x) + \min_{y_{a'} \in \{0,1\}} y_{a'}(-\sigma_k(x) + f(x)) = \sigma_k(x) - \sigma_k(x) + f(x) = f(x).$$

Case 3: $|x| > k+1$.

If, for some $a \in A_k$, $x \notin [a, \tilde{a}]$, then by Fact 3.52 again

$$\min_y y_a h_a(x) = 0.$$

Thus, we get

$$\min_y \sum_{a \in A_k} y_a h_a(x) = \min_y \sum_{\substack{a \in A_k \\ x \in [a,\tilde{a}]}} y_a \left(- \sum_{i \in \delta(a)} \big(\sigma_k(a + \mathbf{e}_i) - f(a + \mathbf{e}_i)\big) x_i \right).$$

Note that the coefficient of each variable x_i in the parenthesis is nonpositive by our induction hypothesis, and hence

$$\min_y \sum_{a \in A_k} y_a h_a(x) = - \sum_{\substack{a \in A_k \\ x \in [a,\tilde{a}]}} \sum_{i \in \delta(a)} \big(\sigma_k(a + \mathbf{e}_i) - f(a + \mathbf{e}_i)\big) x_i.$$

Furthermore $s_{k+2}(x) = D_{k+2}$, since $|x| \geq k+2$. Consequently, (3.59) and the induction hypothesis imply that

$$\begin{aligned} \sigma_{k+1}(x) = {} & D_{k+2} + \sigma_k(x) - \sum_{\substack{a \in A_k \\ x \in [a,\tilde{a}]}} \sum_{i \in \delta(a)} \left(\sigma_k(a + \mathbf{e}_i) - f(a + \mathbf{e}_i)\right) x_i \\ \geq {} & D_{k+2} + f(x) - \sum_{\substack{a \in A_k \\ x \in [a,\tilde{a}]}} \sum_{i \in \delta(a)} \big(\sigma_k(a + \mathbf{e}_i) - f(a + \mathbf{e}_i)\big) x_i. \end{aligned}$$

Note that the double summation in this last expression contains at most $(n - k)|A_k|$ terms, each not larger than $\max_{x \in Q_{k+1}}(\sigma_k(x) - f(x))$. Hence

$$\sigma_{k+1}(x) \geq D_{k+2} + f(x) - (n-k)\,|A_k| \max_{x \in Q_{k+1}} \left(\sigma_k(x) - f(x)\right),$$

and by (3.57), $\sigma_{k+1}(x) \geq f(x)$ holds for all $x \in \{0,1\}^n$, concluding the induction.

The last assertion of the proposition follows now immediately from Fact 3.53. □

We now need to show that there exists a small enough inductor. We shall derive this from classical extremal combinatorial results related to Turán's problem.

Definition 3.55. *A family $\mathcal{T} = \mathcal{T}(n, r, k)$ of k-element subsets of $[n]$ is a* Turán (n, r, k)-system *if every r-element subset of $[n]$ contains at least one element of $\mathcal{T}$. The minimum size of such a family is the* Turán number $T(n, r, k)$*.*

Turán systems are interesting in our context because they can be used to obtain attractive partitions of $\{0,1\}^n$ by identifying each subset of $[n]$ with its characteristic vector. More specifically, for each k, consider a Turán $(n, k+1, k)$-system $\mathcal{T}_k$ and let A_k be the corresponding subset of Q_k. By Definition 3.55, for each vector $x \in Q_{k+1}$ (i.e., subset of $[n]$ of size $k+1$), there is a vector $a(x) \in A_k$ such that $a(x) \leq x$ (if there are several possible choices for $a(x)$, just pick one arbitrarily). Then, for each $a \in A_k$, we can define

$$\Delta(a) = \big\{x \in Q_{k+1} : a = a(x)\big\}$$

and this yields an attractive partition induced by $\mathcal{A}^n = \{A_0, A_1, \ldots, A_{n-1}\}$.

Kim and Roush [100], Frankl and Rödl [68], and Sidorenko [139] (see also the survey by Sidorenko [138]), presented a chain of improved bounds on Turán $\mathcal{T}(n, k+1, k)$-numbers alongside with procedures to construct Turán $\mathcal{T}(n, k+1, k)$-systems satisfying those bounds. For our purposes, either one suffices.

Theorem 3.56 (Frankl and Rödl [68])**.** *It holds that*

$$T(n, k+1, k) \leq \frac{\ln k + O(1)}{k}\binom{n}{k} \qquad \textit{for all} \qquad k = 0, 1, \ldots, n-1.$$

□

We can then establish the main result of this subsection.

Theorem 3.57. *Every pseudo-Boolean function in n variables has a y-linear quadratization involving at most $O\big(\frac{2^n}{n} \cdot \log n\big)$ auxiliary variables.*

Proof. Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function in n variables, and let $\mathcal{A}^n$ be an inductor of an attractive partition for f. Since $|A_0| = 1$, we can estimate $|\mathcal{A}^n|$ as

$$\begin{aligned}|\mathcal{A}^n| = \sum_{k=0}^{n-1} |A_k| &\leq 1 + \sum_{k=1}^{n-1} \frac{\ln k + O(1)}{k} \binom{n}{k} \\ &\leq 1 + 2\frac{\ln n + O(1)}{n+1} \sum_{k=1}^{n-1} \frac{n+1}{k+1} \binom{n}{k} \\ &\leq 1 + \frac{\ln n + O(1)}{n+1} 2^{n+2}.\end{aligned}$$

Now combining the above with Proposition 3.54, we have that the number of auxiliary variables of g_n can be upper bounded as

$$m = O\left(\frac{2^n}{n} \log n\right).$$

□

Before closing this section, let us remark that the attractive partition constructed above depends only on the dimension n, and not on the actual pseudo-Boolean function f. Hence, a deeper analysis of the proof of Theorem 3.57 actually reveals the existence of a universal set of Boolean functions of cardinality $O\big(\frac{2^n}{n} \cdot \log n\big)$ such that any pseudo-Boolean function in n variables has a y-linear quadratization using a subset of this universal set as new variables, in the sense of Definition 3.46. This claim holds notwithstanding the fact that all functions s_k, h_a, g_k, depend to some extent on f: the claim is only that the optimal value assumed by the y-variables is independent of f.

3.7 Lower Bounds on the Number of Auxiliary Variables

In the previous section, we showed some global, semi-oblivious procedures that can obtain general quadratizations and y-linear quadratizations with at most $O(2^{n/2})$ and $O\big(\frac{2^n}{n} \cdot \log n\big)$ auxiliary variables, respectively, for a pseudo-Boolean function in n variables. The global qualification being due to non termwise character of the quadratizations, and the semi-oblivious to the fact that any specific structure the functions might present is ignored; only their values are taken into account to compute the right coefficients in the quadratized forms. Comparing those bounds with the previous one available of $O(n\,2^n)$ (cf. Ishikawa's [91, 92]), they are clearly good improvements.

A natural question at this point then is whether we can further improve those bounds, through the use of different, novel, or more involved combinatorial constructions, or whether we can exhibit a barrier that prevents such advancements. Notice that here we are still interested in global, semi-oblivious procedures: we already showed a specific class of pseudo-Boolean functions, the symmetric ones, where availability of differentiated structural information allowed us to do better.

Below, we provide an answer to that question that favors the second possibility. We show lower bounds (via linear algebraic arguments) matching those upper bounds of Theorems 3.49 and 3.50 up to constant factors, and that of Theorem 3.57 up to a logarithmic factor. Moreover, our proofs reveal that this is the case for almost all pseudo-Boolean functions, thus providing evidence for the strength of the aforementioned theorems and constructions of the previous section.

Afterwards, through a set of different techniques, we also provide some lower bounds for symmetric pseudo-Boolean functions. Contrary to the global, semi-oblivious setting, there are gaps (quadratic and to within a logarithmic factor) between the upper bounds of Section 3.5.3 and the lower bounds presented here.

We start with the general case.

Theorem 3.58. *There are pseudo-Boolean functions in n variables for which every quadratization must involve at least $\Omega(2^{n/2})$ auxiliary variables.*

Proof. Let $f(x) = f(x_1, x_2, \ldots, x_n)$ be a pseudo-Boolean function in n variables and suppose that f can be quadratized using m auxiliary variables: $y_1, y_2, \ldots, y_m$. Hence, there is a quadratic pseudo-Boolean function $g(x, y) : \{0,1\}^{n+m} \to \mathbb{R}$ of the form

$$g(x,y) = a + \sum_{i,j=1}^{n} b_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_i y_j + \sum_{i,j=1}^{m} d_{ij} y_i y_j$$

such that

$$f(x) = \min\bigl\{g(x,y) : y \in \{0,1\}^m\bigr\} \quad \text{for all } x \in \{0,1\}^n.$$

Recall from Equation (3.41) that we can write $f(x) = g(x, y^*(x))$, where

$$y^*(x) = \operatorname{argmin}\bigl\{g(x,y) : y \in \{0,1\}^m\bigr\}, \tag{3.63}$$

thus viewing each y_i^* as a Boolean function of x, say, $y_i^*(x) = h_i(x)$. Then,

$$f(x) = a + \sum_{i,j=1}^{n} b_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_i h_j(x) + \sum_{i,j=1}^{m} c_{ij} h_i(x) h_j(x).$$

This shows that f is a linear combination of the following set of pseudo-Boolean functions defined on $\{0,1\}^n$, where h denotes $(h_1, \ldots, h_m)$:

$$L_h = \{1, x_i x_j, x_i h_r, h_r h_s : 1 \leq i, j \leq n; 1 \leq r, s \leq m\}. \tag{3.64}$$

Note that

$$|L_h| = \ell(n,m) = 1 + n + \binom{n}{2} + mn + m + \binom{m}{2}.$$

The set $\mathscr{F}_n$ of pseudo-Boolean functions in n variables forms a vector space of dimension 2^n and is isomorphic to $\mathbb{R}^{2^n}$. Indeed, any function in $\mathscr{F}_n$ can be regarded as a vector in $\mathbb{R}^{2^n}$, and any such vector corresponds to a pseudo-Boolean function: the components of the vector give the values of the pseudo-Boolean function at each of the 2^n points of the domain, in some fixed order.

Let $\mathscr{V}_m$ be the set of pseudo-Boolean functions which can be quadratized using at most m auxiliary variables, and regard $\mathscr{V}_m$ as a subset of $\mathbb{R}^{2^n}$. The discussion above shows that for any $f \in \mathscr{V}_m$, there exists some choice of Boolean functions $h_1, h_2, \ldots, h_m$, such that f is contained in the subspace $\operatorname{span}(L_h)$ of $\mathbb{R}^{2^n}$ spanned by the functions in L_h. It follows that $\mathscr{V}_m$ is contained in the union $\bigcup_h \operatorname{span}(L_h)$, where the union is over all possible choices of m Boolean functions $(h_1, h_2, \ldots, h_m)$ of n variables. But there is only a finite number, 2^{2^n}, of possibilities for each of the Boolean functions h_i. So $\mathscr{V}_m$ is contained in a finite union of subspaces, each of dimension at most $\ell(n,m)$. If all pseudo-Boolean functions can be quadratized using m auxiliary variables, then $\mathscr{V}_m$ — and hence this union — must be the whole of the space $\mathscr{F}_n \cong \mathbb{R}^{2^n}$. That cannot be the case if $\ell(n,m) < 2^n$. In other words, if m auxiliary variables suffice to quadratize any pseudo-Boolean function, then $\ell(n,m) \geq 2^n$, which implies that

$$2^n - 2 - n \leq m^2 + 2nm + m + n^2 \leq m^2 + 3nm + \frac{9}{4}n^2 = \left(m + \frac{3}{2}n\right)^2,$$

which in turn gives that

$$m \geq 2^{n/2}\sqrt{2 - \frac{n+1}{2^n}} - \frac{3}{2}n.$$

Therefore, $m = \Omega(2^{n/2})$ and this concludes the proof. $\square$

Similarly to what was done in Section 3.6.3, we can specialize the above proof for the class of pseudo-Boolean functions of bounded degree d, i.e., functions expressed by polynomials of degree d.

Theorem 3.59. *For every fixed d, there are pseudo-Boolean functions in n variables and of degree d for which every quadratization must involve at least $\Omega(n^{d/2})$ auxiliary variables.*

Proof. Let $\mathscr{F}_{n,d}$ denotes the set of pseudo-Boolean functions in n variables and of degree d. We have that $\mathscr{F}_{n,d}$ is a linear subspace of the space $\mathscr{F}_{n,n} = \mathscr{F}_n$, and its dimension is $\dim(\mathscr{F}_{n,d}) = \phi(n,d) := \sum_{k=0}^{d} \binom{n}{k}$.

If all functions in $F_{n,d}$ can be quadratized using m auxiliary variables, then each of the subspaces $\text{span}(L_h)$ introduced in the proof of Theorem 3.58 must be of dimension at least $dim(\mathscr{F}_{n,d})$, that is, it must be the case that $\ell(n,m) \geq \phi(n,d)$, which implies that

$$\phi(n,d) - 2 - n \leq m^2 + 2nm + m + n^2 \leq m^2 + 3nm + \frac{9}{4}n^2 = \left(m + \frac{3}{2}n\right)^2,$$

which in turn gives that

$$m \geq \sqrt{2\phi(n,d) - 2 - n} - \frac{3}{2}n.$$

As $\phi(n,d) \geq \binom{n}{d} = \Theta(n^d)$, it follows that $m = \Omega(n^{d/2})$. $\square$

Notice that Theorems 3.58 and 3.59 match the upper bounds given by Theorems 3.49 and 3.50, respectively, up to constant factors.

We now restrict the class of quadratizations that we consider and show that y-linear quadratizations must in a similar way, necessarily contain many auxiliary variables.

Theorem 3.60. *There are pseudo-Boolean functions in n variables for which every y-linear quadratization must involve at least $\Omega(2^n/n)$ auxiliary variables.*

Proof. Let $\mathscr{W}_m$ be the set of pseudo-Boolean functions for which there is a quadratization involving at most m auxiliary variables, and not including any terms of the form $y_i y_j$ with $i \neq j$.

Then we can repeat the argument given in the proof of Theorem 3.58, omitting the products $h_i h_j$ from L_h when $i \neq j$, to obtain a set L'_h, of size

$$|L'_h| = \ell'(n,m) = 1 + n + \binom{n}{2} + mn + m.$$

We conclude as before that for all pseudo-Boolean functions to be quadratizable in this way, we would need $\mathscr{W}_m = \mathbb{R}^{2^n}$ and so $\ell'(n,m) \geq 2^n$, implying that

$$m \geq \frac{2^n - n^2 - n - 1}{n+1},$$

from which it follows that $m = \Omega(2^n/n)$. □

Theorem 3.61. *For every fixed d, there are pseudo-Boolean functions in n variables and of degree d for which every y-linear quadratization must involve at least $\Omega(n^{d-1})$ auxiliary variables.*

Proof. Combining the proofs of Theorems 3.59 and 3.60, it follows that in order for all degree-d pseudo-Boolean functions in n variables to admit a y-linear quadratization involving at most m auxiliary variables, we must have that $\ell'(n,m) \geq \phi(n,d)$, which implies that

$$m \geq \frac{\phi(n,d)}{n+1} - \frac{n^2+n+2}{2n+2}.$$

As $\phi(n,d) \geq \binom{n}{d} = \Theta(n^d)$, it follows that $m = \Omega(n^{d-1})$. □

Notice that Theorem 3.60 matches the upper bound given by Theorem 3.57 up to an $O(\log n)$ factor, and we believe the key to close this gap lies in better constructions for Turán systems. While we do not yet have a counterpart upper bound to Theorem 3.61, we believe it not only to be possible, but to come from a non immediately obvious adaptation of attractive partitions.

Let us formulate some comments about the theorems we presented above. We start by pointing out the obvious fact all the proofs above hinges on the same dimension-based argument. More specifically, they stand upon the fact that for any Euclidean space $\mathbb{R}^k$, the union of any finite collection of subspaces of dimension less than k cannot be $\mathbb{R}^k$; in fact, such union would be a set of Lebesgue measure zero. This in turn implies that the set of pseudo-Boolean functions, regarded as a subset of $\mathbb{R}^{2^n}$, which can be quadratized with fewer variables than the bounds established on the theorem's statements has Lebesgue measure zero. In other words, the proofs reveal that not only there exists certain pseudo-Boolean functions satisfying those claims, but that for *almost all* pseudo-Boolean functions within those hypothesis must use the stated number of auxiliary variables. For instance, for almost all pseudo-Boolean functions, any quadratization must use at least $\Omega(2^{n/2})$ auxiliary variables and any y-linear quadratization must use at least $\Omega(2^n/n)$ auxiliary variables.

It is also worth noticing that another crucial ingredient of these lower bound proofs is the interpretation of auxiliary variables as Boolean functions and that pseudo-Boolean functions can be written in terms of them, without explicitly referring to quadratizations. This fact suggests that universal sets (cf. Definition 3.46) may be indeed ubiquitous in studying and representing pseudo-Boolean functions as low degree multilinear polynomials in higher dimensional spaces.

We now turn out attention to symmetric pseudo-Boolean functions.

3.7.1 Lower Bounds for Symmetric Pseudo-Boolean Functions

As we already mentioned before (and it is also easy to see), there are only 2^{n+1} different symmetric Boolean functions in n variables. This implies that the dimension-based argument used in the proof of Theorem 3.58 does not directly provide any useful information for symmetric pseudo-Boolean functions. In fact, symmetric pseudo-Boolean functions in $\mathbb{R}^{2^n}$ form a subspace of dimension equal to $n+1$, giving that

$$\ell(n,m) = 1 + n + \binom{n}{2} + mn + m + \binom{m}{2} \geq n+1,$$

which readily implies $m \geq 0$. A similar situation happens for y-linear quadratizations. Fortunately, by using a simple, but very clever base transformation argument, we can find a way around and exhibit some nontrivial lower bounds for symmetric pseudo-Boolean functions.

The following result is inspired by (but is different and does not follow from) a transformation given in Siu, Roychowdhury and Kailath [140], in the framework of the representation of Boolean functions by threshold circuits. This result relates quadratizations of arbitrary (possibly non-symmetric) pseudo-Boolean functions to the quadratization of symmetric functions on a larger, related, number of variables.

Lemma 3.62. *Suppose that n, m are positive integers and suppose that every symmetric pseudo-Boolean function $F(z)$ of $N = 2^n - 1$ variables (that is, every symmetric function $F : \{0,1\}^{2^n-1} \to \mathbb{R}$) has an m-quadratization. Then every (arbitrary) pseudo-Boolean function $f(x)$ on $\{0,1\}^n$ also has an m-quadratization.*

Proof. Let $f(x)$ be an arbitrary pseudo-Boolean function of n variables. We are going to construct a sequence of four functions k, F, G, g, such that g is a quadratization of f. For this purpose, let $N = 2^n - 1$.

1. Let $k : \{0, 1, \ldots, N\} \to \mathbb{R}$ be defined as follows: $k(w) := f(x)$ where x is the binary representation of w, that is, $w = \sum_{i=1}^{n} 2^{i-1} x_i$.

2. Let F be the symmetric pseudo-Boolean function of N variables defined by: for all $z \in \{0,1\}^N$, $F(z) := k(|z|)$, where $|z|$ is the Hamming weight of z. (This defines F completely, given that it is symmetric.)

3. Let $G(z, y)$ be an arbitrary quadratization of $F(z)$ using m auxiliary variables. (The hypothesis of the theorem is that such quadratizations exist.)

4. Finally, let $g(x, y)$ be the pseudo-Boolean function on $\{0,1\}^{n+m}$ that is obtained by identifying each of the variables $z_{2^{j-1}}, z_{2^{j-1}+1}, \ldots, z_{2^j-1}$ with x_j in $G(z, y)$, for $j = 1, 2, \ldots, n$; that is,

$$g(x_1, x_2, x_3, \ldots, x_n, y) := G(x_1, x_2, x_2, x_3, x_3, x_3, x_3, \ldots, x_n, \ldots, x_n, y).$$

   (The unification makes sense since $2^{j-1} x_j = z_{2^{j-1}} + z_{2^{j-1}+1} + \cdots + z_{2^j-1}$, for all $j = 1, 2, \ldots, n$.)

