Noname manuscript No.
(will be inserted by the editor)

Solving Non-linear Boolean Equation Systems by Variable Elimination

Bjørn Greve · Øyvind Ytrehus · Håvard Raddum · Gunnar Fløystad

Abstract In this paper we study Boolean equation systems, and how to eliminate variables from them while bounding the degree of polynomials produced. A procedure for variable elimination is introduced, and we relate the techniques to Gröbner bases and XL methods. We prove that by increasing the degree of the polynomials in the system by one for each variable eliminated, we preserve the solution space, provided that the system satisfies a particular condition. We then estimate how many variables we need to eliminate in order to solve the resulting system by re-linearization, and show that we get complexities lower than the trivial brute-force $\mathcal{O}(2^n)$ when the system is overdetermined.

Keywords Systems of Boolean equations · Elimination of variables · Syzygies · Gröbner bases · Solving algorithm

1 Introduction

We consider non-linear systems of Boolean equations,

$$\begin{aligned} f_1(x_0,\ldots,x_{n-1}) &= 0 \\ f_2(x_0,\ldots,x_{n-1}) &= 0 \\ &\vdots \\ f_m(x_0,\ldots,x_{n-1}) &= 0 \end{aligned} \tag{1}$$

Bjørn Greve
Norwegian Defence Research Establishment, Kjeller, Norway

Øyvind Ytrehus · Håvard Raddum
Simula UiB, Bergen, Norway

Gunnar Fløystad
Department of Mathematics, University of Bergen, Norway

where each f_i is of bounded degree. How to solve such systems has been studied extensively, and it is well known that the problem of solving (1) is NP-hard in the general case. However, since the solving process has numerous important applications, many attempts have been made to derive algorithms that can provide solutions within a practical time frame for moderate-sized versions of the problem.

In the literature on solving systems of Boolean equations it is customary to consider quadratic systems of equations, despite the fact that most proposed algorithms also work on systems of equations with higher degrees than 2. The motivation for focusing on quadratic equations is that this is the simplest case beyond linear equation systems, as well as the fact that there exist general methods to transform a system of arbitrarily high degree equations, into an equivalent quadratic system by introducing new "auxiliary" variables [28, p. 47].

1.1 Previous work

In [6] and [7] the authors introduce the XL and XSL algorithms, respectively. The basic idea in these papers is to multiply equations with enough monomials to re-linearize the whole system of Boolean equations, i.e. to solve a linear system of equations where each monomial is considered as an independent variable. The best known solving method for Boolean equation systems is Faugère's F4 algorithm [8]. This algorithm is similar to XL in the sense that equations are multiplied with monomials, but F4 cleverly avoids multiplying equations with "useless" monomials that will result in redundant equations.

The application of Gröbner base theory in connection with Boolean polynomial rings $\mathbb{F}_2[x_0, \ldots, x_{n-1}]/(x_i^2 + x_i)$ was first considered by [5] and by [14]. Computations with Boolean Gröbner bases have been implemented in Singular [4] and `PolyBoRi` [1]. However, computations to find Gröbner bases become computationally heavy since the polynomials quickly grow in degrees and number of terms.

In [11] the authors instead consider Boolean border bases. Border bases exist for ideals I in polynomial rings $k[x_0, \ldots, x_{n-1}]$ such that the quotient ring is finite dimensional as a k-vector space. For an ideal I in a Boolean polynomial ring, the quotient ring is always finite. Since a Boolean polynomial ring is different from an ordinary polynomial ring, this suggests an approach using a tailored version of Boolean border bases, which is followed in [11]. A rationale for this approach is that the resulting algorithm controls and slows the growth of the "computational universe", the set of terms involved in all the linear algebra and computations. Nevertheless one must also here expect to get polynomials of large degrees.

1.2 Contributions

The approach in our paper is also to multiply equations with monomials, but we bound the degree at $\leq d$ for some d to control the complexity, and we do not multiply all the f_i's with all monomials. In this paper we are focused on *variable elimination*, which is not the approach taken in the Buchberger's, XL, or F4 algorithms. A system of polynomial equations in relatively few variables can be solved by simple exhaustive search. In this paper, we try to extract such systems, by investigating how to eliminate variables while working only with polynomials of degree $\leq d$.