We claim that $g(x, y)$ is a quadratization of f. Indeed, g is clearly quadratic, because

G is. Moreover, for every point $x \in \{0,1\}^n$,

$$\begin{aligned}
&\min\big\{g(x,y) : y \in \{0,1\}^m\big\} \\
&\quad = \min\big\{G(x_1, x_2, x_2, x_3, x_3, x_3, x_3, \ldots, x_n, y) : y \in \{0,1\}^m\big\} && (3.65) \\
&\quad = F(x_1, x_2, x_2, x_3, x_3, x_3, x_3, \ldots, x_n) && (3.66) \\
&\quad = k\left(\sum_{i=1}^{n} 2^{i-1} x_i\right) && (3.67) \\
&\quad = f(x), && (3.68)
\end{aligned}$$

where equality (3.65) is by definition of g, (3.66) is by definition of G, (3.67) is by definition of F, and (3.68) is by definition of k. $\square$

We now can use the lower bounds for general pseudo-Boolean functions provided at the beginning of the section in order to obtain lower bound results for symmetric ones.

Theorem 3.63. *There exist symmetric pseudo-Boolean functions of n variables for which any quadratization must involve at least $\Omega(\sqrt{n})$ auxiliary variables.*

Proof. Lemma 3.62 shows that, if every symmetric function $F(z)$ on $\{0,1\}^N$, with $N = 2^n - 1$, has an m-quadratization, then every (arbitrary) function $f(x)$ on $\{0,1\}^n$ also has an m-quadratization. On the other hand, from Theorem 3.58, we know that some pseudo-Boolean functions on n variables require $\Omega(2^{n/2})$ auxiliary variables. It follows that some symmetric pseudo-Boolean functions on N variables must need $\Omega(\sqrt{N})$ auxiliary variables in every quadratization. $\square$

Theorem 3.64. *There exist symmetric pseudo-Boolean functions of n variables for which any y-linear quadratization must involve at least $\Omega(n/\log n)$ auxiliary variables.*

Proof. The proof is similar to the previous one: it suffices to observe that when $G(z,y)$ is y-linear, then so is $g(x,y)$, and to rely on the generic lower bound $\Omega(2^n/n)$ of Theorem 3.60 for the number of auxiliary variables required in every y-linear quadratization of certain pseudo-Boolean functions. $\square$

Note that the lower bound in Theorem 3.64 for the number of auxiliary variables in y-linear quadratizations comes within a factor $O(\log n)$ of the upper bound of $n-2$ from Theorem 3.33.

A Lower Bound for the Parity Function

The results just obtained prove the existence of symmetric pseudo-Boolean functions which require a significant number of auxiliary variables to quadratize. Specifically, there exist functions needing $\Omega(\sqrt{n})$ auxiliary variables in any quadratization, and functions needing $\Omega(n/\log n)$ auxiliary variables in any y-linear quadratization. Those results do not, however, explicitly exhibit particular such functions. We next give a concrete example of a function which needs a significant number of auxiliary variables in any y-linear quadratization.

Theorem 3.65. *Every y-linear quadratization of the parity function on n variables must involve at least $\Omega(\sqrt{n})$ auxiliary variables.*

Proof. Let $g(x,y)$ be an arbitrary y-linear quadratization of the parity function. Then it can be written as

$$g(x,y) = q(x) + \sum_{i=1}^{m} y_i(\ell_i(x) - b_i) \tag{3.69}$$

where $q(x)$ is quadratic, and $\ell_1(x), \ldots, \ell_m(x)$ are linear functions of x only.

For each $i \in [m] = \{1, 2, \ldots, m\}$, consider the regions

$$R_i^+ = \{x \in \mathbb{R}^n : \ell_i(x) \geq b_i\},$$

and

$$R_i^- = \{x \in \mathbb{R}^n : \ell_i(x) \leq b_i\},$$

which are closed half-spaces defined by the linear functions ℓ_i. For each $S \subseteq [m]$, let R_S denote the region

$$R_S = \left(\bigcap_{i \in S} R_i^-\right) \cap \left(\bigcap_{i \notin S} R_i^+\right).$$

This is one of the 'cells' into which the m hyperplanes defining the linear functions ℓ_i partition $\mathbb{R}^n$.

On every cell R_S, the function $f(x) = \min\{g(x, y) : y \in \{0,1\}^m\}$ is quadratic. Indeed, on R_S, we have

$$\min\bigl\{g(x, y) : y \in \{0,1\}^m\bigr\} = q(x) + \sum_{i \in S} (\ell_i(x) - b_i).$$

We now use a result from Saks [131] and Impagliazzo, Paturi and Saks [90] (which was used to obtain lower bounds on the size of threshold circuits representing the parity function). Let us say that a set of hyperplanes *slices* all r-dimensional subcubes of the Boolean hypercube $\{0,1\}^n$ if for each subcube (or face) of $\{0,1\}^n$ of dimension r, there are two vertices of the subcube that lie on opposite sides of one of these hyperplanes. Then (Proposition 3.82 of Saks [131]), if a set of m hyperplanes slices all r-dimensional subcubes, we have $m > \sqrt{n/(r+1) - 1}$. In particular, therefore, any set of hyperplanes that slices every 3-dimensional subcube of $\{0,1\}^n$ must contain more than $\sqrt{n/4 - 1}$ planes.

Suppose the hyperplanes defined by the linear functions ℓ_i do *not* slice all 3-dimensional subcubes. Then there would be some cell R_S containing a subcube of dimension 3. The parity function restricted to that subcube would then be equal to the quadratic expression $q(x) + \sum_{i \in S}(\ell_i(x) - b_i)$.

However, it is well-known (see, for instance Saks [131], Minsky and Papert [118], Wang and Williams[147]) that the parity function on a subcube of dimension r cannot be represented as a pseudo-Boolean function of degree less than r (and it cannot even be represented as the sign of a pseudo-Boolean function of degree less than r). So, we would then have a quadratic representation of parity on a cube of dimension 3, which is not possible.

It follows, therefore, that the set of hyperplanes in question must slice all 3-dimensional subcubes and therefore has size $m > \sqrt{n/4 - 1} = \Omega(\sqrt{n})$. $\square$

3.8 A Polyhedral Cone of Quadratizations

We have already seen in Section 3.7 that a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ can be represented by a vector in $\mathbb{R}^{2^n}$ whose entries are the values of f. There is another interesting vector in $\mathbb{R}^{2^n}$ that can be associated to f: the vector whose entries are the coefficients of its unique multilinear polynomial representation. Let us call this vector the *spectrum* of f. Clearly there is an one-to-one and onto correspondence between the two linear spaces that can be defined, but we shall not dwell on that. Our goal is to introduce the following object.

Definition 3.66. *Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-Boolean function in n variables and let m be an upper bound on the number of auxiliary variables of (some of) the quadratizations $g : \{0,1\}^{n+m} \to \mathbb{R}$ of f. Then for $\delta(n,m) = 1 + n + m + \binom{n+m}{2}$, the set*

$$\mathscr{P}_f(n,m) := \Big\{ g(x,y) \in \mathbb{R}^{\delta(n,m)} : g(x,y) \text{ is quadratic and } \\ f(x) \le g(x,y) \text{ for all } x \in \{0,1\}^n, y \in \{0,1\}^m \Big\}, \quad (3.70)$$

where each vector is the spectrum of a quadratic pseudo-Boolean function that majorates f, is called the polyhedral cone of quadratizations of f.

Notice that $\mathscr{P}_f(n,m)$ is defined by 2^{n+m} linear constraints, and for any two elements $x, y \in \mathscr{P}_f(n,m)$ and any $\lambda, \mu \in \mathbb{R}_{\ge 0}$, the conical combination $\lambda x + \mu y$ also belongs to $\mathscr{P}_f(n,m)$, as quadratic polynomials form an algebraic ring. Hence, $\mathscr{P}_f(n,m)$ is indeed a polyhedral cone.

As an example, if $f(x) = x_1x_2x_3$, the positive monomial in $n = 3$ variables, and we are interested in quadratizations with $m = 1$ auxiliary variable, $g(x,y)$ can be generically written as

$$g(x,y) = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4y + c_{12}x_1x_2 + c_{13}x_1x_3 + c_{23}x_2x_3 \\ + c_{14}x_1y + c_{24}x_2y + c_{34}x_3y,$$

and $\mathscr{P}_f(3,1)$ is then given by the vectors $(c_0, c_1, c_2, c_3, c_4, c_{12}, c_{13}, c_{23}, c_{14}, c_{24}, c_{34}) \in \mathbb{R}^{11}$

which satisfy the system:

$$\left\{\begin{array}{llllllllllll}
c_0 & & & & & & & & & & & \geq 0\\
c_0 & +c_1 & & & & & & & & & & \geq 0\\
c_0 & & +c_2 & & & & & & & & & \geq 0\\
c_0 & & & +c_3 & & & & & & & & \geq 0\\
c_0 & & & & +c_4 & & & & & & & \geq 0\\
c_0 & +c_1 & +c_2 & & & +c_{12} & & & & & & \geq 0\\
c_0 & +c_1 & & +c_3 & & & +c_{13} & & & & & \geq 0\\
c_0 & +c_1 & & & +c_4 & & & & +c_{14} & & & \geq 0\\
c_0 & & +c_2 & +c_3 & & & & +c_{23} & & & & \geq 0\\
c_0 & & +c_2 & & +c_4 & & & & & +c_{24} & & \geq 0\\
c_0 & & & +c_3 & +c_4 & & & & & & +c_{34} & \geq 0\\
c_0 & +c_1 & +c_2 & & +c_4 & +c_{12} & & & +c_{14} & +c_{24} & & \geq 0\\
c_0 & +c_1 & & +c_3 & +c_4 & & +c_{13} & & +c_{14} & & +c_{34} & \geq 0\\
c_0 & & +c_2 & +c_3 & +c_4 & & & +c_{23} & & +c_{24} & +c_{34} & \geq 0\\
c_0 & +c_1 & +c_2 & +c_3 & & +c_{12} & +c_{13} & +c_{23} & & & & \geq 1\\
c_0 & +c_1 & +c_2 & +c_3 & +c_4 & +c_{12} & +c_{13} & +c_{23} & +c_{14} & +c_{24} & +c_{34} & \geq 1
\end{array}\right\}.$$

We generated polyhedral cones of quadratizations of positive and negative monomials for some small values of n and m (through the development of a simple computer code) and fed their descriptions, similar to the one presented above, to Avis' *lrs* software package, which generates the vertices and rays of a polyhedron (see Avis [14], or visit http://cgm.cs.mcgill.ca/˜avis/C/lrs.html). We present the values obtained for some pairs (n, m) in the table below.

n	m	# vertices (pos)	# rays (pos)	# vertices (neg)	# rays (neg)
2	1	1	16	1	16
2	2	1	56	1	56
3	1	12	109	12	148
3	2	292	2109	292	3808
4	1	328	4807	353	6024
4	2	100824	1677491	74107	1484561
5	1	95954	1448766	52902	1187510

Observe that the number of extreme rays is always larger (by at least one order of magnitude) than the number of vertices. This suggests, in spite of the availability of data for very low dimensions only[1], that polyhedral cones of quadratizations of monomials are narrow.

An interesting fact, perhaps, is that not every vertex is a quadratization. In fact, we obtained the following values, where up to isomorphism in this context means up to permutation of the x and y variables (inside each group and altogether) and up to complementation of each variable.

$f(x)$	# quadratizations	$(2,1)$	$(2,2)$	$(3,1)$	$(3,2)$	$(4,1)$	$(4,2)$	$(5,1)$
positive	total:	1	1	8	288	2	5048	0
monomial	up to isomorphism:	1	1	2	28	1	138	0
negative	total:	1	1	8	288	10	3504	12
monomial	up to isomorphism:	1	1	2	28	2	91	2

We shall list the quadratizations up to isomorphism indicated in the above table in Section 3.11. For now, we show the two quaratizations with 1 auxiliary variable

[1]We tried to go beyond 6 variables in total, but two full weeks of computation on a MacBook Pro equipped with a 2.7 GHz Intel Core i7 processor, 8 GB of DDR3 RAM memory, and running MacOS X Lion were not sufficient for *lrs* to output a single vertex. Simple and crude estimations indicated that at least 3 full moths of dedicated computational time would be necessary for 7 variables.

obtained to $-x_1x_2x_3$, the negative monomial in 3 variables:

$$g_1^-(x,y) = 2y_1 - x_1y_1 - x_2y_1 - x_3y_1, \tag{3.71}$$

and

$$g_2^-(x,y) = x_2 + x_3 - x_1x_2 - x_1x_3 + x_1y_1 - x_2y_1 - x_3y_1. \tag{3.72}$$

Equation (3.71) is the standard quadratization of Freedman and Drineas (3.17). Equation (3.72) is a non-submodular quadratization of a submodular function, and has the form of what we have been calling (and shall justify in the next section) by extended standard quadratization of negative monomials.

For $x_1x_2x_3$, the positive monomial in 3 variables, the two 1-quadratizations are

$$g_1^+(x,y) = y_1 + x_2x_3 + x_1y_1 - x_2y_1 - x_3y_1, \tag{3.73}$$

and

$$g_2^+(x,y) = y_1 + x_1x_2 + x_1x_3 + x_2x_3 - x_1y_1 - x_2y_1 - x_3y_1. \tag{3.74}$$

The first is obtained from Proposition 3.14, and the second through Ishikawa's HOCR method depicted in Equation (3.22).

The fact that permutations of (original and auxiliary) variables, complementation of auxiliary variables, and that quadratizations added up with extremal rays give rise to different, but in some sense equivalent quadratizations motivates the following definitions.

Definition 3.67. *Two quadratizations $g(x,y)$ and $h(x,y)$ are called* switch-equivalent *if $g(x,u,v) = h(x,u,\overline{v})$ for an appropriate partition of y into two subsets of variables u, v.*

Definition 3.68. *A quadratization $g(x,y)$ of $f(x)$ is* lean *if there is no quadratization of f which involves fewer auxiliary variables than g.*

Definition 3.69. *A quadratization $g(x,y)$ of $f(x)$ is* prime *if there is no other quadratization of f, say $h(x,y)$, such that $h(x,y) \leq g(x,y)$ for all $(x,y) \in \{0,1\}^{n+m}$, and such that $h(x^*,y^*) < g(x^*,y^*)$ for at least one point (x^*,y^*).*

The idea behind the above definitions would be to restrict our attention to "minimal" or "elementary" quadratizations while studying their properties.

The large number of vertices and extremal rays found for monomials in few variables and quadratizations with few auxiliary variables, suggests that a complete characterization of vertices, extremal rays, faces, and facets of the polyhedral cone of quadratizations may be difficult to accomplish — even for small classes of pseudo-Boolean functions. Nevertheless, we are able to prove the following.

Theorem 3.70. *Every m-quadratization of a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is on the boundary of $\mathscr{P}_f(n,m)$.*

Proof. Let $g^* : \{0,1\}^{n+m} \to \mathbb{R}$ be an m-quadratization of f and let

$$h(y) = \big(h_1(y), h_2(y), \ldots, h_m(y)\big),$$

be the Boolean function associated to g^* as in Equation (3.41), so that $g^*(x,y) = g^*(x, h(y))$. That is, each $h_i : \{0,1\}^m \to \{0,1\}$ is associated to the auxiliary variable y_i of g^*.

Consider the linear programming problem

$$\min \left\{ \sum_{x \in \{0,1\}^n} g(x, h(y)) : g \in \mathscr{P}_f(n,m) \right\}, \tag{3.75}$$

whose minimum value is lower bounded by $\sum_{x \in \{0,1\}^n} f(x)$. Since $g^* \in \mathscr{P}_f(n,m)$ and since

$$f(x) = \min_{y \in \{0,1\}^m} g^*(x, h(y)) \quad \text{for all} \quad x \in \{0,1\}^n,$$

we have that g^* is a minimum point for Problem (3.75) of value $\sum_{x \in \{0,1\}^n} f(x)$.

It is known that if Problem 3.75 admits a minimum, then there is a minimum point on its boundary (cf. Korte and Vygen [104], Schrijver [132]). Suppose that g^* belongs to the interior of $\mathscr{P}_f(n,m)$. Then, less than $\delta(n,m)$ of the constraints defining $\mathscr{P}_f(n,m)$ are satisfied with equality by g^*. That implies there is an $x' \in \{0,1\}^n$ and an $y' \in \{0,1\}^m$ such that $f(x') < g^*(x',y')$ — as some of the coefficients of g^* are slightly larger than they would be if the aforementioned constraint were satisfied with equality.

But this contradicts the facts that g^* is a quadratization of f and that it is a minimum of Problem (3.75). Hence, g^* is on the boundary of $\mathscr{P}_f(n,m)$. $\square$

Recently, Crama and Rodríguez-Heck [52] communicated the following result.

Theorem 3.71 (Crama and Rodríguez-Heck [52])**.** *Every prime m-quadratization of a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is a vertex of $\mathscr{P}_f(n,m)$.* $\square$

3.9 A Characterization of 1-Quadratizations of Negative Monomials

Based on the computational experiments we mentioned in the previous section, we were able to have a better picture of how quadratizations of low-degree monomials look like. Since there is a plethora of termwise quadratization techniques out there, it is important to understand what are the best possible quadratizations of a single monomial. In this section, we are now going to characterize all lean prime quadratizations of the negative monomials.

Definition 3.72. *The* standard quadratization *of the negative monomial*

$$M_n = -\prod_{i=1}^{n} x_i$$

is the quadratic function

$$s_n(x,y) = (n-1)y - \sum_{i=1}^{n} x_i y. \tag{3.76}$$

The extended standard quadratization *of M_n is the function*

$$s_n^+(x,y) = (n-2)x_n y - \sum_{i=1}^{n-1} x_i(y - \overline{x}_n). \tag{3.77}$$

Proposition 3.73. *For all $n \geq 1$, the functions $s_n = s(x,y)$ and $s_n^+ = s^+(x,y)$ are quadratizations of $M_n = -\prod_{i=1}^{n} x_i$.*

Proof. The standard quadratization s_n was already introduced in Section 3.2.2 (Equation (3.17), Freedman and Drineas [69]). For s_n^+, the case $n = 1$ is trivial. For $n \geq 2$, fix $x \in \{0,1\}^n$ and suppose first that $x_n = 0$. Then, $M_n(x) = 0$ and

$$\min_y s_n^+(x,y) = \min_y (1-y) \sum_{i=1}^{n-1} x_i = 0$$

is attained for $y = 1$. On the other hand, if $x_n = 1$, then $s_n^+(x, y) = s_{n-1}(x, y)$ for all $y \in \{0, 1\}$, and the statement follows from the fact that $s_{n-1}(x, y)$ is a quadratization of M_{n-1}. □

It is not hard to see that s_n and s_n^+ are lean quadratizations of M_n since they use a single auxiliary variable. When $n \leq 2$, M_n is quadratic and, clearly, it is its own unique prime quadratization. For $n > 2$, we intend to prove that s_n and s_n^+ are essentially the only prime 1-quadratizations of M_n.

Theorem 3.74. *For $n \geq 3$, assume that $g(x, y)$ is a prime 1-quadratization of M_n. Then, up to an appropriate permutation of the x-variables and up to a possible switch of the y-variable, either $g(x, y) = s_n(x, y)$ or $g(x, y) = s_n^+(x, y)$.*

Proof. The proof involves a detailed analysis which turns out to be different according to whether $n = 3$ or $n \geq 4$. For the sake of brevity, we restrict ourselves here to the generic case $n \geq 4$ and we refer the reader to the technical report of crama and Rodríguez-Heck [53] for the special case $n = 3$.

So, assume now that $n \geq 4$ and that $g(x, y)$ is a 1-quadratization of M_n. Since $M_n(x) = \min_{y \in \{0,1\}} g(x, y)$ for all binary vectors x, we can assume $g(0, 0) = 0$ after substituting $\overline{y}$ for y if necessary. Thus, without any loss of generality we can write

$$g(x, y) = ay + \sum_{i=1}^{n} b_i x_i y + \sum_{i=1}^{n} c_i x_i + \sum_{1 \leq i < j \leq n} p_{ij} x_i x_j. \tag{3.78}$$

Let us introduce some useful notations. For any subset $S \subseteq N = [n]$, we write

$$b(S) = \sum_{i \in S} b_i, \quad c(S) = \sum_{i \in S} c_i, \quad \text{and} \quad p(S) = \sum_{i,j \in S,\, i<j} p_{ij}.$$

Furthermore, since binary vectors can be viewed as characteristic vectors of subsets, we simply write

$$g(S, y) = ay + b(S)y + c(S) + p(S)$$

instead of Equation (3.78), when x is the characteristic vector of S.

Then, the fact that g is a quadratization of M_n can be expressed as

$$0 = \min_{y\in\{0,1\}} (a + b(S))y + c(S) + p(S), \qquad \text{for all } S \subset N, \tag{3.79}$$

$$-1 = \min_{y\in\{0,1\}} (a + b(N))y + c(N) + p(N). \tag{3.80}$$

Let us now note that by Equation (3.79), we have $g(0, 1) \geq 0$, and hence

$$a \geq 0. \tag{3.81}$$

Furthermore, we must have $g(\{i\}, 0) \geq 0$ for all $i \in N$ since $n > 1$, implying

$$c_i \geq 0 \quad \text{for all} \quad i \in N. \tag{3.82}$$

Let us partition the set of indices as $N = N^0 \cup N^+$, where

$$N^0 = \{u \in N \mid c_u = 0\}, \tag{3.83}$$

$$N^+ = \{i \in N \mid c_i > 0\}. \tag{3.84}$$

Since $g(\{i\}, 0) = c_i$, relation (3.79) implies

$$g(\{i\}, 1) = a + b_i + c_i = 0 \quad \text{for all} \quad i \in N^+, \text{ and} \tag{3.85}$$

$$g(\{u\}, 1) = a + b_u \geq 0 \quad \text{for all} \quad u \in N^0. \tag{3.86}$$

Let us next write Equation (3.79) for subsets of size two. Consider first a pair $u, v \in N^0$, $u \neq v$. Since $c_u = c_v = 0$, we get $g(\{u, v\}, y) = (a + b_u + b_v)y + p_{uv}$, implying

$$\min\{p_{uv}, a + b_u + b_v + p_{uv}\} = 0. \tag{3.87}$$

Let us consider next $i, j \in N^+$, $i \neq j$. Then, by Equation (3.85) and by the definitions we get $g(\{i, j\}, 1) = p_{ij} - a \geq 0$. This, together with Equation (3.81) implies that $p_{ij} \geq a \geq 0$. Thus, $g(\{i, j\}, 0) = c_i + c_j + p_{ij} > 0$ implying that $g(\{i, j\}, 1) = 0$, that is,

$$p_{ij} = a \geq 0 \quad \text{for all} \quad i, j \in N^+. \tag{3.88}$$

This allows us to establish a first property of N^0.

Claim 1. $N^0 \neq \emptyset$.

Proof. If $N^0 = \emptyset$, then we have $g(N, y) = (a + b(N^+))y + c(N^+) + \binom{|N^+|}{2}a$ by Equation (3.88). Since $|N^+|a + b(N^+) + c(N^+) = 0$ by Equation (3.85), we get $g(N, 1) = \binom{|N^+|-1}{2}a \geq 0$ by Equation (3.81), and $g(N, 0) = c(N^+) + \binom{|N^+|}{2}a \geq 0$ by Equations (3.81) and (3.82). This contradicts Equation (3.80) and proves the claim. $\square$

In contrast with Claim 1, the set N^+ may be empty or not.

Claim 2. *If $N^+ = \emptyset$ and $N = N^0$, then $p_{uv} = 0$ for all $u, v \in N$. Furthermore,*

$$\begin{aligned} &a + b(S) \geq 0 \quad \textit{for all subsets} \quad S \neq N, \ \textit{and} \\ &a + b(N) = -1. \end{aligned}$$

Proof. Assume that $u, v \in N$ are such that $p_{uv} > 0$ (we know by Equation (3.87) that $p_{uv} \geq 0$). Then for any subset $S \subseteq N$ such that $u, v \in S$, we have $g(S, 0) = p(S) > 0$ and hence it must be the case that $g(S, 1) = 0$ if $S \neq N$ and $g(N, 1) = -1$.

Introducing the set function $d(S) = a + b(S) + p(S)$, we can write the above implications as

$$\begin{aligned} d(S) &= 0 \text{ if } u, v \in S \neq N, \text{ and,} \\ d(N) &= -1. \end{aligned}$$

Let us consider now two arbitrary elements $w, t \in N$ different from u and v. Let $X = N \setminus \{w\}$, $Y = N \setminus \{t\}$ and $Z = X \cap Y = N \setminus \{w, t\}$.

Then, by the above equalities, we have $-1 = d(N) - d(Y) = b_t + \sum_{k \neq t} p_{tk}$. Similarly, we have $0 = d(X) - d(Z) = b_t + \sum_{k \neq t,w} p_{tk}$. Taking the difference of these two expressions we get $-1 = p_{tw}$ contradicting Equation (3.87).

Thus, we have $p_{uv} = 0$ for all $u, v \in N$. Finally, the claimed inequalities and equality follow from Equations (3.79)–(3.80). $\square$

The previous relations allow us to establish a first case of Theorem 3.74.

Claim 3. *The statement of Theorem 3.74 holds when $N^+ = \emptyset$ and $N = N^0$.*

Proof. If $N = N^0$, then $c(S) = 0$ for all $S \subseteq N$ by definition and, by Claim 2, $p(S) = 0$ for all $S \subseteq N$, and $a = -1 - b(N)$. Therefore, we can write

$$g(x, y) = (-1 - b(N))y + \sum_{u \in N} b_u x_u y.$$

Since $s_n(x, y) = (n - 1)y - \sum_{u \in N} x_u y$, we obtain

$$g(x, y) - s_n(x, y) = (-n - b(N))y + \sum_{u \in N} (b_u + 1)x_u y = \sum_{u \in N} (-1 - b_u)y\overline{x}_u. \tag{3.89}$$

The relations $a + b(N \setminus \{u\}) \geq 0$ and $a + b(N) = -1$ imply that $b_u \leq -1$ for all $u \in N$. Hence, the right-hand side of Equation (3.89) is always nonnegative, and if g is prime, then it must be the case that $g = s_n$. □

From now on, let us assume that $|N^+| \geq 1$. Consider $u \in N^0$ and $i \in N^+$. We get $g(\{u, i\}, 0) = c_i + p_{ui}$, and in light of Equation (3.85), $g(\{u, i\}, 1) = b_u + p_{ui}$. Thus, we can write $N^0 \times N^+ = E_B \cup E_C$, where

$$E_B = \{(u, i) : u \in N^0,\ i \in N^+,\ p_{ui} = -b_u\},\ \text{and} \tag{3.90}$$

$$E_C = \{(u, i) : u \in N^0,\ i \in N^+,\ p_{ui} = -c_i\}. \tag{3.91}$$

We show next some properties of E_B, E_C, which will be useful to complete the proof of the main theorem.

We use several times the following identity: when $u \in N^0$ and $i, j \in N^+$, since $p_{ij} = a$ by Equation (3.88), we have

$$g(\{u, i, j\}, y) = (a + b_u + b_i + b_j)y + c_i + c_j + p_{ui} + p_{uj} + a. \tag{3.92}$$

Claim 4. *For all $u \in N^0$, we have either $\{u\} \times N^+ \subseteq E_B$, or $\{u\} \times N^+ \subseteq E_C$.*

Proof. Assume that this is not the case, so that there exist $u \in N^0$ and $i, j \in N^+$ such that $(u, i) \in E_B$ and $(u, j) \in E_C$. Then, since $|N| > 3$, we have $0 \leq g(\{u, i, j\}, 1)$. By Equation (3.85) we have $a + b_i + c_i = a + b_j + c_j = 0$, by Equation (3.91) $c_j + p_{uj} = 0$, and by Equation (3.90) $b_u + p_{ui} = 0$. Thus, Equation (3.92) yields $0 \leq g(\{u, i, j\}, 1) = a + b_j = -c_j$. But this contradicts $j \in N^+$. □

Consider the sets

$$B = \{u \in N^0 : \{u\} \times N^+ \subseteq E_B\}, \text{ and} \tag{3.93}$$

$$C = \{u \in N^0 : \{u\} \times N^+ \subseteq E_C\}. \tag{3.94}$$

The proof of Claim 4 actually establishes the following statement.

Claim 5. *$B \cup C = N^0$ and, if $|N^+| \geq 2$, then $B \cap C = \emptyset$.*

Thus, (B, C) forms a partition of N^0 when $|N^+| \geq 2$. But this is not necessarily true when $|N^+| = 1$. Let us now establish some auxiliary properties of the sets B and C.

Claim 6. *If $|N^+| \geq 1$, then $|C| \leq 1$, and either $B \cap C = \emptyset$ or $B = N^0$.*

Proof. Assume that $i \in N^+$ and $u, v \in C, u \neq v$. Then $b_u \geq c_i = -p_{ui}$ and $b_v \geq c_i = -p_{vi}$. Hence, $a + b_u + b_v + p_{uv} \geq a + 2c_i + p_{uv} > p_{uv}$ and by Equation (3.87), we must have $p_{uv} = 0$. Then, from Equation (3.79), $0 \leq g(\{u, v, i\}, 0) = c_i + p_{uv} + p_{ui} + p_{vi} = -c_i$, which contradicts the definition of N^+. This proves that $|C| \leq 1$.

If $B \cap C \neq \emptyset$, then $C \subseteq B$, and hence $B = N^0$. □

Claim 7. *If $|N^+| \geq 1$, $u \in B$, $v \in C \setminus B$, and $u \neq v$, then $p_{uv} = b_u$.*

Proof. Let $i \in N^+$. According to the definitions, $g(\{u, v, i\}, y) = (a + b_u + b_v + b_i)y - b_u + p_{uv}$. By definition of C, $b_v - c_i = b_v + p_{vi}$ and since $v \notin B$, $b_v + p_{vi} > 0$.

By Equation (3.85) we have $a + b_i = -c_i$, and hence we get $g(\{u, v, i\}, 1) = b_v - c_i + p_{uv}$. Thus, $g(\{u, v, i\}, 1) > 0$, since $p_{uv} \geq 0$ by Equation (3.87). Consequently, $g(\{u, v, i\}, 0) = -b_u + p_{uv} = 0$ proving the claim. □

Claim 8. *If $|N^+| \geq 2$, then $B = \emptyset$ and $C = N^0$.*

Proof. Assume by contradiction that $B \neq \emptyset$. Let us consider an arbitrary $u \in B$ and $i, j \in N^+$, $i \neq j$. Then, we have $g(\{u, i, j\}, 1) = -b_u$ by Equations (3.92), (3.85), (3.88) and the definition of B. Thus we have $-b_u \geq 0$ from which $g(\{u, i, j\}, 0) = c_i + c_j - 2b_u + a > 0$ follows, implying that we must have $g(\{u, i, j\}, 1) = -b_u = 0$.

Hence $p_{ui} = 0$ for all $i \in N^+$, by definition of B. Also, for all $v \in C$, Claim 5 implies that $v \notin B$, and by Claim 7, $p_{uv} = b_u = 0$.

Assume now that $|B| = 1$, $B = \{u\}$. Then, all terms of Equation (3.78) containing x_u vanish, since $b_u = c_u = p_{ui} = p_{uv} = 0$ for all $i \in N^+$, $v \in C$. Thus, x_u does not appear in g, a contradiction with the fact that M_n depends on all its variables.

On the other hand, if $|B| > 1$ and $v \in B$, $v \neq u$, then $g(\{u, v, i, j\}, y) = (a + b_i + b_j)y + c_i + c_j + a + p_{uv}$. Here a and p_{uv} are nonnegative by Equations (3.81) and (3.87), and c_i and c_j are both positive by the definition of N^+, therefore $g(\{u, v, i, j\}, 0) > 0$. Thus, $g(\{u, v, i, j\}, 1) = p_{uv} \leq 0$ follows by Equations (3.79) and (3.80). Since $p_{uv} \geq 0$ by Equation (3.87), $p_{uv} = 0$ follows. Consequently, $b_u = p_{uv} = 0$ follows for all $u \in N^0$ and $v \neq u$, implying again that x_u does not play any role in g, which is a contradiction and proves our claim. □

Claim 9. *If $|N^+| \geq 2$, then $|C| = 1$, $|N^+| = n - 1$, and $a = b_i + c_i = p_{ij} = 0$ for all $i, j \in N^+$.*

Proof. When $|N^+| \geq 2$, Claim 6 and Claim 8 together imply that $B = \emptyset$, $C = N^0$, and $|C| \leq 1$. Since $N^0 \neq \emptyset$ by Claim 1, it follows that $|C| = 1$ and $|N^+| = n - 1$.

We assumed $|N| \geq 4$. So, let $i, j, k \in N^+$ be three distinct indices. Then

$$g(\{i, j, k\}, 0) = c_i + c_j + c_k + 3a > 0$$

by Equation (3.88), by definition of N^+ and by Equation (3.81). Thus, we must have $g(\{i, j, k\}, 1) = 0$ by Equation (3.79). By Equation (3.85), this implies $a = 0$, and the claim follows by Equation (3.88). □

Claim 10. *If $g(x, y)$ is a quadratization of M_n with $|N^+| \geq 2$, then $h(x, y) = g(x, \overline{y})$ is another quadratization of M_n with either $|N^+| = 1$ and $|B| = n - 1$, or $N^+ = \emptyset$ and $N = N^0$.*

Proof. This follows from the definitions and from Claim 9. □

In view of Claim 3 and Claim 9, up to switching the y-variable, we are left with the case $|N^+| = 1$.

Claim 11. *If $N^+ = \{i\}$, then $p_{uv} = 0$ for all $u, v \in B$.*

Proof. Let us assume there exist $u, v \in B$ such that $p_{uv} > 0$ (we know by Equation (3.87) that $p_{uv} \geq 0$.) Then $g(\{u,v,i\},1) = (a+b_u+b_v+b_i)+c_i+p_{uv}-b_u-b_v = p_{uv} > 0$ by Equation (3.85) and by the definition of B. Thus, $g(\{u,v,i\},0) = c_i+p_{uv}-b_u-b_v = 0$ follows by Equation (3.79). On the other hand, we have $g(\{u,v\},0) = p_{uv} > 0$ and thus $g(\{u,v\},1) = a+b_u+b_v+p_{uv} = 0$ follows again by Equation (3.79). Adding these two equalities, we get $a + c_i + 2p_{uv} = 0$ which is impossible since $a \geq 0$, $c_i > 0$ and $p_{uv} > 0$. $\square$

Claim 12. *If $N^+ = \{i\}$, then $|B| = n-1$. Furthermore, we have*

$$c_i = b(B) - 1, \quad \textit{and} \tag{3.95}$$

$$c_i \geq b(S) \quad \textit{for all subsets} \quad S \subseteq B,\ S \neq B. \tag{3.96}$$

Proof. Assume first that $|B| < n-1$. It follows from Claim 6 that $|C| = 1$ and $B \cap C = \emptyset$. Let $C = \{w\}$. We obtain

$$g(N,y) = (a + b(B) + b_w + b_i)y + c_i + p(B) + \sum_{u\in B} p_{uw} + \sum_{u\in B} p_{ui} + p_{wi}. \tag{3.97}$$

Now, $a + b_i = -c_i$ by Equation (3.85), $p(B) = 0$ by Claim 11, $\sum_{u\in B} p_{uw} = \sum_{u\in B} b_u$ by Claim 7, $\sum_{u\in B} p_{ui} = -\sum_{u\in B} b_u$ by definition of B, and $p_{wi} = -c_i$ by definition of C. Hence,

$$g(N,y) = (b(B) + b_w - c_i)y. \tag{3.98}$$

In view of Claim 7 and of Equation (3.87), $b_u = p_{uw} \geq 0$ for all $u \in B$. Moreover, $g(\{w,i\},1) = b_w + p_{wi} = b_w - c_i$ by definition of C, and hence $b_w - c_i \geq 0$. This implies that $g(N,y) \geq 0$ for all y, contradicting Equation (3.80).

Thus, $|B| = n-1$. In this case we obtain $g(N,1) = 0$ by definition of B, and thus we must have $g(N,0) = c_i - b(B) = -1$. Furthermore, for any subset $S \subseteq B$, $S \neq B$ we have $g(S,0) = c_i - b(S) \geq 0$. $\square$

We are now ready to prove the remaining case of Theorem 3.74.

Claim 13. *The statement of Theorem 3.74 holds when $|N^+| = 1$.*

Proof. In view of Claim 12, we can assume that $N^+ = \{n\}$ and that $B = \{1, 2, ..., n-1\}$. By Equation (3.85), by the definition of B and by Claim 11, we have $b_n = -a - c_n$, $p_{nu} = -b_u$ for all $u \in B$, and $p_{uv} = 0$ for all $u, v \in B$. Thus,

$$g(x, y) = ay\overline{x}_n + c_n x_n \overline{y} + \sum_{u \in B} b_u x_u (y - x_n).$$

Since $s_n^+(x, \overline{y}) = (n-2)x_n\overline{y} + \sum_{u \in B} x_u(y - x_n)$, we get

$$g(x, y) - s_n^+(x, \overline{y}) = ay\overline{x}_n + (c_n - n + 2)x_n\overline{y} + \sum_{u \in B}(b_u - 1)x_u(y - x_n).$$

By Equation (3.95), we have $\sum_{u \in B}(b_u - 1) = c_n - n + 2$. Hence, we can write

$$\begin{aligned} g(x, y) - s_n^+(x, \overline{y}) &= ay\overline{x}_n + \sum_{u \in B}(b_u - 1)[x_u(y - x_n) + x_n\overline{y}] \\ &= ay\overline{x}_n + \sum_{u \in B}(b_u - 1)[yx_u\overline{x}_n + \overline{y}\,\overline{x}_u x_n]. \end{aligned}$$

The relations (3.95)–(3.96) imply that $b_u \geq 1$ for all $u \in B$. Hence, $g(x, y) - s_n^+(x, \overline{y})$ is always nonnegative, and this completes the proof of the theorem. □

□

3.10 Concluding Remarks

In this chapter, we studied quadratizations of pseudo-Boolean functions, that is, transformations that given a pseudo-Boolean function $f : \{0, 1\}^n \to \mathbb{R}$ in n variables, produce a quadratic pseudo-Boolean function $g : \{0, 1\}^{n+m} \to \mathbb{R}$ in the original n variables plus m auxiliary variables such that $f(x) = \min_{y \in \{0,1\}^m} g(x, y)$ for all binary vectors $x \in \{0, 1\}^n$. Our motivation stemmed from the wide attention such technique has received in the past decade, being successfully employed to solve very large unconstrained nonlinear binary optimization problems, specially those originating form some imaging applications inside the computer vision community. Our goal was to obtain a better understanding of its strengths and weaknesses.

Upon our arrival, the scenario was dominated by termwise transformations, i.e., by procedures that quadratize a given pseudo-Boolean function one monomial at a time. We initially proposed a still termwise technique, inspired by the consensus operation used for logical inferences in Boolean functions represented in disjunctive normal forms (DNFs), that allows for multiple splits of a monomial, thus generalizing some of the existing techniques and also providing some new ones. We then introduced the first transformation taking into account common parts of different monomials, in what could be described as a factoring approach. Quadratizing many monomials together produced a lot less positive terms than the state of the art at the time and that had a practical impact. When empirically evaluated in problems as image restauration and stereo reconstruction, it performed very well: it was able to fix up to 96% of the variables to their provable optimum values through persistence, and it was a lot faster that its existing counterparts.

We then started to investigate quadratizations of a pseudo-Boolean function form a global standpoint, i.e., instead of focusing in quadratizing terms, concentrate on techniques to quadratize the function as a whole. We first studied the class of symmetric pseudo-Boolean functions, a small but important class of functions that include monomials. We showed a representation theorem and based on it we were able to derive that every symmetric pseudo-Boolean function in n variables can be quadratized with at most $n-2$ auxiliary variables, and such quadratizations are what we call y-linear, that is, they do not include products between the original variables. Afterwards, we showed that some popular symmetric pseudo-Boolean functions like t-out-of-n, exact-t, parity, and co-parity admit y-linear quadratizations with at most $\lceil n/2 \rceil$, $\lfloor n/2 \rfloor$, $\lfloor n/2 \rfloor$, and $\lfloor (n-1)/2 \rfloor$ auxiliary variables, respectively. We revisited quadratizations of positive monomials and were able to give a new proof that they can be quadratized with at most $\lfloor (n-1)/2 \rfloor$ auxiliary variables.