When we limit the degree and the number of variables decreases, the resulting equation systems become easier to solve and they require less memory to keep in storage. When enough variables have been eliminated, we may increase the allowed degree without needing more memory than what we used to start with.

We propose and investigate an elimination procedure, where the degrees of the modified equation sets grow slowly. The main contribution is a proof that, under some mild conditions, eliminating k variables from a quadratic equation system and increasing the degree of the modified equations by k, the solution space of the system will be preserved. This leads to a new algorithm for solving Boolean equation systems, with favorable complexity analysis. The algorithm has been tested on some cases of random equation systems, and it is verified that theory and practice align very well.

1.3 Paper structure

The paper is structured as follows: Section 2 introduces the basic notation used in the paper and elaborates on prior work. In Section 4 we introduce *normalization, resultants, coefficient constraints* and *syzygies*, the basic tools and concepts we use for variable elimination. In Section 5 we combine the tools to construct a method for variable elimination, which leads to a procedure for solving a system of Boolean equations. Section 6 follows an alternative approach, with the motivation to limit the growth of degrees. We show that when using the resulting algorithm, no false solutions will enter the modified systems, given some mild condition. Complexity issues including experimental results are presented in Section 7, while Section 8 contains a conclusion and sketches some avenues for further research.

2 Notation and preliminaries

2.1 Table of notation

For reference, we list some of the notation that will be used throughout the paper.

Term	Meaning
n	Number of variables
m	Number of polynomials or equations
$x_0, \ldots, x_{n-1}$	The variables of the set of equations
$\mathbb{B}[x_j, \ldots, x_{n-1}]$	$\mathbb{F}_2[x_j, \ldots, x_{n-1}]/(x_i^2 + x_i \vert i = j, \ldots, n-1)$
t	Monomial term
f^i	Polynomial of degree i
F	Set of polynomials, or equations on the form $f = 0$
F^i	Set of polynomials, or equations, all of degree i
$F^{\leq i}$	Set of polynomials, or equations, all of degree $\leq i$
L^i	Set of all monomials of degree i
$L^{\leq i}$	Set of all monomials of degree $\leq i$
$\langle F \rangle$	The linear span of the polynomials in F
$F_{\overline{x_i}}$	Maximal subset of F so that no $f \in F_{\overline{x_i}}$ depends on x_i
F_{x_i}	Maximal subset of F so that $\langle F_{x_i} \rangle \cap F_{\overline{x_i}} = \emptyset$
$Z(F)$	Set of zero points for all polynomials in F
$I(F)$	The ideal generated by F
π_i	Projection omitting coordinates $0, \ldots, i$, that is, $\pi_i : \mathbb{F}_2^n \to \mathbb{F}_2^{n-i-1}$
$M(F)$ or $M_d(F)$	Macaulay matrix corresponding to F or $F^{\leq d}$

2.2 Definitions and notation

A monomial is a product $x_{i_1} \cdots x_{i_d}$ of d distinct (because $x^2 = x$) variables, where d is the degree of this monomial. The degree of a polynomial

$$p = \sum_i m_i$$

where the m_i's are distinct monomials, is the maximum degree over the monomials in p. Given a set of polynomial equations

$$F = \{f_i(x_0, \ldots, x_{n-1}) = 0 | i = 1, \ldots, m\},$$

our objective is to find its set of solutions in the space $\mathbb{F}_2^n$. The approach we take in this paper is to solve the system of equations by eliminating variables.

Consider the projection which omits the first coordinate:

$$\pi_0 : \mathbb{F}_2^n \to \mathbb{F}_2^{n-1}$$

$$(a_0, a_1, \ldots, a_{n-1}) \mapsto (a_1, \ldots, a_{n-1}),$$

We also consider a sequence of k projections

$$\mathbb{F}_2^n \to \mathbb{F}_2^{n-1} \to \cdots \to \mathbb{F}_2^{n-k},$$

where the i'th projection is denoted $\pi_i : \mathbb{F}_2^{n-i} \to \mathbb{F}_2^{n-i-1}$ for $0 \leq i < k$.