Applying the same mindset to general pseudo-Boolean functions, we first showed that every pseudo-Boolean function in n variables admits a canonical quadratization, namely, the minterm quadratization, in at most 2^n auxiliary variables — thus improving the previous existing upper bound of $O(n\,2^n)$ due to Ishikawa [91, 92]. With some nice results about decomposition and extension of the function with respect to subcubes of its Boolean hypercube, we slightly improved the above upper bound to $\frac{3}{8}2^n$, in what we called y-full quadratizations. We progressed introducing what is perhaps the central concept behind quadratizations: that of universal sets (which are closely related to 2-bases of hypergraphs). We showed the existence of universal sets and how they can be used to quadratize pseudo-Boolean functions in at most $O(2^{n/2})$ auxiliary variables; and if the function has bounded degree d, the number of auxiliary variables required in a quadratization is at most $O(n^{d/2})$. We then introduced attractive partitions, which can be constructed from Turán systems and showed that every pseudo-Boolean function in n variables admits a y-linear quadratization with at most $O\big(\frac{2^n}{n} \cdot \log n\big)$.

We then addressed the question of how many auxiliary variables are necessary to quadratize a pseudo-Boolean function if every possible way of doing it is considered. In other words, we went looking for lower bounds on the number of auxiliary variables. And we found them! We showed that almost all pseudo-Boolean functions in n variables require at least $\Omega(2^{n/2})$ and $\Omega(2^n/n)$ auxiliary variables in any quadratization and in any y-linear quadratization, respectively. If the pseudo-Boolean functions have bounded degree d, the lower bounds become $\Omega(n^{d/2})$ and $\Omega(n^{d-1})$, respectively. Comparing such bounds to the upper bounds of the previous paragraph, we can say that they are essentially tight for general quadratizations and almost tight (off by only a $\log n$ factor) for y-linear quadratizations. For symmetric pseudo-Boolean functions, we showed that there are functions whose quadratizations must involve at least $\Omega(\sqrt{n})$ auxiliary variables and whose y-linear quadratizations must involve at least $\Omega(n/\log n)$ auxiliary variables (leaving a $O(\log n)$ gap to our y-linear upper bound of $n-2$ auxiliary variables, mentioned two paragraphs above). Using different techniques, we showed a concrete example: every y-linear quadratization for the parity function in n variables requires at least $\Omega(\sqrt{n})$ auxiliary variables.

Finally, we defined a combinatorial / geometric object that we used to generate and investigate quadratizations of low-degree monomials: the polyhedral cone of quadratizations. Based upon computer experiments realized over it, we were able present a full characterization of quadratizations of negative monomials involving only one auxiliary variable. The complexity of such proof reveals the richness of details and the full intricacy behind these so easy to define transformations.

After all that was described above, our understanding of quadratizations is considerably better than when we first started. Nevertheless, a complete picture is still far from being fully rendered. We therefore, finish this chapter with a list of open questions and some extra considerations.

Closing the existing gaps between some of our lower and upper bounds is the first open question that comes to mind. Specifically, we have an $O(\log n)$ gap (mentioned above) for y-linear quadratizations of general pseudo-Boolean functions, and we believe closing it to be an interesting theoretical endeavor: our lower bound seems correct, so we believe improvements on the construction of Turán systems or perhaps, the use of a different combinatorial structure to be the way to go; the former sounds more plausible, nonetheless. We also have a logarithmic gap between the bounds for y-linear symmetric pseudo-Boolean functions and closing this gap does not seem out of reach, but new ideas will be necessary. Considering particular specimens, improving the lower bound for the parity function and/or providing a simliar bound for the all-popular majority function would be highly desirable, as these functions have a long and prominent history in Boolean circuit complexity (cf. Jukna [96]).

Another open question related to symmetric pseudo-Boolean functions is to determine whether a similar, or better lower bound can be obtained for non y-linear quadratizations. Notice that all the quadratizations we proposed for symmetric pseudo-Boolean functions are y-linear; any example of a symmetric function where a non y-linear quadratization needs fewer variables than the y-linear counterpart would be of interest.

An interesting venue of research consists in extending our work on symmetric pseudo-Boolean functions for larges classes of functions, like the weakly symmetric, the partially symmetric, and so on. We have already started investigating quadratizations of d-part symmetric pseudo-Boolean functions, and it is rather notable the relation between symmetry and the number of auxiliary variables: the fewer the former is available, the more of the latter is needed.

Regarding y-linear quadratizations of general pseudo-Boolean functions again, we believe it to be possible to tailor our algorithm to the case of bounded degree functions. This specialization is of interest as it could have good practical implications in solving some computer vision applications, where the pseudo-Boolean functions are locally dense and do have bounded degree. Along the same line, but slightly more general, a thorough empirical evaluation of our global methods could reveal positive results for those applications (even with sub-exponential number of auxiliary variables). It is worth mentioning that before the landing of pseudo-Boolean methods inside the computer vision community, many of their problems were tackled with variations of the belief-propagation algorithm. It is just natural to wonder if the combo quadratization techniques plus the QPBO algorithm can accomplish something similar in different realms, where belief-propagation reigns.

The number of positive quadratic terms is equal to $n-1$ in every known quadratization of the positive monomial in n variables, but no lower bound on such quantity has been found so far. Settling this question is of great interest, as it is related to the quality of relaxations based on quadratizations for PBO problems. We conducted preliminary studies on the quality of relaxations based on termwise quadratizations versus relaxations based on the classical linearization, and the results obtained so far favored the latter. Notice however that at the moment, we are still a bit far from being able to settle the score in either one's favor, and our new algorithms still need to be evaluated, as previously alluded. Nevertheless, finding classes of functions where (some species of) quadratizations beat the classical linearization is another highly interesting task, as it would be showing improvements upon the roof duality bound. Pushing the bar even further, we wonder if we can find some classes of pseudo-Boolean functions, other than the submodular ones, such that when quadratized, the QBPO algorithm solves them exactly, to optimality. Naturally, unless $\mathsf{P} = \mathsf{NP}$, a complete characterization of those classes is out of question, but any insight harvested would be valuable.

A better understanding of the polyhedral cone of quadratizations also figures as a worthy, open question. Even if a complete description may be out of reach, we might be able to collect revealing information about the structure of quadratizations for some specific pseudo-Boolean functions, or even some classes thereof. For instance, we wonder if we can find a characterization of quadratizations of positive monomials (or find any non constant lower bound on the number of auxiliary variables required), similarly to what was done to negative ones. Moreover, can a better understanding of the facial structure of the cone provide a simpler proof for the latter? What can be said about the elements of the polar cone? Can they provide good structural or algorithmic insights?

Another open question consists of determining the computational complexity of given two pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$ and $g : \{0,1\}^{n+m} \to \mathbb{R}$, whether g is a quadratization of f. Related to that, is determining the complexity of minimizing the number of auxiliary variables for a given pseudo-Boolean function. The latter problem seems to be very high in the polynomial hierarchy, and we conjecture it to be Σ_3^p-complete.

Finally, we see as a next reasonable step to try and extend the notion of quadratization to approximate settings. For instance, we could define

$$f(x) \leq \min_{y \in \{0,1\}^m} g(x,y) \leq \rho f(x) \quad \text{for all} \quad x \in \{0,1\}^n,$$

for $f(x) \geq 0$ and some constant $\rho \geq 1$, or

$$\left\| f(x) - \min_{y \in \{0,1\}^m} g(x,y) \right\|_{\ell} \leq \delta,$$

for $\ell = 1$ or $\ell = 2$ (squaring the LHS in this case) and some constant δ, or still

$$\frac{1}{2^n} \left| \left\{ x \in \{0,1\}^n : f(x) \neq \min_{y \in \{0,1\}^m} g(x,y) \right\} \right| \leq \varepsilon,$$

for some constant $0 < \varepsilon \leq 1$, preferably bounded away from 1. It is possible that we might be able to reduce the number of auxiliary variables required in quadratizations and in y-linear quadratizations in these approximate settings.

3.11 Appendix: List of Quadratizations of Low-Degree Monomials

Below, we present a list of 1- and 2-quadratizations up to isomorphism of cubic, quartic, and quintic negative and positive monomials, where isomorphism in this context means up to permutations of the x-variables, of the y-variables, of both altogether, and up to complementation of all variables.

- Quadratizations of $-x_1x_2x_3$ with one auxiliary variable y_1:

$$2y_1 - x_1y_1 - x_2y_1 - x_3y_1$$

$$x_2 + x_3 - x_1x_2 - x_1x_3 + x_1y_1 - x_2y_1 - x_3y_1$$

- Quadratizations of $-x_1x_2x_3$ with two auxiliary variables y_1, y_2:

$$2y_2 - x_1y_2 - x_2y_2 - x_3y_2$$

$$2y_1 + 2y_2 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_3 + 2y_2 - x_1x_3 + x_1y_1 - x_1y_2 - x_2y_2 - x_3y_1 - y_1y_2$$

$$x_2 + x_3 - x_1x_2 - x_1x_3 + x_1y_2 - x_2y_2 - x_3y_2$$

$$x_2 + x_3 + y_2 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_1 + x_2 + x_3 - x_1x_2 - x_1y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_1 + x_2 + x_3 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + 2y_1y_2$$

$$2y_2 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - y_1y_2$$

$$x_3 + y_2 - x_1x_3 + x_1y_1 + x_2y_1 - x_2y_2 - x_3y_2 - y_1y_2$$

$$x_3 + y_2 - x_1x_3 + x_1y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_3 + 2y_2 - x_1x_3 - x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - y_1y_2$$

$$x_2 + x_3 + y_2 - x_1x_2 - x_1x_3 - x_2x_3 + x_1y_1 - x_2y_1 + x_2y_2 - x_3y_2 - y_1y_2$$

$$x_2 + x_3 + y_2 - x_1x_2 - x_1x_3 + x_1y_1 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_2 + x_3 + y_2 - x_1x_2 - x_2x_3 + x_1y_1 - x_1y_2 - x_2y_1 + x_2y_2 - x_3y_2 - y_1y_2$$

$$2y_2 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - 2y_1y_2$$

$$2y_1 + 2y_2 + x_1x_3 + x_2x_3 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - 2x_3y_1 - 2x_3y_2 + y_1y_2$$

$$x_3 + 2y_2 - x_1x_3 - x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 + x_3y_2 - 2y_1y_2$$

$$x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_3y_2 + y_1y_2$$

$$x_2 + x_3 - x_1x_2 - x_1x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 + y_1y_2$$

$$x_2 + x_3 - x_1x_2 - x_1x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + 2y_1y_2$$

$$x_2 + x_3 + y_2 - x_1x_2 - x_1x_3 + x_1y_1 - x_1y_2 - x_2y_1 + x_2y_2 - x_3y_1 + x_3y_2 - y_1y_2$$

$$x_1 + x_2 + x_3 + y_2 - 2x_1x_2 - 2x_1x_3 + 2x_1y_1 - 2x_1y_2 - x_2y_1 + x_2y_2 - x_3y_1 + x_3y_2 - y_1y_2$$

$$1 - x_1 + 2x_2 + 2x_3 - y_1 - y_2 - 2x_1x_2 - 2x_1x_3 + 2x_1y_1 + 2x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$1 - x_1 + 2x_2 + 2x_3 - y_1 - y_2 - x_1x_2 - x_1x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$2 - x_1 - x_2 - x_3 - 2y_1 + 3y_2 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - y_1y_2$$

$$x_2 + x_3 + y_2 - x_1x_2 + x_1x_3 - 2x_2x_3 + x_1y_1 - x_1y_2 - x_2y_1 + x_2y_2 + 2x_3y_1 - 2x_3y_2 - y_1y_2$$

$$x_2 + 3x_3 - x_1x_2 - 2x_1x_3 + x_2x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_2y_2 - 2x_3y_1 - 2x_3y_2 + y_1y_2$$

$$2 - 2x_1 - x_2 - x_3 - 2y_1 + 3y_2 + x_1x_2 + x_1x_3 + 2x_1y_1 - 2x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - y_1y_2$$

- Quadratizations of $-x_1x_2x_3x_4$ with one auxiliary variable y_1:

$$3y_1 - x_1y_1 - x_2y_1 - x_3y_1 - x_4y_1$$

$$x_2 + x_3 + x_4 - x_1x_2 - x_1x_3 - x_1x_4 + 2x_1y_1 - x_2y_1 - x_3y_1 - x_4y_1$$

- Quadratizations of $-x_1x_2x_3x_4$ with two auxiliary variables y_1, y_2:

$$3y_2 - x_1y_2 - x_2y_2 - x_3y_2 - x_4y_2$$

$$2y_1 + 2y_2 - x_1y_1 - x_2y_1 - x_3y_2 - x_4y_2 - y_1y_2$$

$$3y_1 + 3y_2 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$x_4 + 3y_2 - x_1x_4 + x_1y_1 - x_1y_2 - x_2y_2 - x_3y_2 - x_4y_1 - y_1y_2$$

$$x_3 + x_4 + 2y_2 - x_1x_3 - x_1x_4 + 2x_1y_1 - x_1y_2 - x_2y_2 - x_3y_1 - x_4y_1 - y_1y_2$$

$$x_3 + x_4 + 2y_2 - x_1y_2 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$x_2 + x_3 + x_4 - x_1x_2 - x_1x_3 - x_1x_4 + 2x_1y_2 - x_2y_2 - x_3y_2 - x_4y_2$$

$$x_2 + x_3 + x_4 + y_2 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + 2y_1y_2$$

$$x_1 + x_2 + x_3 + x_4 - x_1x_2 - x_1y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + 2y_1y_2$$

$$x_1 + x_2 + x_3 + x_4 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + 3y_1y_2$$

$$3y_2 - x_1y_2 - x_2y_2 + x_3y_1 - x_3y_2 + x_4y_1 - x_4y_2 - y_1y_2$$

$$x_4 + 2y_2 - x_1x_4 + x_1y_1 - x_2y_2 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$x_2 + x_3 + x_4 + y_2 - x_1x_2 - x_1x_3 - x_1x_4 + x_1y_1 + x_1y_2 - x_2y_1 - x_3y_2 - x_4y_2 - y_1y_2$$

$$x_2 + x_3 + x_4 + y_2 - x_1x_2 - x_1x_3 - x_2x_4 + 2x_1y_1 - x_1y_2 - x_2y_1 + x_2y_2 - x_3y_1 - x_4y_2 - y_1y_2$$

$$x_2 + x_3 + x_4 + y_2 - x_1x_2 - x_2x_3 - x_2x_4 + x_1y_1 - x_1y_2 - x_2y_1 + 2x_2y_2 - x_3y_2 - x_4y_2 - y_1y_2$$

$$x_2 + x_3 + x_4 + 2y_2 - x_1x_2 - x_1x_3 - x_1x_4 + 2x_1y_1 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$3y_2 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 + x_4y_1 - x_4y_2 - 2y_1y_2$$

$$2y_1 + 3y_2 + x_2x_4 + x_3x_4 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - 2x_4y_1 - 2x_4y_2 + y_1y_2$$

$$x_4 + 2y_2 - x_1x_4 + x_1y_1 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - x_4y_2 - 2y_1y_2$$

$$x_4 + 2y_2 - x_1x_4 + x_1y_1 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 - y_1y_2$$

$$2x_4 + y_2 - x_1x_4 - x_2x_4 + x_1y_1 + x_2y_1 - x_3y_2 - 2x_4y_1 - x_4y_2 + y_1y_2$$

$$2x_4 + 3y_2 - x_1x_4 - x_2x_4 - x_3x_4 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - 2x_4y_1 - y_1y_2$$

$$x_3 + x_4 + y_2 - x_1x_3 - x_1x_4 + 2x_1y_1 + x_1y_2 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + 2y_1y_2$$

$$x_3 + 2x_4 + 2y_2 - x_1x_4 - x_2x_4 - x_3x_4 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_2 - 2x_4y_1 + 2x_4y_2 - 2y_1y_2$$

$$x_3 + 2x_4 + 2y_2 - x_1x_4 - x_2x_4 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - 2x_4y_1 + x_4y_2 - y_1y_2$$

$$x_2 + x_3 + x_4 - x_1x_2 - x_1x_3 - x_1x_4 + x_1y_1 + 2x_1y_2 - x_2y_1 - x_2y_2 - x_3y_2 - x_4y_2 + y_1y_2$$

$$x_2 + x_3 + x_4 - x_1x_2 - x_1x_3 - x_1x_4 + 2x_1y_1 + 2x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 + 2y_1y_2$$

$$x_2 + x_3 + x_4 - x_1x_2 - x_1x_3 - x_1x_4 + 2x_1y_1 + 2x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + 3y_1y_2$$

$$x_2 + x_3 + x_4 - x_1x_2 - x_2x_3 - x_2x_4 + x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - x_4y_1 + y_1y_2$$

$$x_2 + x_3 + x_4 + y_2 - x_1x_2 - x_1x_3 - x_1x_4 + 2x_1y_1 - x_1y_2 - x_2y_1 - x_3y_1 + x_3y_2 - x_4y_1 + x_4y_2 - y_1y_2$$

$$x_2 + x_3 + 2x_4 + y_2 - x_1x_2 - x_1x_3 - x_1x_4 - x_2x_4 - x_3x_4 + 2x_1y_1 - x_2y_1 + x_2y_2 - x_3y_1 + x_3y_2 - 2x_4y_2 - y_1y_2$$

$$1 - x_1 + x_2 + 2x_3 + 2x_4 - y_1 - y_2 - x_1x_2 - x_1x_3 - x_1x_4 + x_1y_1 + 2x_1y_2 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$1 - x_1 + 2x_2 + 2x_3 + 2x_4 - y_1 - y_2 - 2x_1x_2 - 2x_1x_3 - 2x_1x_4 + 3x_1y_1 + 3x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$1 + 2x_3 + 2x_4 - y_1 - y_2 - x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 + 2x_1y_2 + 2x_2y_1 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$\begin{aligned} &2 - 2x_1 + 2x_2 + 2x_3 + 2x_4 - 2y_1 - 2y_2 - x_1x_2 - x_1x_3 - x_1x_4 + 2x_1y_1 + 2x_1y_2 \\ &\qquad - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + 2y_1y_2 \end{aligned}$$

$$2 - x_1 - x_2 - 2y_1 + y_2 + x_1y_1 + x_2y_1 - x_3y_2 - x_4y_2 + y_1y_2$$

$$3y_2 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 + x_4y_1 - x_4y_2 - 3y_1y_2$$

$$2y_1 + 3y_2 + x_2x_3 + x_2x_4 + x_3x_4 - x_1y_1 - x_2y_1 - 2x_2y_2 - x_3y_1 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + y_1y_2$$

$$3y_1 + 3y_2 + x_1x_4 + x_2x_4 + x_3x_4 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - 3x_4y_1 - 3x_4y_2 + 2y_1y_2$$

$$2x_4 + 3y_2 - x_1x_4 - x_2x_4 - x_3x_4 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - 2x_4y_1 + 2x_4y_2 - 3y_1y_2$$

$$x_3 + x_4 + y_2 - x_1x_3 - x_1x_4 + 2x_1y_1 + x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$x_3 + x_4 + y_2 + x_1x_2 - x_1x_3 - x_2x_3 - x_3x_4 + 2x_1y_1 - x_1y_2 + 2x_2y_1 - x_2y_2 - x_3y_1 + x_3y_2 - x_4y_2 - y_1y_2$$

$$x_3 + x_4 + 3y_2 + x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 + 2x_1y_1 - 2x_1y_2 + x_2y_1 - 2x_2y_2 - x_3y_1 + x_3y_2 - x_4y_1 - 2y_1y_2$$

$$x_3 + x_4 + 3y_2 + x_1x_2 - x_1x_3 - x_2x_3 + x_1y_1 - 2x_1y_2 + x_2y_1 - 2x_2y_2 - x_3y_1 + x_3y_2 - x_4y_1 - x_4y_2 - y_1y_2$$

$$x_3 + 2x_4 - x_1x_3 - x_1x_4 - x_2x_4 + x_1y_1 + x_1y_2 + x_2y_2 - x_3y_1 - x_4y_1 - 2x_4y_2 + y_1y_2$$

$$x_2+x_3+x_4+2y_2-x_1x_2-x_1x_3-x_1x_4+2x_1y_1-2x_1y_2-x_2y_1+x_2y_2-x_3y_1+x_3y_2-x_4y_1+x_4y_2-2y_1y_2$$