2.3 Other algorithms for solving non-linear systems of Boolean equations

For completeness, we end this section by outlining alternative approaches, which to the best of our knowledge collectively represent the state of the art of solving sets of Boolean quadratic equations. Most of these approaches are rather different from ours. Hence, the terminology used in the rest of this section is inherited from the respective papers, and is less convenient for our approach and not always consistent with the terminology that we use from Section 4 and onwards. Readers who are already familiar with this prior work, or who want to focus on our contributions, may skip to Section 4.

1. An algorithm faster than exhaustive search: In [23] the authors proposed an algorithm where they use the following techniques. We note that although the work from [23] also considers larger finite fields, we will restrict ourselves to $\mathbb{F}_2$ which is our case of interest.

The authors of [23] observe that the single polynomial

$$P_s(\mathbf{x}) = 1 - \prod_{i=1}^{m}(1 - f_i(\mathbf{x}))$$

is equivalent to the set of equations in (1), in the sense that for any $a \in \mathbb{F}_2^n$, $P_s(a) = 0$ iff $f_1(a) = \cdots = f_m(a) = 0$. For some appropriately chosen $n' < n$, we may define a polynomial $R \in \mathbb{F}_2[x_0, \ldots x_{n-n'-1}]$ as

$$R(x_0, \ldots x_{n-n'-1}) = \prod_{c \in \mathbb{F}_2^{n'}} P_s(x_0, \ldots x_{n-n'-1}, c).$$

It is easy to see that there is an $a = (b|c) \in \mathbb{F}_2^n$ such that $P_s(a) = 0$ if and only if there is a $b \in \mathbb{F}_2^{n-n'}$ such that $R(b) = 0$. In [23], it is noted that we can evaluate R on all $2^{n-n'}$ points b in $2^{n-n'} \cdot poly(n, m)$ time by applying probabilistic polynomial constructions presented in [27,13], together with an algorithm for efficient sums-of-monomials evaluation. The complexity of solving a system of m equations over $\mathbb{F}_2$ is shown in [23] to approach $\mathcal{O}(2^{0.8765n})$ (assuming that the number of solutions is less than $2^{0.8765n}$), which is faster than the brute force approach. This asymptotic behavior is independent of any heuristic hypothesis on the system of equations.

The approach of [23] has several limitations. First of all we note that the values of n must be large for the estimated complexity to hold, which means that it seems unlikely that the method will outperform exhaustive search for systems that are anywhere close to be solvable on a computer. Another drawback of this approach is that it cannot take advantage of large values of m compared to n since the method of [23] assumes independence between the polynomials.

2. Algebraic methods and Macaulay matrices [22,12]:

Let $I = (f_1, \ldots, f_m) \subseteq \mathbb{B}[x_0, \ldots, x_{n-1}]$ be the ideal describing the system of equations in (1) and let d be the maximum degree in F, with the corresponding Macaulay matrix $M_d(F)$ as introduced in the end of Section 4.

It was shown in [22] that one can use the Macaulay matrix to compute the Gröbner basis of I. If a system of equations is inconsistent, the constant 1 is a linear combination with polynomial coefficients, of the generators of the ideal, and if a system of equations has k solutions, then there exist at least $n - k$ linearly independent linear forms in the corresponding ideal. It follows that the low degree equations describing the solutions of the system can be obtained by reducing a Macaulay matrix in which the columns are sorted by total degree. The smallest D for which we can deduce linear equations from $M_D(F)$ is called the *degree of regularity* of the system of equations, formally defined as follows [16].