$$x_2+2x_3+2x_4-x_1x_2-x_1x_3-x_1x_4+x_3x_4+x_1y_1+x_1y_2-x_2y_1-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+y_1y_2$$

$$x_1+x_2+x_3+2x_4+y_2-2x_1x_4-2x_2x_4-2x_3x_4-x_1y_1+x_1y_2-x_2y_1+x_2y_2-x_3y_1+x_3y_2+3x_4y_1-3x_4y_2-y_1y_2$$

$$3-x_1-x_2-x_3-x_4-3y_1+4y_2+x_1y_1-x_1y_2+x_2y_1-x_2y_2+x_3y_1-x_3y_2+x_4y_1-x_4y_2-y_1y_2$$

$$x_3+x_4+y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_1y_1-x_1y_2+2x_2y_1-x_2y_2-x_3y_1+x_3y_2-x_4y_1+x_4y_2-y_1y_2$$

$$x_3+x_4+2y_2-x_1x_3-x_1x_4+x_3x_4+x_1y_1+x_1y_2+x_2y_1-x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+y_1y_2$$

$$x_3+x_4+3y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_1y_1-2x_1y_2+2x_2y_1-2x_2y_2-x_3y_1+x_3y_2-x_4y_1+x_4y_2-3y_1y_2$$

$$x_3+x_4+y_1+2y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4-x_1y_1+3x_1y_2-x_2y_1+3x_2y_2+x_3y_1-2x_3y_2+x_4y_1-2x_4y_2-3y_1y_2$$

$$x_3+2x_4+y_2-x_1x_3-x_1x_4-x_2x_4+x_3x_4+x_1y_1+x_1y_2+x_2y_1-x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+2y_1y_2$$

$$x_2+x_3+2x_4+y_2-x_1x_2-x_1x_3-2x_1x_4+x_2x_4+x_3x_4+2x_1y_1+x_1y_2-x_2y_1-x_2y_2-x_3y_1-x_3y_2-2x_4y_1-2x_4y_2+y_1y_2$$

$$\begin{aligned}&x_2+x_3+2x_4+2y_2-x_1x_2-x_1x_3-2x_1x_4+x_2x_4+x_3x_4+2x_1y_1+3x_1y_2\\&\qquad-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-2x_4y_1-4x_4y_2+4y_1y_2\end{aligned}$$

$$x_2+x_3+4x_4-x_1x_2-x_1x_3-3x_1x_4+x_2x_4+x_3x_4+2x_1y_1+2x_1y_2-x_2y_1-x_2y_2-x_3y_1-x_3y_2-3x_4y_1-3x_4y_2+2y_1y_2$$

$$2x_2+2x_3+2x_4-x_1x_2-x_1x_3-x_1x_4+x_2x_4+x_3x_4+x_1y_1+x_1y_2-2x_2y_1-x_2y_2-x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+2y_1y_2$$

$$\begin{aligned}&1-x_1+2x_2+2x_3+4x_4-y_1-y_2-x_1x_2-x_1x_3-2x_1x_4+x_2x_4+x_3x_4+x_1y_1\\&\qquad+3x_1y_2-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-2x_4y_1-4x_4y_2+3y_1y_2\end{aligned}$$

$$2-2x_1-x_2-x_3-2y_1+4y_2+x_1x_2+x_1x_3+2x_1y_1-2x_1y_2+x_2y_1-x_2y_2+x_3y_1-x_3y_2-x_4y_2-y_1y_2$$

$$2y_1+6y_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+2x_3x_4-x_1y_1-2x_1y_2-x_2y_1-2x_2y_2-2x_3y_1-4x_3y_2-2x_4y_1-4x_4y_2+3y_1y_2$$

$$3y_1+3y_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4-2x_1y_1-x_1y_2-x_2y_1-2x_2y_2-2x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+2y_1y_2$$

$$2x_4+2y_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-2x_3x_4+x_1y_1-x_1y_2+x_2y_1-x_2y_2+2x_3y_1-2x_3y_2-2x_4y_1+x_4y_2-y_1y_2$$

$$2x_4+y_1+2y_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-2x_3x_4+2x_1y_1-x_1y_2+2x_2y_1-x_2y_2+4x_3y_1-2x_3y_2-3x_4y_1+x_4y_2-3y_1y_2$$

$$x_3+x_4+2y_2+x_1x_2-x_1x_3-x_2x_3+x_2x_4-x_3x_4+2x_1y_1-x_1y_2+2x_2y_1-2x_2y_2-x_3y_1+x_3y_2+x_4y_1-2x_4y_2-2y_1y_2$$

$$x_3+2x_4+2y_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-3x_3x_4+x_1y_1-x_1y_2+x_2y_1-x_2y_2+3x_3y_1-3x_3y_2-2x_4y_1+2x_4y_2-2y_1y_2$$

$$\begin{aligned}&x_3+3x_4+y_1+3y_2+x_1x_2-x_1x_3-2x_1x_4-x_2x_3-2x_2x_4+x_3x_4+3x_1y_1\\&\qquad-2x_1y_2+3x_2y_1-2x_2y_2-2x_3y_1+x_3y_2-4x_4y_1+2x_4y_2-4y_1y_2\end{aligned}$$

$$2x_3+2x_4-x_1x_3-x_1x_4-x_2x_3-x_2x_4+x_3x_4+x_1y_1+x_1y_2+x_2y_1+x_2y_2-2x_3y_1-x_3y_2-2x_4y_1-x_4y_2+y_1y_2$$

$$2x_3+2x_4-x_1x_3-x_1x_4-x_2x_3-x_2x_4+x_3x_4+x_1y_1+x_1y_2+x_2y_1+x_2y_2-2x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+3y_1y_2$$

$$\begin{aligned}&x_2+x_3+x_4+2y_2-x_1x_2-x_1x_3-x_1x_4+x_2x_3+x_2x_4+x_3x_4+x_1y_1\\&\qquad+x_1y_2-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+y_1y_2\end{aligned}$$

$$\begin{aligned}&x_2+x_3+x_4+3y_2-x_1x_2-x_1x_3-x_1x_4+x_2x_3+x_2x_4+x_3x_4+x_1y_1\\&\qquad+2x_1y_2-x_2y_1-3x_2y_2-x_3y_1-3x_3y_2-x_4y_1-3x_4y_2+3y_1y_2\end{aligned}$$

$$x_2 + 2x_3 + 2x_4 + y_2 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_1$$
$$+ x_1y_2 - x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 2y_1y_2$$

$$2x_2 + 2x_3 + 2x_4 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_1 + x_1y_2$$
$$- 2x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$2x_2 + 2x_3 + 4x_4 + y_2 - x_1x_2 - x_1x_3 - 2x_1x_4 + x_2x_3 + 2x_2x_4 + 2x_3x_4 + x_1y_1$$
$$+ 2x_1y_2 - 2x_2y_1 - 3x_2y_2 - 2x_3y_1 - 3x_3y_2 - 3x_4y_1 - 5x_4y_2 + 4y_1y_2$$

$$1 - x_1 - x_2 + 2x_3 + 2x_4 - y_1 - y_2 + x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 + x_1y_1$$
$$+ 3x_1y_2 + x_2y_1 + 3x_2y_2 - x_3y_1 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$1 - x_1 + 3x_2 + 3x_3 + 3x_4 - y_1 - y_2 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$
$$+ x_1y_1 + 2x_1y_2 - 2x_2y_1 - 3x_2y_2 - 2x_3y_1 - 3x_3y_2 - 2x_4y_1 - 3x_4y_2 + 4y_1y_2$$

$$3 - 2x_1 - 2x_2 - 2x_3 - 3y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_2x_3 + 2x_1y_1 - x_1y_2 + 2x_2y_1 - x_2y_2 + 2x_3y_1 - x_3y_2 - x_4y_2 - y_1y_2$$

$$3y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - 2x_1y_1 - 2x_1y_2$$
$$- 2x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$3y_1 + 6y_2 + x_1x_2 + x_1x_3 + 2x_1x_4 + x_2x_3 + 2x_2x_4 + 2x_3x_4 - 2x_1y_1 - 3x_1y_2$$
$$- 2x_2y_1 - 3x_2y_2 - 2x_3y_1 - 3x_3y_2 - 3x_4y_1 - 5x_4y_2 + 4y_1y_2$$

$$x_4 + 3y_2 + x_1x_2 + x_1x_3 - x_1x_4 + x_2x_3 - x_2x_4 - x_3x_4 + 2x_1y_1 - 2x_1y_2$$
$$+ 2x_2y_1 - 2x_2y_2 + 2x_3y_1 - 2x_3y_2 - x_4y_1 + x_4y_2 - 3y_1y_2$$

$$x_4+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+x_1y_1-2x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2-x_4y_1+x_4y_2-2y_1y_2$$

$$x_4+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+x_1y_1-2x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2-x_4y_1+x_4y_2-y_1y_2$$

$$x_4+6y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+x_1y_1-3x_1y_2+x_2y_1-3x_2y_2+x_3y_1-3x_3y_2-x_4y_1+2x_4y_2-3y_1y_2$$

$$x_4 + y_1 + 3y_2 + x_1x_2 + x_1x_3 - x_1x_4 + x_2x_3 - x_2x_4 - x_3x_4 + 3x_1y_1 - 2x_1y_2$$
$$+ 3x_2y_1 - 2x_2y_2 + 3x_3y_1 - 2x_3y_2 - 2x_4y_1 + x_4y_2 - 4y_1y_2$$

$$x_3 + x_4 + 5y_2 + x_1x_2 - x_1x_3 + 2x_1x_4 - x_2x_3 + 2x_2x_4 - 2x_3x_4 + 2x_1y_1 - 3x_1y_2$$
$$+ 2x_2y_1 - 3x_2y_2 - x_3y_1 + 2x_3y_2 + 3x_4y_1 - 5x_4y_2 - 4y_1y_2$$

$$1 - x_1 - x_2 + 3x_3 + 3x_4 - y_1 - y_2 + x_1x_2 - 2x_1x_3 - 2x_1x_4 - x_2x_3 - x_2x_4 + x_3x_4$$
$$+ 2x_1y_1 + 4x_1y_2 + x_2y_1 + 2x_2y_2 - 2x_3y_1 - 3x_3y_2 - 2x_4y_1 - 3x_4y_2 + 4y_1y_2$$

$$3-3x_1-x_2-x_3-x_4-3y_1+5y_2+x_1x_2+x_1x_3+x_1x_4+3x_1y_1-3x_1y_2+x_2y_1-x_2y_2+x_3y_1-x_3y_2+x_4y_1-x_4y_2-2y_1y_2$$

$$2 - 2x_1 - 2x_2 - x_3 - x_4 - 2y_1 + 9y_2 + 2x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + 2x_1y_1$$
$$- 4x_1y_2 + 2x_2y_1 - 4x_2y_2 + x_3y_1 - 2x_3y_2 + x_4y_1 - 2x_4y_2 - 3y_1y_2$$

$$3 - 2x_1 - 2x_2 - 2x_3 - x_4 - 3y_1 + 5y_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_2x_4 + x_3x_4 + 2x_1y_1$$
$$- x_1y_2 + 2x_2y_1 - 2x_2y_2 + 2x_3y_1 - 2x_3y_2 + x_4y_1 - 2x_4y_2 - 2y_1y_2$$

$$3 - 3x_1 - 2x_2 - 2x_3 - 2x_4 - 3y_1 + 10y_2 + 2x_1x_2 + 2x_1x_3 + 2x_1x_4 + x_2x_3 + x_2x_4$$
$$+ x_3x_4 + 3x_1y_1 - 5x_1y_2 + 2x_2y_1 - 3x_2y_2 + 2x_3y_1 - 3x_3y_2 + 2x_4y_1 - 3x_4y_2 - 4y_1y_2$$

$$3 - 2x_1 - 2x_2 - 2x_3 - 2x_4 - 3y_1 + 6y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$
$$+ 2x_1y_1 - 2x_1y_2 + 2x_2y_1 - 2x_2y_2 + 2x_3y_1 - 2x_3y_2 + 2x_4y_1 - 2x_4y_2 - 3y_1y_2$$

- Quadratizations of $-x_1x_2x_3x_4x_5$ with one auxiliary variable y_1:

$$4y_1 - x_1y_1 - x_2y_1 - x_3y_1 - x_4y_1 - x_5y_1$$

$$x_2 + x_3 + x_4 + x_5 - x_1x_2 - x_1x_3 - x_1x_4 - x_1x_5 + 3x_1y_1 - x_2y_1 - x_3y_1 - x_4y_1 - x_5y_1$$

- Quadratizations of $x_1x_2x_3$ with one auxiliary variable y_1:

$$y_1 + x_2x_3 + x_1y_1 - x_2y_1 - x_3y_1$$

$$y_1 + x_1x_2 + x_1x_3 + x_2x_3 - x_1y_1 - x_2y_1 - x_3y_1$$

- Quadratizations of $x_1x_2x_3$ with two auxiliary variables y_1, y_2:

$$y_2 + x_2x_3 + x_1y_2 - x_2y_2 - x_3y_2$$

$$x_3 + y_2 + x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - y_1y_2$$

$$y_2 + x_1x_2 + x_1x_3 + x_2x_3 - x_1y_2 - x_2y_2 - x_3y_2$$

$$y_1 + y_2 + x_2x_3 - x_1y_1 + x_1y_2 - x_2y_2 + x_3y_1 - x_3y_2 - y_1y_2$$

$$y_1 + y_2 + x_1x_3 + x_2x_3 - x_1y_1 + x_2y_1 - x_2y_2 - x_3y_2 - y_1y_2$$

$$y_1 + y_2 + x_1x_3 + x_2x_3 - x_1y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_3 + x_1y_2 + x_2y_1 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_3 + y_2 + x_1x_2 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - y_1y_2$$

$$x_2 + x_3 + x_2x_3 + x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$y_1 + y_2 + x_2x_3 - x_1y_1 + x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - 2y_1y_2$$

$$y_1 + y_2 + x_2x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$y_1 + y_2 + x_1x_2 + x_1x_3 + x_2x_3 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 + y_1y_2$$

$$y_1 + y_2 + x_1x_2 + x_1x_3 + x_2x_3 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + 2y_1y_2$$

$$x_3 + y_1 + y_2 - x_1x_3 + 2x_2x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_2y_2 - 2x_3y_1 - 2x_3y_2 + y_1y_2$$

$$x_3 + y_1 + y_2 + x_1x_2 + 2x_1x_3 + 2x_2x_3 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - 2x_3y_1 - 2x_3y_2 + y_1y_2$$

$$2x_3 + y_2 + x_1x_2 - x_1x_3 - x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - 2x_3y_1 + 2x_3y_2 - y_1y_2$$

$$x_2 + x_3 - x_1x_2 - x_1x_3 + x_2x_3 + 2x_1y_1 + 2x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$1 - x_1 - x_2 - y_1 + 2y_2 + x_1x_2 + x_2x_3 + x_1y_1 + x_2y_1 - x_2y_2 - x_3y_2 - y_1y_2$$

$$1 - x_1 - x_2 + x_3 - y_1 + 2y_2 + x_1x_2 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 + x_3y_2 - y_1y_2$$

$$1 - x_1 - x_2 + x_3 - y_1 + 2y_2 + 2x_1x_2 - x_1x_3 + 2x_1y_1 - 2x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 + x_3y_2 - y_1y_2$$

$$1 - x_1 + x_3 - y_1 + x_2x_3 + x_1y_1 + x_1y_2 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$1 - x_1 + x_3 - y_1 + x_1x_2 + x_2x_3 + x_1y_1 - x_2y_2 - x_3y_1 - x_3y_2 + y_1y_2$$

$$1 - x_1 + x_2 + x_3 - y_1 - y_2 + x_2x_3 + x_1y_1 + x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 + 2y_1y_2$$

$$1 - x_1 - x_2 - y_1 + 2y_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_2 - y_1y_2$$

$$y_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - y_1y_2$$

$$1 - x_1 - x_2 - x_3 - y_1 + 2y_2 + x_1x_2 + 2x_1x_3 + 2x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + 2x_3y_1 - 2x_3y_2 - y_1y_2$$

$$1 - x_1 - x_2 - x_3 - y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - 2y_1y_2$$

- Quadratizations of $x_1x_2x_3x_4$ with one auxiliary variable y_1:

$$3y_1 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - 2x_1y_1 - 2x_2y_1 - 2x_3y_1 - 2x_4y_1$$

- Quadratizations of $x_1x_2x_3x_4$ with two auxiliary variables y_1, y_2:

$$y_1 + y_2 + x_3x_4 - x_1y_1 + x_1y_2 + x_2y_1 - x_3y_2 - x_4y_2 - y_1y_2$$

$$y_1 + y_2 + x_1x_4 + x_2x_3 - x_1y_1 - x_2y_2 - x_3y_2 - x_4y_1 + y_1y_2$$

$$y_1 + 2y_2 + x_3x_4 + x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2$$

$$y_1 + y_2 + x_3x_4 + x_1y_2 + x_2y_1 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$y_1 + y_2 + x_1x_4 + x_2x_3 - x_1y_1 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - x_4y_1 - y_1y_2$$

$$x_4 + y_1 + y_2 - x_1x_4 + x_3x_4 + x_1y_1 + x_1y_2 - x_2y_1 + x_2y_2 - x_3y_2 - x_4y_1 - 2x_4y_2$$

$$x_3 + x_4 + y_2 - x_1x_3 - x_1x_4 + x_3x_4 + x_1y_1 + 2x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2$$

$$x_3 + x_4 + y_2 + x_1x_2 + x_3x_4 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2$$

$$x_3 + x_4 + 3y_2 + x_1x_2 - x_1x_3 - x_2x_3 + x_1y_1 - 2x_1y_2 + x_2y_1 - 2x_2y_2 - x_3y_1 + 3x_3y_2 - x_4y_1 - 2y_1y_2$$

$$x_3 + x_4 + 3y_2 + x_1x_2 - x_1x_3 - x_2x_3 + x_1y_1 - 2x_1y_2 + x_2y_1 - 2x_2y_2 - x_3y_1 + 3x_3y_2 - x_4y_1 - x_4y_2 - y_1y_2$$

$$x_3 + x_4 + y_1 + 2y_2 + x_1x_2 - x_1x_3 - x_2x_3 - x_1y_1 + 2x_1y_2 - x_2y_1 + 2x_2y_2 + 2x_3y_1 - 3x_3y_2 - x_4y_1 - 3y_1y_2$$

$$1 - x_1 - x_2 - y_1 + 2y_2 + x_1x_2 + x_3x_4 + x_1y_1 + x_2y_1 - x_3y_2 - x_4y_2 - y_1y_2$$

$$1 - x_1 - x_2 + x_4 - y_1 + 2y_2 + x_1x_2 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_2 - x_4y_1 - y_1y_2$$

$$1 - x_1 + x_4 - y_1 + x_2x_3 + x_1y_1 + x_1y_2 - x_2y_2 - x_3y_2 - x_4y_1 + y_1y_2$$

$$2 - 2x_1 - 2x_2 + x_3 + 2x_4 - 2y_1 + 2y_2 + 2x_1x_2 - x_1x_4 - x_2x_4 + 3x_1y_1 - x_1y_2 + 3x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - 2x_4y_1 + x_4y_2 - y_1y_2$$

$$2 - 2x_1 + x_2 + 2x_3 + 2x_4 - 2y_1 - 2y_2 - x_1x_3 - x_1x_4 + x_3x_4 + 2x_1y_1 + 3x_1y_2 - x_2y_1 - x_3y_1 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$y_1 + y_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$y_1 + 2y_2 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - 2x_4y_2$$

$$y_1+3y_2+x_2x_3+x_2x_4+x_3x_4+x_1y_2-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+2y_1y_2$$

$$2y_1+3y_2+x_1x_4+x_2x_3+x_2x_4+x_3x_4-x_1y_1-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+3y_1y_2$$

$$x_4+y_1+y_2-x_1x_4+x_2x_4+x_3x_4+x_1y_1+x_1y_2-x_2y_1-x_3y_2-2x_4y_1-2x_4y_2+y_1y_2$$

$$x_4+y_1+y_2+x_1x_4+x_2x_3-x_2x_4-x_3x_4-x_1y_1+x_2y_1-x_2y_2+x_3y_1-x_3y_2-2x_4y_1+2x_4y_2-y_1y_2$$

$$x_4+y_1+y_2+x_1x_4+x_2x_3-x_2x_4-x_3x_4-x_1y_1+x_2y_1-x_2y_2+x_3y_1-x_3y_2-2x_4y_1+x_4y_2$$

$$x_4+y_1+3y_2+x_1x_4+x_2x_3-x_2x_4-x_3x_4-x_1y_1+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2-2x_4y_1+3x_4y_2-2y_1y_2$$

$$x_4+2y_1+2y_2+x_1x_3-x_1x_4+x_2x_3-x_3x_4-x_1y_1+2x_1y_2-x_2y_1-2x_3y_1+2x_3y_2+2x_4y_1-3x_4y_2-3y_1y_2$$

$$x_3+x_4+y_2-x_1x_3-x_1x_4+x_3x_4+x_1y_1+3x_1y_2+x_2y_1-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+2y_1y_2$$

$$x_3+x_4+2y_2-x_1x_3-x_1x_4+x_3x_4+x_1y_1+3x_1y_2+x_2y_1-x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+y_1y_2$$

$$x_3+x_4+y_1+2y_2-x_1x_3-x_1x_4+2x_3x_4+x_1y_1+2x_1y_2-x_2y_1+x_2y_2-x_3y_1-3x_3y_2-x_4y_1-3x_4y_2+y_1y_2$$

$$x_3+x_4+y_1+2y_2-x_1x_3-x_1x_4+x_2x_4+2x_3x_4+x_1y_1+2x_1y_2-x_2y_1-x_3y_1-3x_3y_2-2x_4y_1-3x_4y_2+2y_1y_2$$

$$x_3+3x_4+3y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_1y_1-2x_1y_2+2x_2y_1-2x_2y_2-x_3y_1+3x_3y_2-3x_4y_1+x_4y_2-3y_1y_2$$

$$x_2+x_3+x_4+y_2+x_2x_3+x_2x_4+x_3x_4+x_1y_1-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+2y_1y_2$$

$$\begin{aligned}x_2+x_3+x_4+2y_1+2y_2-x_1x_2-x_1x_3-x_1x_4+2x_2x_4+2x_3x_4+2x_1y_1+2x_1y_2\\ -3x_2y_1-x_2y_2-x_3y_1-3x_3y_2-3x_4y_1-3x_4y_2+3y_1y_2\end{aligned}$$

$$1-x_1-x_2+x_4-y_1+2y_2+x_1x_2+x_2x_3-x_2x_4+x_1y_1+2x_2y_1-2x_2y_2-x_3y_2-x_4y_1+x_4y_2-y_1y_2$$

$$1-x_1-x_2+x_3+x_4-y_1+x_1x_2+x_3x_4+x_1y_1+x_2y_1-x_3y_1-x_3y_2-x_4y_1-x_4y_2+y_1y_2$$

$$2-2x_1-2x_2+2x_4-2y_1+3y_2+2x_1x_2-x_1x_4+x_2x_3-x_2x_4+3x_1y_1-x_1y_2+3x_2y_1-2x_2y_2-x_3y_2-2x_4y_1+x_4y_2-2y_1y_2$$

$$2-2x_1+2x_3+2x_4-2y_1-y_2-x_1x_3-x_1x_4+x_2x_4+x_3x_4+3x_1y_1+2x_1y_2-x_2y_2-2x_3y_1-x_3y_2-2x_4y_1-2x_4y_2+3y_1y_2$$

$$\begin{aligned}2-2x_1+2x_2+2x_3+2x_4-2y_1-2y_2-x_1x_2-x_1x_3-x_1x_4+x_2x_4+x_3x_4+3x_1y_1\\ +3x_1y_2-2x_2y_1-x_2y_2-x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+4y_1y_2\end{aligned}$$

$$3y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4-2x_1y_2-2x_2y_2-2x_3y_2-2x_4y_2$$

$$y_1+3y_2+x_1x_4+x_2x_3+x_2x_4+x_3x_4-x_1y_1+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2-2x_4y_2-2y_1y_2$$

$$y_1+3y_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4-x_1y_1-x_1y_2+x_2y_1-2x_2y_2-2x_3y_2-2x_4y_2-y_1y_2$$

$$3y_1+3y_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4-2x_1y_1-x_1y_2-x_2y_1-2x_2y_2-2x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+4y_1y_2$$

$$x_4+3y_2+x_1x_2+x_1x_3+x_2x_3+x_1y_1-2x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2-x_4y_1-2y_1y_2$$

$$x_4+y_1+3y_2+x_1x_4+x_2x_3-x_2x_4-x_3x_4-x_1y_1+x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2-2x_4y_1+3x_4y_2-3y_1y_2$$

$$x_3+x_4+2y_2+x_1x_2+x_2x_3+x_2x_4+x_3x_4+x_1y_1-x_1y_2-2x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+y_1y_2$$

$$\begin{aligned}x_3+x_4+y_1+2y_2-x_1x_3-x_1x_4+x_2x_3+x_2x_4+2x_3x_4+x_1y_1+2x_1y_2\\ -x_2y_1-x_2y_2-2x_3y_1-3x_3y_2-2x_4y_1-3x_4y_2+3y_1y_2\end{aligned}$$

$$x_3+x_4+y_1+2y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4-x_1y_1 \\ +2x_1y_2-x_2y_1+2x_2y_2+2x_3y_1-3x_3y_2+2x_4y_1-3x_4y_2-3y_1y_2$$

$$x_3+x_4+y_1+2y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4-x_1y_1 \\ +2x_1y_2-x_2y_1+2x_2y_2+x_3y_1-3x_3y_2+2x_4y_1-3x_4y_2-2y_1y_2$$

$$x_3+x_4+y_1+2y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4-x_1y_1 \\ +2x_1y_2-x_2y_1+2x_2y_2+x_3y_1-3x_3y_2+x_4y_1-3x_4y_2-y_1y_2$$

$$x_3+x_4+y_1+4y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4-x_1y_1 \\ +3x_1y_2-x_2y_1+3x_2y_2+x_3y_1-4x_3y_2+x_4y_1-4x_4y_2-3y_1y_2$$

$$x_3+x_4+2y_1+2y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4-2x_1y_1 \\ +2x_1y_2-x_2y_1+2x_2y_2+3x_3y_1-3x_3y_2+3x_4y_1-3x_4y_2-4y_1y_2$$

$$x_3+x_4+2y_1+2y_2+x_1x_2-x_1x_3-x_2x_3+x_2x_4-x_3x_4-x_1y_1+2x_1y_2 \\ -2x_2y_1+2x_2y_2+3x_3y_1-3x_3y_2-2x_4y_1+x_4y_2-4y_1y_2$$

$$x_3+x_4+2y_1+3y_2-x_1x_3-x_1x_4+2x_2x_3+2x_2x_4+2x_3x_4+2x_1y_1+x_1y_2 \\ -2x_2y_1-3x_2y_2-3x_3y_1-3x_3y_2-3x_4y_1-3x_4y_2+4y_1y_2$$

$$x_3+x_4+2y_1+3y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4+2x_1y_1 \\ -2x_1y_2+2x_2y_1-2x_2y_2-3x_3y_1+3x_3y_2-3x_4y_1+3x_4y_2-5y_1y_2$$

$$x_3+x_4+2y_1+3y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4+2x_1y_1 \\ -2x_1y_2+2x_2y_1-2x_2y_2-3x_3y_1+x_3y_2-3x_4y_1+3x_4y_2-3y_1y_2$$

$$x_3+x_4+2y_1+3y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3+2x_3x_4+2x_1y_1-2x_1y_2 \\ +x_2y_1-2x_2y_2-3x_3y_1+3x_3y_2-3x_4y_1+2x_4y_2-4y_1y_2$$

$$x_3+x_4+2y_1+5y_2+x_1x_2-x_1x_3-x_1x_4-x_2x_3-x_2x_4+2x_3x_4+2x_1y_1 \\ -3x_1y_2+2x_2y_1-3x_2y_2-3x_3y_1+4x_3y_2-3x_4y_1+4x_4y_2-6y_1y_2$$

$$x_3+3x_4+3y_1+3y_2+x_1x_2-x_1x_3-2x_1x_4-x_2x_3-2x_2x_4+3x_3x_4-2x_1y_1 \\ +3x_1y_2-2x_2y_1+3x_2y_2+3x_3y_1-4x_3y_2+4x_4y_1-6x_4y_2-6y_1y_2$$

$$2x_3+2x_4-x_1x_3-x_1x_4-x_2x_3-x_2x_4+x_3x_4+x_1y_1+3x_1y_2+3x_2y_1+x_2y_2-2x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+3y_1y_2$$

$$x_2+x_3+x_4+y_2-x_1x_2-x_1x_3-x_1x_4+x_2x_3+x_2x_4+x_3x_4+x_1y_1+3x_1y_2 \\ -x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+2y_1y_2$$

$$x_2+x_3+x_4+3y_2-x_1x_2-x_1x_3-x_1x_4+x_2x_3+x_2x_4+x_3x_4+x_1y_1+4x_1y_2 \\ -x_2y_1-3x_2y_2-x_3y_1-3x_3y_2-x_4y_1-3x_4y_2+3y_1y_2$$

$$x_2+x_3+x_4+2y_1+2y_2-x_1x_2-x_1x_3-x_1x_4+2x_2x_3+2x_2x_4+2x_3x_4+2x_1y_1 \\ +2x_1y_2-3x_2y_1-3x_2y_2-3x_3y_1-3x_3y_2-3x_4y_1-2x_4y_2+4y_1y_2$$

$$x_2+x_3+x_4+2y_1+2y_2-x_1x_2-x_1x_3-x_1x_4+2x_2x_3+2x_2x_4+2x_3x_4+2x_1y_1 \\ +2x_1y_2-3x_2y_1-3x_2y_2-3x_3y_1-3x_3y_2-3x_4y_1-3x_4y_2+5y_1y_2$$

$$x_2 + x_3 + 2x_4 + 2y_1 + 4y_2 - x_1x_2 - x_1x_3 - 2x_1x_4 + 2x_2x_3 + 3x_2x_4 + 3x_3x_4 + 2x_1y_1 + 3x_1y_2 - 3x_2y_1 - 4x_2y_2 - 3x_3y_1 - 4x_3y_2 - 4x_4y_1 - 6x_4y_2 + 6y_1y_2$$

$$2x_2 + 2x_3 + 2x_4 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_1 + 3x_1y_2 - 2x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$1 - x_1 - x_2 - x_3 + x_4 - y_1 + 2y_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_1y_1 - x_1y_2 + x_2y_1 - x_2y_2 + x_3y_1 - x_3y_2 - x_4y_1 - y_1y_2$$

$$1 - x_1 - x_2 + x_3 + x_4 - y_1 + x_1x_2 - x_1x_3 - x_1x_4 + x_3x_4 + 2x_1y_1 + 2x_1y_2 + x_2y_1 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$1 - x_1 - x_2 + x_3 + x_4 - y_1 + y_2 + x_1x_2 - x_1x_3 - x_1x_4 + x_3x_4 + 2x_1y_1 + 3x_1y_2 + x_2y_1 - x_3y_1 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + 2y_1y_2$$

$$1 - x_1 + x_3 + x_4 - y_1 + y_2 + x_1x_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_1 - 2x_2y_2 - x_3y_1 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + 2y_1y_2$$

$$1 - x_1 + x_2 + x_3 + x_4 - y_1 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_1 + x_1y_2 - x_2y_1 - 2x_2y_2 - x_3y_1 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$1 - x_1 + x_2 + x_3 + 2x_4 - y_1 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + 2x_1y_1 + 3x_1y_2 - x_2y_1 - 2x_2y_2 - x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$2 - 2x_1 - 2x_2 + 2x_4 - 2y_1 + 5y_2 + 2x_1x_2 - x_1x_4 + 2x_2x_3 - x_2x_4 - x_3x_4 + 3x_1y_1 - x_1y_2 + 3x_2y_1 - 3x_2y_2 + x_3y_1 - 3x_3y_2 - 2x_4y_1 + 2x_4y_2 - 3y_1y_2$$

$$2 - 2x_1 - x_2 + 2x_3 + 2x_4 - 2y_1 - y_2 + x_1x_2 - x_1x_3 - x_1x_4 + x_3x_4 + 3x_1y_1 + x_1y_2 + x_2y_1 - 2x_3y_1 - x_3y_2 - 2x_4y_1 - x_4y_2 + 2y_1y_2$$

$$2 - 2x_1 + x_2 + 2x_3 + 2x_4 - 2y_1 - y_2 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + 3x_1y_1 + 3x_1y_2 - x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 4y_1y_2$$

$$2 - 2x_1 + 2x_2 + 2x_3 + 2x_4 - 2y_1 - 2y_2 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + 3x_1y_1 + 3x_1y_2 - 2x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 5y_1y_2$$

$$2 - 2x_1 + 2x_2 + 2x_3 + 4x_4 - 2y_1 - y_2 - x_1x_2 - x_1x_3 - 2x_1x_4 + x_2x_3 + 2x_2x_4 + 2x_3x_4 + 3x_1y_1 + 4x_1y_2 - 2x_2y_1 - 3x_2y_2 - 2x_3y_1 - 3x_3y_2 - 3x_4y_1 - 5x_4y_2 + 6y_1y_2$$

$$3 - 3x_1 + 3x_2 + 3x_3 + 3x_4 - 3y_1 - 3y_2 - x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + 3x_1y_1 + 4x_1y_2 - 2x_2y_1 - 3x_2y_2 - 2x_3y_1 - 3x_3y_2 - 2x_4y_1 - 3x_4y_2 + 6y_1y_2$$

$$3 - 2x_1 - 2x_2 - 2x_3 + x_4 - 3y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_2x_3 + 2x_1y_1 - x_1y_2 + 2x_2y_1 - x_2y_2 + 2x_3y_1 - x_3y_2 - x_4y_1 - 2y_1y_2$$

$$y_1 + y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - x_1y_1 - x_1y_2 - x_2y_1 - x_2y_2 - x_3y_1 - x_3y_2 - x_4y_1 - x_4y_2 + y_1y_2$$

$$y_1 + 3y_2 + x_2x_3 + x_2x_4 + x_3x_4 - x_1y_1 + x_1y_2 + x_2y_1 - 2x_2y_2 + x_3y_1 - 2x_3y_2 + x_4y_1 - 2x_4y_2 - 3y_1y_2$$

$$y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - x_1y_1 - 2x_1y_2 - x_2y_1 - 2x_2y_2 - 2x_3y_2 - 2x_4y_2 + y_1y_2$$

$$y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - x_1y_1 - 2x_1y_2 - x_2y_1 - 2x_2y_2 - x_3y_1 - 2x_3y_2 - 2x_4y_2 + 2y_1y_2$$

$$y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - x_1y_1 - 2x_1y_2 - x_2y_1 - 2x_2y_2 - x_3y_1 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$2y_1 + 3y_2 + x_1x_2 + x_1x_3 + 2x_1x_4 + x_2x_3 + 2x_2x_4 + 2x_3x_4 - x_1y_1 - 2x_1y_2 - x_2y_1 - 2x_2y_2 - x_3y_1 - 2x_3y_2 - 2x_4y_1 - 3x_4y_2 + y_1y_2$$

$$3y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - 2x_1y_1 - 2x_1y_2 - 2x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 + 3y_1y_2$$

$$3y_1 + 3y_2 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 - 2x_1y_1 - 2x_1y_2 - 2x_2y_1 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 5y_1y_2$$

$$3y_1+6y_2+x_1x_2+x_1x_3+2x_1x_4+x_2x_3+2x_2x_4+2x_3x_4-2x_1y_1 \\ -3x_1y_2-2x_2y_1-3x_2y_2-2x_3y_1-3x_3y_2-3x_4y_1-5x_4y_2+6y_1y_2$$

$$x_4+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+2x_1y_1-2x_1y_2 \\ +2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2-x_4y_1+3x_4y_2-3y_1y_2$$

$$x_4+6y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+x_1y_1-3x_1y_2+x_2y_1-3x_2y_2+x_3y_1-3x_3y_2-x_4y_1+4x_4y_2-3y_1y_2$$

$$x_4+y_1+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+x_1y_1 \\ -2x_1y_2+x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2-2x_4y_1+3x_4y_2-3y_1y_2$$

$$x_4+y_1+3y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+2x_2x_4+2x_3x_4-2x_1y_2-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-2x_4y_1-3x_4y_2+y_1y_2$$

$$x_4+y_1+3y_2+x_1x_2+x_1x_3+2x_1x_4+x_2x_3+2x_2x_4+2x_3x_4-x_1y_1 \\ -2x_1y_2-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-2x_4y_1-4x_4y_2+2y_1y_2$$

$$x_4+2y_1+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+2x_1y_1 \\ -2x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2-3x_4y_1+3x_4y_2-5y_1y_2$$

$$x_4+2y_1+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4+x_1y_1 \\ -2x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2-3x_4y_1+3x_4y_2-4y_1y_2$$

$$x_4+2y_1+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4+x_3x_4+2x_1y_1-2x_1y_2+2x_2y_1-2x_2y_2-2x_3y_2-3x_4y_1+x_4y_2-3y_1y_2$$

$$x_4+3y_1+3y_2+x_1x_2+x_1x_3-x_1x_4+x_2x_3-x_2x_4-x_3x_4-2x_1y_1 \\ +3x_1y_2-2x_2y_1+3x_2y_2-2x_3y_1+3x_3y_2+3x_4y_1-4x_4y_2-6y_1y_2$$

$$2x_4+3y_1+3y_2+x_1x_2+x_1x_3+3x_1x_4+x_2x_3+3x_2x_4+3x_3x_4-2x_1y_1 \\ -2x_1y_2-2x_2y_1-2x_2y_2-2x_3y_1-2x_3y_2-5x_4y_1-5x_4y_2+3y_1y_2$$

$$x_3+x_4+y_1+5y_2+x_1x_2+2x_1x_3+2x_1x_4+2x_2x_3+2x_2x_4+3x_3x_4 \\ -x_1y_1-3x_1y_2-x_2y_1-3x_2y_2-2x_3y_1-5x_3y_2-2x_4y_1-5x_4y_2+3y_1y_2$$

$$x_3+x_4+2y_1+5y_2+x_1x_2-x_1x_3+2x_1x_4-x_2x_3+2x_2x_4-2x_3x_4 \\ +2x_1y_1-3x_1y_2+2x_2y_1-3x_2y_2-3x_3y_1+4x_3y_2+3x_4y_1-5x_4y_2-6y_1y_2$$

$$x_2+x_3+x_4+2y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4-2x_1y_2-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+y_1y_2$$

$$x_1+x_2+x_3+x_4+y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4 \\ -x_1y_1-2x_1y_2-x_2y_1-2x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+2y_1y_2$$

$$x_1+x_2+x_3+2x_4+3y_2+x_1x_2+x_1x_3+2x_1x_4+x_2x_3+2x_2x_4+2x_3x_4 \\ -x_1y_1-3x_1y_2-x_2y_1-3x_2y_2-x_3y_1-3x_3y_2-2x_4y_1-5x_4y_2+3y_1y_2$$

$$2x_1+2x_2+2x_3+2x_4+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4-2x_1y_1 \\ -2x_1y_2-2x_2y_1-2x_2y_2-2x_3y_1-2x_3y_2-2x_4y_1-2x_4y_2+3y_1y_2$$

$$1-x_1-x_2-x_3+x_4-y_1+5y_2+x_1x_2+x_1x_3+2x_2x_3-x_2x_4-x_3x_4+x_1y_1 \\ -x_1y_2+2x_2y_1-3x_2y_2+2x_3y_1-3x_3y_2-x_4y_1+2x_4y_2-3y_1y_2$$

$$1-x_1-x_2+x_3+x_4-y_1+x_1x_2-x_1x_3-x_1x_4+x_3x_4+2x_1y_1+3x_1y_2+x_2y_1+x_2y_2-x_3y_1-2x_3y_2-x_4y_1-2x_4y_2+3y_1y_2$$

$$1 - x_1 - x_2 + 2x_3 + 2x_4 - y_1 + y_2 + x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 + 2x_3x_4 \\ + x_1y_1 + 2x_1y_2 + x_2y_1 + 2x_2y_2 - x_3y_1 - 3x_3y_2 - x_4y_1 - 3x_4y_2 + y_1y_2$$