Definition 1 *Let $F = (f_1, \ldots f_m) \subseteq \mathbb{F}_2[x_0, \ldots, x_{n-1}]$ be a system of equations with finite number of solutions where $m \geq n$ defining a zero dimensional ideal $I(F)$. The degree of regularity of $I(F)$ is defined by*

$$d_{reg} = \min\{d \geq 0, \dim_K(\{f \in I, \deg(f) = d\}) = \binom{n+d-1}{d}\}$$

Determining the degree of regularity of a system of equations is not an easy problem. A special case is when a system of equations is *semi-regular*. In this case, [18] gives asymptotic estimates of the degree of regularity d_{reg} for semi-regular systems. In particular it can be shown that for semi-regular system, if $m \approx \alpha n$ for some fixed $\alpha \geq 1$, then $d_{reg} \approx M(\alpha)n$ where $M(\alpha)$ is an explicit decreasing function of α as n goes to infinity. This implies that when $\alpha = 1$ and if matrix reduction can be done with complexity $\mathcal{O}(n^2)$, the authors obtain an asymptotic complexity of $\mathcal{O}(2^{0.8728n})$ for solving a system of m equations in n variables.

Note that the algorithm depends on the system being semi-regular in contrast to [20,23], which means that particular systems may be harder to solve with this technique. It is conjectured that the complexity holds with a probability converging to 1 as n grows. It has also been estimated that these methods will not outperform exhaustive search before $n \geq 200$ ([20]).

3. Hybrid algorithms: To make algorithms better than the pure algebraic ones, *hybrid* algorithms combining exhaustive search and linear algebra on the Macaulay matrix have been proposed, including the BooleanSolve algorithm ([17]). The idea is to reduce the search space combined with the information provided by the Macaulay matrix, and can formally be described as follows:

1. Let $F = (f_1, \ldots f_m) \subseteq \mathbb{B}[x_0, \ldots, x_{n-1}]$ and choose a parameter $k \leq n$.
2. For each $a = (a_k, \ldots, a_{n-1}) \in \mathbb{F}_2^{n-k}$ and each i, compute the specialized polynomials $f_{i,a} = f_i(x_0, \ldots, x_{k-1}, a_k, \ldots, a_{n-1})$, corresponding to each f_i evaluated at a. This provides us with a system $F_a = \{f_{1,a}, \ldots, f_{m,a}\} \subseteq \mathbb{B}[x_0, \ldots, x_{k-1}]$.

3. Let $M_{d_{reg}}(F_a)$ be the Macaulay matrix of F_a in degree d_{reg}. Using this matrix, check if the system F_a admits a solution.
4. If no solution is found, continue with the next value of $a \in \mathbb{F}_2^{n-k}$.
5. If F_a admits a solution, find the solution of F_a by for example exhaustive search on the remaining variables $x_0, \ldots, x_{k-1}$.

The benefit from fixing values for some of the variables is that the size of the corresponding Macaulay matrices is reduced in addition to a reduction in the complexity to $\mathcal{O}(2^{n-k})$. Note that the degree of regularity decreases while $m/(n-k)$ increases, so the algorithm is conditional on how many times one has to check for new points $a \in \mathbb{F}_2^{n-k}$. This means that for most values of a, the system will not have any solution. Despite this, when running the algorithm one only needs to test whether the constant polynomial can be found in the Macaulay matrix, which means that one does not need to compute the full echelon form of the matrix.

In [17], the authors show that if one assumes that semi-regular systems behave like random systems and remain semi-regular during the algorithm, one can obtain a complexity of $\mathcal{O}(2^{(1-0.208\alpha)n})$. The values here are $m \approx \alpha n$ where $\alpha \geq 1$ and n goes to infinity. With $\alpha < 1.82$ together with best choice of $k = 0.55\alpha n$, the authors give an asymptotic complexity estimate of $\mathcal{O}(2^{0.792n})$. If $\alpha \geq 1.82$, the best choice is $n = k$ which means that no variables are specialized and the algorithm does not improve over the standard reduction of the full Macaulay matrix.