$$1 - x_1 - x_2 + 2x_3 + 2x_4 - y_1 + y_2 + x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 + 2x_3x_4 \\ + x_1y_1 + 3x_1y_2 + x_2y_1 + 3x_2y_2 - x_3y_1 - 4x_3y_2 - x_4y_1 - 4x_4y_2 + 3y_1y_2$$

$$1 - x_1 - x_2 + 2x_3 + 3x_4 - y_1 + x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 + 2x_3x_4 + x_1y_1 \\ + 2x_1y_2 + x_2y_1 + 2x_2y_2 - x_3y_1 - 3x_3y_2 - 2x_4y_1 - 3x_4y_2 + 2y_1y_2$$

$$1 - x_1 - x_2 + 3x_3 + 3x_4 - y_1 - y_2 + x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 + 2x_3x_4 \\ + x_1y_1 + 2x_1y_2 + x_2y_1 + 2x_2y_2 - 2x_3y_1 - 3x_3y_2 - 2x_4y_1 - 3x_4y_2 + 3y_1y_2$$

$$1 - x_1 + x_4 - y_1 + 2y_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_2x_4 + x_3x_4 + x_1y_1 - x_1y_2 - 2x_2y_2 - 2x_3y_2 - x_4y_1 - 2x_4y_2 + y_1y_2$$

$$2 - 2x_1 - 2x_2 - 2x_3 + 2x_4 - 2y_1 + 6y_2 + 2x_1x_2 + 2x_1x_3 - x_1x_4 + 2x_2x_3 - x_2x_4 \\ - x_3x_4 + 3x_1y_1 - 3x_1y_2 + 3x_2y_1 - 3x_2y_2 + 3x_3y_1 - 2x_3y_2 - 2x_4y_1 + 2x_4y_2 - 4y_1y_2$$

$$2 - 2x_1 - 2x_2 - 2x_3 + 2x_4 - 2y_1 + 7y_2 + 2x_1x_2 + 2x_1x_3 - x_1x_4 + 2x_2x_3 - x_2x_4 \\ - x_3x_4 + 3x_1y_1 - 3x_1y_2 + 3x_2y_1 - 3x_2y_2 + 3x_3y_1 - 3x_3y_2 - 2x_4y_1 + 2x_4y_2 - 5y_1y_2$$

$$2 - 2x_1 - 2x_2 - 2x_3 + 2x_4 - 2y_1 + 7y_2 + 2x_1x_2 + 2x_1x_3 - x_1x_4 + 2x_2x_3 - x_2x_4 \\ + 3x_1y_1 - 3x_1y_2 + 3x_2y_1 - 3x_2y_2 + 2x_3y_1 - 3x_3y_2 - 2x_4y_1 + x_4y_2 - 4y_1y_2$$

$$2 - 2x_1 - 2x_2 - 2x_3 + 2x_4 - 2y_1 + 10y_2 + 2x_1x_2 + 3x_1x_3 - x_1x_4 + 3x_2x_3 - x_2x_4 \\ - 2x_3x_4 + 3x_1y_1 - 4x_1y_2 + 3x_2y_1 - 4x_2y_2 + 4x_3y_1 - 6x_3y_2 - 2x_4y_1 + 3x_4y_2 - 6y_1y_2$$

$$2 - 2x_1 - 2x_2 + x_3 + 2x_4 - 2y_1 - y_2 + 2x_1x_2 - x_1x_3 - x_1x_4 - x_2x_4 + x_3x_4 \\ + 3x_1y_1 + 3x_1y_2 + 3x_2y_1 + 2x_2y_2 - x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 4y_1y_2$$

$$2 - 2x_1 - 2x_2 + 2x_3 + 2x_4 - 2y_1 - 2y_2 + 2x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 \\ + x_3x_4 + 3x_1y_1 + 3x_1y_2 + 3x_2y_1 + 3x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - x_4y_2 + 4y_1y_2$$

$$2 - 2x_1 - 2x_2 + 2x_3 + 2x_4 - 2y_1 - 2y_2 + 2x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 \\ + x_3x_4 + 3x_1y_1 + 3x_1y_2 + 3x_2y_1 + 3x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 5y_1y_2$$

$$2 - 2x_1 - 2x_2 + 2x_3 + 2x_4 - 2y_1 - y_2 + 2x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 \\ + x_3x_4 + 3x_1y_1 + 4x_1y_2 + 3x_2y_1 + 4x_2y_2 - 2x_3y_1 - 3x_3y_2 - 2x_4y_1 - 3x_4y_2 + 6y_1y_2$$

$$2 - 2x_1 - 2x_2 + 2x_3 + 2x_4 - 2y_1 + 2x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 + x_3x_4 \\ + 3x_1y_1 + x_1y_2 + 3x_2y_1 + 3x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$2 - 2x_1 + 2x_3 + 2x_4 - 2y_1 + x_1x_2 - x_1x_3 - x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + 3x_1y_1 \\ + x_1y_2 - 2x_2y_2 - 2x_3y_1 - 2x_3y_2 - 2x_4y_1 - 2x_4y_2 + 3y_1y_2$$

$$3 - 3x_1 - 3x_2 + 3x_3 + 3x_4 - 3y_1 - 3y_2 + 3x_1x_2 - 2x_1x_3 - 2x_1x_4 - x_2x_3 - x_2x_4 \\ + x_3x_4 + 4x_1y_1 + 6x_1y_2 + 3x_2y_1 + 4x_2y_2 - 2x_3y_1 - 3x_3y_2 - 2x_4y_1 - 3x_4y_2 + 6y_1y_2$$

$$3-2x_1-2x_2-2x_3-3y_1+5y_2+x_1x_2+x_1x_3+x_2x_3+x_3x_4+2x_1y_1-x_1y_2+2x_2y_1-x_2y_2+2x_3y_1-2x_3y_2-x_4y_2-3y_1y_2$$

$$1-x_1-x_2-y_1+4y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4+x_1y_1-2x_1y_2+x_2y_1-2x_2y_2-2x_3y_2-2x_4y_2-y_1y_2$$

$$3y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4-2x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2+x_4y_1-2x_4y_2-y_1y_2$$

$$\begin{aligned}&1-x_1-x_2-x_3-y_1+4y_2+x_1x_2+2x_1x_3+x_1x_4+2x_2x_3+x_2x_4+x_3x_4\\&\qquad+x_1y_1-2x_1y_2+x_2y_1-2x_2y_2+2x_3y_1-3x_3y_2-2x_4y_2-y_1y_2\end{aligned}$$

$$\begin{aligned}&1-x_1-x_2-x_3-y_1+5y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4+x_1y_1\\&\qquad-2x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2-2x_4y_2-2y_1y_2\end{aligned}$$

$$\begin{aligned}&3-2x_1-2x_2-2x_3-x_4-3y_1+7y_2+x_1x_2+x_1x_3+x_2x_3+x_2x_4+x_3x_4+2x_1y_1\\&\qquad-x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2+x_4y_1-2x_4y_2-4y_1y_2\end{aligned}$$

$$\begin{aligned}&3-2x_1-2x_2-2x_3-3y_1+6y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4+2x_1y_1\\&\qquad-2x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2-2x_4y_2-3y_1y_2\end{aligned}$$

$$3y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4+2x_1y_1-2x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2+2x_4y_1-2x_4y_2-3y_1y_2$$

$$3y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4+x_1y_1-2x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2+x_4y_1-2x_4y_2-2y_1y_2$$

$$6y_2+x_1x_2+x_1x_3+2x_1x_4+x_2x_3+2x_2x_4+2x_3x_4+x_1y_1-3x_1y_2+x_2y_1-3x_2y_2+x_3y_1-3x_3y_2+2x_4y_1-5x_4y_2-3y_1y_2$$

$$\begin{aligned}&1-x_1-x_2-x_3-x_4-y_1+2y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4\\&\qquad+x_1y_1-x_1y_2+x_2y_1-x_2y_2+x_3y_1-x_3y_2+x_4y_1-x_4y_2-y_1y_2\end{aligned}$$

$$\begin{aligned}&1-x_1-x_2-x_3-x_4-y_1+5y_2+x_1x_2+x_1x_3+2x_1x_4+x_2x_3+2x_2x_4+2x_3x_4\\&\qquad+x_1y_1-2x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2+2x_4y_1-4x_4y_2-2y_1y_2\end{aligned}$$

$$\begin{aligned}&1-x_1-x_2-x_3-x_4-y_1+6y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4\\&\qquad+x_1y_1-2x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2+x_4y_1-2x_4y_2-3y_1y_2\end{aligned}$$

$$\begin{aligned}&1-x_1-x_2-x_3-x_4-y_1+8y_2+x_1x_2+2x_1x_3+2x_1x_4+2x_2x_3+2x_2x_4+3x_3x_4\\&\qquad+x_1y_1-3x_1y_2+x_2y_1-3x_2y_2+2x_3y_1-5x_3y_2+2x_4y_1-5x_4y_2-3y_1y_2\end{aligned}$$

$$\begin{aligned}&2-2x_1-x_2-x_3-x_4-2y_1+4y_2+2x_1x_2+2x_1x_3+2x_1x_4+x_2x_3+x_2x_4+x_3x_4\\&\qquad+2x_1y_1-3x_1y_2+x_2y_1-2x_2y_2+x_3y_1-2x_3y_2+x_4y_1-2x_4y_2-y_1y_2\end{aligned}$$

$$\begin{aligned}&3-3x_1-2x_2-2x_3-2x_4-3y_1+6y_2+3x_1x_2+3x_1x_3+3x_1x_4+x_2x_3+x_2x_4+x_3x_4\\&\qquad+5x_1y_1-5x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2+2x_4y_1-2x_4y_2-3y_1y_2\end{aligned}$$

$$\begin{aligned}&3-3x_1-2x_2-2x_3-2x_4-3y_1+12y_2+2x_1x_2+2x_1x_3+2x_1x_4+x_2x_3+x_2x_4+x_3x_4\\&\qquad+3x_1y_1-5x_1y_2+2x_2y_1-3x_2y_2+2x_3y_1-3x_3y_2+2x_4y_1-3x_4y_2-6y_1y_2\end{aligned}$$

$$\begin{aligned}&3-2x_1-2x_2-2x_3-2x_4-3y_1+8y_2+x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4\\&\qquad+2x_1y_1-2x_1y_2+2x_2y_1-2x_2y_2+2x_3y_1-2x_3y_2+2x_4y_1-2x_4y_2-5y_1y_2\end{aligned}$$

- Quadratizations of $x_1x_2x_3x_4x_5$ with one auxiliary variable y_1:

none

References

[1] Adams, W. P., Bowers Lassiter, J., and Sherali, H. D. Persistency in 0-1 polynomial programming. *Math. Oper. Res. 23*, 2 (1998), 359–389. (Cited on page 89.)

[2] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. *Network flows: Theory, algorithms, and applications.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1993. (Cited on page 90.)

[3] Anthony, M. *Discrete mathematics of neural networks: Selected topics.* SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. (Cited on page 1.)

[4] Anthony, M., Boros, E., Crama, Y., and Gruber, A. Quadratic reformulations of nonlinear binary optimization problems. In preparation, 2014. (Cited on pages 20, 21, and 78.)

[5] Anthony, M., Boros, E., Crama, Y., and Gruber, A. Quadratization of symmetric pseudo-Boolean functions. ArXiv: http://arxiv.org/abs/1404.6535, Submitted to Discrete Appl. Math., 2014. (Cited on pages 19 and 78.)

[6] Anthony, M., and Harvey, M. *Linear Algebra: Concepts and Methods.* Cambridge University Press, Cambridge, 2012. (Cited on page 23.)

[7] Arora, S., Babai, L., Stern, J., and Sweedyk, Z. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. System Sci. 54*, 2, part 2 (1997), 317–331. 34th Annual Symposium on Foundations of Computer Science (Palo Alto, CA, 1993). (Cited on page 32.)

[8] Arora, S., and Barak, B. *Computational complexity: A modern approach.* Cambridge University Press, Cambridge, 2009. (Cited on pages 1, 2, 23, and 36.)

[9] Arora, S., Bollobás, B., Lovász, L., and Tourlakis, I. Proving integrality gaps without knowing the linear program. *Theory of Computing 2*, 1 (2006), 19–51. (Cited on page 22.)

[10] Arora, S., and Lund, C. Hardness of approximations. *in Approximation Algorithms for NP-Hard Problems, D. Hochbaum, ed., PWS Publishing, Boston* (1996), 1–54. Chapter 10. (Cited on pages 33, 36, 67, and 68.)

[11] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. Proof verification and the hardness of approximation problems. *J. ACM 45*, 3 (1998), 501–555. (Cited on page 32.)

[12] Arora, S., and Safra, S. Probabilistic checking of proofs: a new characterization of NP. *J. ACM 45*, 1 (1998), 70–122. (Cited on pages 26, 32, and 75.)

[13] Ausiello, G., D'Atri, A., and Saccà, D. Minimal representation of directed hypergraphs. *SIAM J. Comput. 15*, 2 (1986), 418–431. (Cited on pages 5 and 6.)

[14] Avis, D., and Devroye, L. Estimating the number of vertices of a polyhedron. *Inf. Process. Lett. 73*, 3-4 (2000), 137–143. (Cited on page 152.)

[15] Balas, E. Disjunctive programming: properties of the convex hull of feasible points. *Discrete Appl. Math. 89*, 1-3 (1998), 3–44. (Cited on page 22.)

[16] Balas, E., Ceria, S., and Cornuéjols, G. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Programming 58*, 3, Ser. A (1993), 295–324. (Cited on page 22.)

[17] Bertsimas, D., Natarajan, K., and Teo, C.-P. Persistence in discrete optimization under data uncertainty. *Math. Program. 108*, 2-3, Ser. B (2006), 251–274. (Cited on page 89.)

[18] Bhattacharya, A., DasGupta, B., Mubayi, D., and Turán, G. On approximate Horn formula minimization. In *ICALP (1)* (2010), S. Abramsky, C. Gavoille, C. Kirchner, F. M. auf der Heide, and P. G. Spirakis, Eds., vol. 6198 of *Lecture Notes in Computer Science*, Springer, pp. 438–450. (Cited on pages 6, 7, 8, 9, and 10.)

[19] Billionnet, A., and Elloumi, S. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Math. Program. 109*, 1, Ser. A (2007), 55–68. (Cited on page 13.)

[20] Billionnet, A., and Minoux, M. Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions. *Discrete Applied Mathematics 12*, 1 (1985), 1 – 11. (Cited on pages 86 and 99.)

[21] Blake, A., Kohli, P., and Rother, C. *Markov Random Fields for Vision and Image Processing.* MIT Press, Cambridge, MA, 2011. editors. (Cited on page 90.)

[22] Boros, E., Čepek, O., Kogan, A., and Kučera, P. Exclusive and essential sets of implicates of Boolean functions. *Discrete Appl. Math. 158*, 2 (2010), 81–96. (Cited on pages 8, 10, 30, and 31.)

[23] Boros, E., Čepek, O., and Kučera, P. A decomposition method for CNF minimality proofs. *Theoret. Comput. Sci. 510* (2013), 111–126. (Cited on pages 5, 31, and 32.)

[24] Boros, E., Crama, Y., and Hammer, P. L. Upper-bounds for quadratic 0-1 maximization. *Oper. Res. Lett. 9*, 2 (1990), 73–79. (Cited on page 91.)

[25] Boros, E., and Gruber, A. Hardness results for approximate pure Horn CNF formulae minimization. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM 2012), Fort Lauderdale, Florida, USA, January 9-11* (2012), M. Golumbic and R. H. Sloan, Eds. (Cited on pages 9 and 27.)

[26] Boros, E., and Gruber, A. On quadratization of pseudo-boolean functions. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM 2012), Fort Lauderdale, Florida, USA, January 9-11* (2012), M. Golumbic and R. H. Sloan, Eds. (Cited on pages 15, 17, 18, 19, and 78.)

[27] Boros, E., and Gruber, A. Hardness results for approximate pure Horn CNF formulae minimization. *Annals of Mathematics and Artificial Intelligence* (2014), 1–37. (Cited on pages 9 and 27.)

[28] Boros, E., and Hammer, P. L. A max-flow approach to improved roof-duality in quadratic $0-1$ minimization. RUTCOR Research Report RRR 15-89, Rutgers University, New Brunswick, NJ, USA, 1989. (Cited on pages 3 and 90.)

[29] Boros, E., and Hammer, P. L. Pseudo-boolean optimization. *Discrete Applied Mathematics 123*, 1-3 (2002), 155–225. (Cited on pages 12, 23, 78, 79, 85, and 86.)

[30] Boros, E., Hammer, P. L., Sun, R., and Tavares, G. A max-flow approach to improved lower bounds for quadratic unconstrained binary optimization (QUBO). *Discrete Optimization 5*, 2 (2008), 501–529. (Cited on pages 3, 13, 14, and 90.)

[31] Boros, E., Hammer, P. L., and Tavares, G. Local search heuristics for quadratic unconstrained binary optimization (QUBO). *J. Heuristics 13*, 2 (2007), 99–132. (Cited on page 13.)

[32] Boykov, Y., and Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 9 (2004), 1124–1137. (Cited on page 90.)

[33] Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (2001), 1222–1239. (Cited on pages 14 and 15.)

[34] Braun, G., Fiorini, S., Pokutta, S., and Steurer, D. Approximation limits of linear programs (beyond hierarchies). In *FOCS* (2012), IEEE Computer Society, pp. 480–489. (Cited on page 22.)

[35] Buchheim, C., and Rinaldi, G. Efficient reduction of polynomial zero-one optimization to the quadratic case. *SIAM J. Optim. 18*, 4 (2007), 1398–1413. (Cited on page 21.)

[36] Buchheim, C., and Rinaldi, G. Terse integer linear programs for Boolean optimization. *J. Satisf. Boolean Model. Comput. 6*, 1-3 (2010), 121–139. (Cited on page 21.)

[37] Burer, S., and Letchford, A. N. Non-convex mixed-integer nonlinear programming: a survey. *Surv. Oper. Res. Manag. Sci. 17*, 2 (2012), 97–106. (Cited on page 13.)

[38] Cameron, P. J. *Permutation groups*, vol. 45 of *London Mathematical Society Student Texts.* Cambridge University Press, Cambridge, 1999. (Cited on page 81.)

[39] Čepek, O. *Structural Properties and Minimization of Horn Boolean Functions.* PhD thesis, Rutgers University, New Brunswick, NJ, October 1995. (Cited on page 5.)

[40] Čepek, O., Kucera, P., and Savický, P. Boolean functions with a simple certificate for cnf complexity. *Discrete Applied Mathematics 160*, 4-5 (2012), 365–382. (Cited on page 10.)

[41] Čepek, O., and Kučera, P. On the complexity of minimizing the number of literals in Horn formulae. RUTCOR Research Report RRR 11-208, Rutgers University, New Brunswick, NJ, March 2008. (Cited on page 5.)

[42] Charikar, M., Makarychev, K., and Makarychev, Y. Integrality gaps for sherali-adams relaxations. In *STOC* (2009), M. Mitzenmacher, Ed., ACM, pp. 283–292. (Cited on page 22.)

[43] Chlamtac, E., and Tulsiani, M. Convex relaxations and integrality gaps. In *Handbook on semidefinite, conic and polynomial optimization*, vol. 166 of *Internat. Ser. Oper. Res. Management Sci.* Springer, New York, 2012, pp. 139–169. (Cited on page 22.)

[44] Conforti, M., Cornuéjols, G., and Zambelli, G. Extended formulations in combinatorial optimization. *Ann. Oper. Res. 204* (2013), 97–143. (Cited on page 22.)

[45] Cook, S. A. The complexity of theorem-proving procedures. In *STOC* (1971), M. A. Harrison, R. B. Banerji, and J. D. Ullman, Eds., ACM, pp. 151–158. (Cited on page 4.)

[46] Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A. *Combinatorial optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York, 1998. A Wiley-Interscience Publication. (Cited on page 90.)

[47] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*, third ed. MIT Press, Cambridge, MA, 2009. (Cited on pages 23, 25, and 54.)

[48] Crama, Y. Recognition problems for special classes of polynomials in $0-1$ variables. *Mathematical Programming 44* (1989), 139–155. 10.1007/BF01587085. (Cited on pages 21 and 86.)

[49] Crama, Y., and Hammer, P. L. Pseudo-boolean optimization. In *Handbook of Applied Optimization*, P.M. Pardalos and M.G.C. Resende, eds. Oxford University Press, Oxford, 2002, pp. 445–450. (Cited on page 12.)

[50] Crama, Y., and Hammer, P. L. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, vol. 134 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 2010. (Cited on pages 1 and 78.)

[51] Crama, Y., and Hammer, P. L. *Boolean functions: Theory, algorithms, and applications*, vol. 142 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 2011. (Cited on pages 1, 4, 6, 12, 18, 23, 28, 29, 79, and 80.)

[52] Crama, Y., and Rodríguez-Heck, E. Personal communication. 2014. (Cited on page 156.)

[53] Crama, Y., and Rodríguez-Heck, E. Short prime quadratizations of cubic negative monomials. Research Report 05-2014, QuantOM, HEC Management School, University of Liege, Liege, Belgium, 2014. (Cited on page 157.)

[54] Dinur, I., and Harsha, P. Composition of low-error 2-query PCPs using decodable PCPs. *SIAM J. Comput. 42*, 6 (2013), 2452–2486. (Cited on pages 9, 67, 69, and 75.)

[55] Dinur, I., and Safra, S. On the hardness of approximating label-cover. *Inform. Process. Lett. 89*, 5 (2004), 247–254. (Cited on pages 9, 34, 35, 69, and 75.)

[56] Dowling, W. F., and Gallier, J. H. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Log. Program. 1*, 3 (1984), 267–284. (Cited on page 5.)

[57] Ellis, D., and Sudakov, B. Generating all subsets of a finite set with disjoint unions. *J. Combin. Theory Ser. A 118*, 8 (2011), 2319–2345. (Cited on page 130.)

[58] Feige, U., Goldwasser, S., Lovász, L., Safra, S., and Szegedy, M. Interactive proofs and the hardness of approximating cliques. *J. ACM 43*, 2 (1996), 268–292. (Cited on page 32.)

[59] Feige, U., Langberg, M., and Nissim, K. On the hardness of approximating NP witnesses. In *Approximation algorithms for combinatorial optimization (Saarbrücken, 2000)*, vol. 1913 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 2000, pp. 120–131. (Cited on page 91.)

[60] Feige, U., and Lovász, L. Two-prover one-round proof systems: Their power and their problems (extended abstract). In *STOC* (1992), ACM, pp. 733–744. (Cited on page 32.)

[61] Fiorini, S., Massar, S., Pokutta, S., Tiwary, H. R., and de Wolf, R. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In *STOC* (2012), H. J. Karloff and T. Pitassi, Eds., ACM, pp. 95–106. (Cited on page 22.)

[62] Fix, A. Reductions for rewriting QPBFs with spanning trees. Unpublished notes, 2011. (Cited on pages 17, 19, 113, and 116.)

[63] Fix, A., Gruber, A., Boros, E., and Zabih, R. A graph cut algorithm for higher-order Markov Random Fields. In Metaxas et al. [116], pp. 1020–1027. (Cited on pages 15, 19, 21, 78, 93, 100, 106, 108, and 109.)

[64] Fix, A., Gruber, A., Boros, E., and Zabih, R. A hypergraph-based reduction for higher-order Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (to appear). (Cited on pages 15, 19, 21, 78, 93, 106, 108, and 109.)

[65] Flum, J., and Grohe, M. *Parameterized complexity theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006. (Cited on page 130.)

[66] Fortet, R. L'Algèbre de Boole et ses applications en recherche opérationnelle. *Cahiers Centre Etudes Rech. Oper. no. 4* (1959), 5–36. (Cited on page 12.)

[67] Fortet, R. L'algèbre de Boole et ses applications en recherche opérationnelle. *Trabajos Estadíst. 11* (1960), 111–118. (Cited on page 12.)

[68] Frankl, P., and Rödl, V. Lower bounds for Turán's problem. *Graphs Combin. 1*, 3 (1985), 213–216. (Cited on page 140.)

[69] Freedman, D., and Drineas, P. Energy minimization via graph cuts: Settling what is possible. In *CVPR (2)* (2005), IEEE Computer Society, pp. 939–946. (Cited on pages 15, 16, 97, and 156.)

[70] Frein, Y., Lévêque, B., and Sebő, A. Generating all sets with bounded unions. *Combin. Probab. Comput. 17*, 5 (2008), 641–660. (Cited on page 130.)

[71] Füredi, Z., and Katona, G. O. H. 2-Bases of quadruples. *Combin. Probab. Comput. 15*, 1-2 (2006), 131–141. (Cited on pages 130 and 133.)

[72] Gallo, G., and Simeone, B. On the supermodular knapsack problem. *Math. Programming 45*, 2, (Ser. B) (1989), 295–309. (Cited on pages 21 and 86.)

[73] Garey, M. R., and Johnson, D. S. *Computers and intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Co., San Francisco, Calif., 1979. A Series of Books in the Mathematical Sciences. (Cited on pages 4, 9, 12, 23, 51, and 95.)

[74] Glover, F., Alidaee, B., Rego, C., and Kochenberger, G. One-pass heuristics for large-scale unconstrained binary quadratic problems. *European J. Oper. Res. 137*, 2 (2002), 272–287. Graphs and scheduling (Bendor, 1999). (Cited on page 13.)

[75] Glover, F., and Hao, J.-K. Efficient evaluations for solving large 0-1 unconstrained quadratic optimisation problems. *Int. J. Metaheuristics 1*, 1 (2010), 3–10. (Cited on page 13.)

[76] Grötschel, M., Lovász, L., and Schrijver, A. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1*, 2 (1981), 169–197. (Cited on page 85.)

[77] Guruswami, V., and Rudra, A. Soft decoding, dual BCH codes, and better list-decodable ϵ-biased codes. *IEEE Trans. Inform. Theory 57*, 2 (2011), 705–717. (Cited on page 91.)

[78] Hammer, P., Hansen, P., and Simeone, B. Roof duality, complementation and persistency in quadratic $0 - 1$ optimization. *Mathematical Programming 28* (1984), 121–155. (Cited on pages 3, 14, 86, 88, 89, and 90.)

[79] Hammer, P., and Rudeanu, S. *Boolean Methods in Operations Research and Related Areas*. Springer Verlag, 1968. (Cited on pages 1, 12, and 79.)

[80] Hammer, P. L. Some network flow problems solved with pseudo-boolean programming. *Operations Research 13*, 3 (1965), 388–399. (Cited on pages 86 and 99.)

[81] Hammer, P. L., and Kogan, A. Horn functions and their DNFs. *Inf. Process. Lett. 44*, 1 (1992), 23–29. (Cited on page 29.)

[82] Hammer, P. L., and Kogan, A. Optimal compression of propositional Horn knowledge bases: complexity and approximation. *Artificial Intelligence 64*, 1 (1993), 131–145. (Cited on pages 5, 6, 30, and 74.)

[83] Hammer, P. L., and Kogan, A. Quasi-acyclic propositional horn knowledge bases: Optimal compression. *IEEE Trans. Knowl. Data Eng. 7*, 5 (1995), 751–762. (Cited on page 5.)

[84] Hansen, P., Jaumard, B., and Mathon, V. Constrained nonlinear 0-1 programming. *ORSA J. Comput. 5*, 2 (1993), 97–119. (Cited on page 13.)

[85] Hansen, P., and Meyer, C. Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem. *Discrete Appl. Math. 157*, 6 (2009), 1267–1290. (Cited on page 13.)

[86] Hellerstein, L., and Kletenik, D. On the gap between ess(f) and cnf_size(f). *Discrete Applied Mathematics 161*, 1-2 (2013), 19–27. (Cited on page 10.)

[87] Helmberg, C., and Rendl, F. Solving quadratic $(0, 1)$-problems by semidefinite programs and cutting planes. *Math. Programming 82*, 3, Ser. A (1998), 291–315. (Cited on page 13.)

[88] Huttenlocher, D., Medioni, G., and Rehg, J., Eds. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA* (2009), IEEE. (Cited on pages 189 and 192.)

[89] Impagliazzo, R., and Paturi, R. On the complexity of k-SAT. *J. Comput. System Sci. 62*, 2 (2001), 367–375. Special issue on the Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999). (Cited on pages 8, 9, and 66.)

[90] Impagliazzo, R., Paturi, R., and Saks, M. E. Size-depth tradeoffs for threshold circuits. *SIAM J. Comput. 26*, 3 (1997), 693–707. (Cited on page 150.)

[91] Ishikawa, H. Higher-order clique reduction in binary graph cut. In Huttenlocher et al. [88], pp. 2993–3000. (Cited on pages 14, 15, 17, 19, 22, 95, 100, 101, 109, 118, 123, 141, and 165.)

[92] Ishikawa, H. Transformation of general binary MRF minimization to the first-order case. *IEEE Trans. Pattern Anal. Mach. Intell. 33*, 6 (2011), 1234–1249. (Cited on pages 14, 15, 17, 19, 22, 95, 100, 101, 109, 118, 123, 141, and 165.)

[93] Itai, A., and Makowsky, J. A. Unification as a complexity measure for logic programming. *J. Log. Program. 4*, 2 (1987), 105–117. (Cited on page 5.)

[94] Iwata, S., Fleischer, L., and Fujishige, S. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. In *STOC* (2000), pp. 97–106. (Cited on page 85.)

[95] Jukna, S. *Extremal combinatorics: With applications in computer science*, second ed. Texts in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg, 2011. (Cited on pages 1, 23, and 80.)

[96] Jukna, S. *Boolean function complexity: Advances and frontiers*, vol. 27 of *Algorithms and Combinatorics*. Springer, Heidelberg, 2012. (Cited on pages 1, 80, 83, 110, and 167.)

[97] Kahl, F., and Strandmark, P. Generalized roof duality for pseudo-boolean optimization. In Metaxas et al. [116], pp. 255–262. (Cited on page 21.)

[98] Kahl, F., and Strandmark, P. Generalized roof duality. *Discrete Applied Mathematics 160*, 16-17 (2012), 2419–2434. (Cited on page 21.)

[99] Kearns, M. J., and Vazirani, U. V. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, 1994. (Cited on page 1.)

[100] Kim, K. H., and Roush, F. W. On a problem of Turán. In *Studies in pure mathematics.* Birkhäuser, Basel, 1983, pp. 423–425. (Cited on page 140.)

[101] Kolmogorov, V. Generalized roof duality and bisubmodular functions. *Discrete Applied Mathematics 160*, 4-5 (2012), 416–426. (Cited on page 22.)

[102] Kolmogorov, V., and Rother, C. Minimizing nonsubmodular functions with graph cuts - a review. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 7 (2007), 1274–1279. (Cited on pages 14, 19, 90, and 109.)

[103] Kolmogorov, V., and Zabih, R. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 2 (2004), 147–159. (Cited on pages 14, 15, 16, 85, and 97.)

[104] Korte, B., and Vygen, J. *Combinatorial optimization: Theory and algorithms*, fifth ed., vol. 21 of *Algorithms and Combinatorics.* Springer, Heidelberg, 2012. (Cited on pages 23, 90, and 155.)

[105] Kortsarz, G. On the hardness of approximating spanners. *Algorithmica 30*, 3 (2001), 432–450. Approximation algorithms for combinatorial optimization problems. (Cited on page 7.)

[106] Kumar, R., and Sivakumar, D. Proofs, codes, and polynomial-time reducibilities. In *Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999).* IEEE Computer Soc., Los Alamitos, CA, 1999, pp. 46–53. (Cited on page 91.)

[107] Lasserre, J. B. Global optimization with polynomials and the problem of moments. *SIAM J. Optim. 11*, 3 (2000/01), 796–817. (Cited on page 22.)

[108] Lasserre, J. B. An explicit exact SDP relaxation for nonlinear 0-1 programs. In *Integer programming and combinatorial optimization (Utrecht, 2001)*, vol. 2081 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 2001, pp. 293–303. (Cited on page 22.)

[109] Laurent, M. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Math. Oper. Res. 28*, 3 (2003), 470–496. (Cited on page 22.)

[110] Lovász, L., and Schrijver, A. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim. 1*, 2 (1991), 166–190. (Cited on page 22.)

[111] Maghout, K. Sur la détermination des nombres de stabilité et du nombre chromatique d'un graphe. *C. R. Acad. Sci. Paris 248* (1959), 3522–3523. (Cited on page 12.)

[112] Maghout, K. Applications de l'algèbre de Boole à la théorie des graphes et aux programmes linéaires et quadratiques. *Cahiers Centre Études Rech. Opér. 5* (1963), 21–99. (Cited on page 12.)

[113] Maier, D. Minimum covers in relational database model. *J. ACM 27*, 4 (1980), 664–674. (Cited on pages 5 and 6.)

[114] Mendelson, E. *Introduction to mathematical logic*, fifth ed. Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, 2010. (Cited on page 4.)

[115] Merz, P., and Freisleben, B. Greedy and local search heuristics for unconstrained binary quadratic programming. *J. Heuristics 8*, 2 (2002), 197–213. (Cited on page 13.)

[116] Metaxas, D. N., Quan, L., Sanfeliu, A., and Gool, L. J. V., Eds. *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011* (2011), IEEE. (Cited on pages 187 and 190.)

[117] Minoux, M. Ltur: A simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Inf. Process. Lett. 29*, 1 (1988), 1–12. (Cited on page 5.)

[118] Minsky, M., and Papert, S. *Perceptrons - an introduction to computational geometry.* MIT Press, 1987. (Cited on pages 83 and 150.)

[119] Moshkovitz, D., and Raz, R. Two-query PCP with subconstant error. *J. ACM 57*, 5 (2010), Art. 29, 29. (Cited on pages 9, 36, 69, 70, and 75.)

[120] Nemhauser, G. L., and Trotter, Jr., L. E. Vertex packings: structural properties and algorithms. *Math. Programming 8* (1975), 232–248. (Cited on page 89.)

[121] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions. I. *Math. Programming 14*, 3 (1978), 265–294. (Cited on pages 21 and 86.)

[122] O'Donnell, R. *Analysis of Boolean Functions.* Cambridge University Press, Cambridge, 2014. (Cited on page 1.)

[123] Papadimitriou, C. H. *Computational complexity.* Addison-Wesley Publishing Company, Reading, MA, 1994. (Cited on page 4.)

[124] Parrilo, P. A. Semidefinite programming relaxations for semialgebraic problems: Algebraic and geometric methods in discrete optimization. *Math. Program. 96*, 2, Ser. B (2003), 293–320. (Cited on page 22.)

[125] Picard, J.-C., and Queyranne, M. On the integer-valued variables in the linear vertex packing problem. *Math. Programming 12*, 1 (1977), 97–101. (Cited on page 89.)

[126] Ramalingam, S., Russell, C., Ladicky, L., and Torr, P. H. S. Efficient minimization of higher order submodular functions using monotonic boolean functions. *CoRR abs/1109.2304* (2011), 1–25. (Cited on page 15.)

[127] Rosenberg, I. Reduction of bivalent maximization to the quadratic case. Tech. rep., Centre d'Etudes de Recherche Operationnelle, 1975. (Cited on pages 3, 13, 85, 92, 93, 94, and 101.)

[128] Roth, S., and Black, M. J. Fields of experts. *International Journal of Computer Vision 82*, 2 (2009), 205–229. (Cited on pages 85 and 108.)

[129] Rother, C., Kohli, P., Feng, W., and Jia, J. Minimizing sparse higher order energy functions of discrete variables. In Huttenlocher et al. [88], pp. 1382–1389. (Cited on pages 15, 17, 98, and 99.)

[130] Rother, C., Kolmogorov, V., Lempitsky, V. S., and Szummer, M. Optimizing binary MRFs via extended roof duality. In *CVPR* (2007), IEEE Computer Society. (Cited on page 14.)

[131] Saks, M. E. Slicing the hypercube. In *Surveys in combinatorics, 1993 (Keele)*, vol. 187 of *London Math. Soc. Lecture Note Ser.* Cambridge Univ. Press, Cambridge, 1993, pp. 211–255. (Cited on page 150.)

[132] Schrijver, A. *Theory of linear and integer programming.* Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, Ltd., Chichester, 1986. A Wiley-Interscience Publication. (Cited on pages 23 and 155.)

[133] Schrijver, A. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B 80*, 2 (2000), 346–355. (Cited on page 85.)

[134] Sheldon, D., and Young, N. E. Hamming approximation of NP witnesses. *Theory Comput. 9* (2013), 685–702. (Cited on page 91.)

[135] Sherali, H. D., and Adams, W. P. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math. 3*, 3 (1990), 411–430. (Cited on page 22.)

[136] Sherali, H. D., and Adams, W. P. Reformulation-linearization techniques for discrete optimization problems. In *Handbook of combinatorial optimization, Vol. 1*. Kluwer Acad. Publ., Boston, MA, 1998, pp. 479–532. (Cited on page 13.)

[137] Sherali, H. D., and Adams, W. P. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, vol. 31 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, 1999. (Cited on page 13.)

[138] Sidorenko, A. What we know and what we do not know about Turán numbers. *Graphs Combin. 11*, 2 (1995), 179–199. (Cited on page 140.)

[139] Sidorenko, A. Upper bounds for Turán numbers. *J. Combin. Theory Ser. A 77*, 1 (1997), 134–147. (Cited on page 140.)

[140] Siu, K.-Y., Roychowdhury, V., and Kailath, T. *Discrete neural computation: A theoretical foundation*. Prentice Hall Information and System Sciences Series. Prentice Hall PTR, Englewood Cliffs, NJ, 1995. With a foreword by Marvin Minsky. (Cited on page 146.)

[141] Strang, G. *Introduction to Linear Algebra, 4th ed.*, vol. 134 of *Encyclopedia of Mathematics and its Applications*. Wellesley-Cambridge Press, 2009. (Cited on page 23.)

[142] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M. F., and Rother, C. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 6 (2008), 1068–1080. (Cited on page 101.)

[143] Tavares, G. *New Algorithms for Quadratic Unconstrained Binary Optimization (QUBO) with Applications in Engeneering and Social Sciences*. Ph.D. thesis, RUTCOR, Rutgers University, New Brunswick, NJ, USA, April 2008. (Cited on pages 23 and 90.)

[144] Umans, C. Hardness of approximating Σ_2^p minimization problems. In *40th Annual Symposium on Foundations of Computer Science (New York, 1999)*. IEEE Computer Soc., Los Alamitos, CA, 1999, pp. 465–474. (Cited on page 75.)

[145] van Lint, J. H., and Wilson, R. M. *A course in combinatorics*, second ed. Cambridge University Press, Cambridge, 2001. (Cited on pages 23 and 80.)

[146] Vazirani, V. V. *Approximation algorithms*. Springer, 2001. (Cited on pages 8 and 23.)

[147] Wang, C., and Williams, A. C. The threshold order of a Boolean function. *Discrete Appl. Math. 31*, 1 (1991), 51–69. (Cited on page 150.)

[148] Wang, D., and Kleinberg, R. Analyzing quadratic unconstrained binary optimization problems via multicommodity flows. *Discrete Appl. Math. 157*, 18 (2009), 3746–3753. (Cited on pages 14 and 92.)

[149] Williamson, D. P., and Shmoys, D. B. *The design of approximation algorithms.* Cambridge University Press, Cambridge, 2011. (Cited on pages 8 and 23.)

[150] Woodford, O. J., Torr, P. H. S., Reid, I. D., and Fitzgibbon, A. W. Global stereo reconstruction under second-order smoothness priors. *IEEE Trans. Pattern Anal. Mach. Intell. 31*, 12 (2009), 2115–2128. (Cited on page 108.)

[151] Živný, S., Cohen, D. A., and Jeavons, P. G. The expressive power of binary submodular functions. *Discrete Applied Mathematics 157*, 15 (2009), 3347–3358. (Cited on pages 21, 22, and 99.)

[152] Živný, S., Cohen, D. A., and Jeavons, P. G. The expressive power of binary submodular functions. In *MFCS* (2009), R. Královic and D. Niwinski, Eds., vol. 5734 of *Lecture Notes in Computer Science*, Springer, pp. 744–757. (Cited on page 99.)

[153] Živný, S., and Jeavons, P. G. Classes of submodular constraints expressible by graph cuts. *Constraints 15*, 3 (2010), 430–452. (Cited on page 22.)