4. Crossbred algorithm optimizing the BooleanSolve algorithm: In [19] the authors present an algorithm which is an optimization of the BooleanSolve algorithm from [17]. The main observation is to avoid the most costly step of the algorithm when reducing the Macaulay matrix which is performed 2^{n-k} times. The idea is to perform the specialization of $n-k$ variables after working with the Macaulay matrix, which means that we can reduce a partial Macaulay matrix before specifying parts of the variables. The algorithm from [19] depends on three parameters: $D \geq 2$ (global degree), $1 \leq d < D$ (local degree) and $1 \leq k \leq n$ (number of variables to specialize). To make distinction on the degrees, for a polynomial $p \in \mathbb{B}[x_0, \ldots, x_{n-1}]$ the total degree of p in the variables $x_0, \ldots, x_{k-1}$ is denoted by $\deg_k(p)$. The description of the algorithm from [19] is as follows:

1. First construct the submatrix $M_{D,d,k}(F)$ of the full degree D Macaulay matrix $M_d(F)$, where the rows consists of products uf_i where $\deg_k(u) \geq d-1$.
2. Then construct the submatrix $M'_{D,d,k}(F)$ of $M_{D,d,k}(F)$, where the columns consist of monomials m with $\deg_k(m) > d$.
3. Search for elements $v_1, \ldots, v_r$ in the kernel of $M'_{D,d,k}(F)$, and compute the polynomials $p_i = v_i \cdot M_{D,d,k}(F)$. Note that the total degree of p_i is at most D, and $\deg_k(p) \leq d$.
4. For all $a = (a_k, \ldots, a_{n-1}) \in \mathbb{F}_2^{n-k}$
   (a) Create the degree d Macaulay matrix $M_d(F^*)$ corresponding to the polynomials in F evaluated at a.

(b) Evaluate p_i at a and append them to $M_d(F^*)$. Check if the resulting system in degree d can be solved in $x_0, \ldots, x_{k-1}$.

When $d = 1$, the idea is to generate new equations that are linear in $x_1, \ldots, x_k$ from the degree D Macaulay matrix with the implication that all monomials of degree larger than 1 in these variables are eliminated. This is achieved by computing elements in the kernel of $M'_{D,1,k}(F)$, from which the monomials containing more than one of the variables $x_0, \ldots, x_{k-1}$ have been removed. In practice there is a trade-off when choosing D: It has to be large enough for the reduced equations to exist, and small enough such that the Macaulay matrix does not become too large. Note that we only need k of the linear equations created from the equations p_i that are evaluated at $x_k, \ldots, x_{n-1}$, to check whether the system is solvable in $x_0, \ldots, x_{k-1}$.

When $d > 1$, the idea is similar: Construct new equations but of degree at most d in the variables $x_0, \ldots, x_{k-1}$, where the number of equations needed in order to solve the system at the end becomes larger. When d is larger than the degree d_{in} of the initial system, the initial equations can be included with the equations kept for specialization. In [19] the analysis for the optimal choices of parameters D, d, k for given m, n, d_{in} is only carried out in the case when $d = 1$ and the initial systems are quadratic. However, the authors demonstrate in practice that the algorithm beats exhaustive search when $n = m$ already at $n = 37$, which is lower than the previously expectation of $n \geq 200$ from [20].

3 Systems of Boolean equations and ideals

Any Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be written as a polynomial in the ring $\mathbb{B}[x_0, \ldots, x_{n-1}]$ (see [21, p. 346] for the basics on Boolean functions). Such a function is uniquely determined by the zero set

$$Z(f) = \{\mathbf{a} \in \mathbb{F}_2^n | f(\mathbf{a}) = 0\}.$$

Conversely, for any given subset Z of $\mathbb{F}_2^n$ there is a unique Boolean function with this as zero set. So there are one-to-one correspondences between Boolean functions, Boolean polynomials, and subsets of $\mathbb{F}_2^n$.

Lemma 2 *Let f and g be Boolean polynomials. Then:*

$$f \text{ is a multiple of } g \Leftrightarrow \text{ the zero sets } Z(f) \supseteq Z(g).$$

Proof The implication $\Rightarrow$ is clear. Suppose $Z(f) \supseteq Z(g)$ and let H be the difference set. It corresponds then to a Boolean polynomial h with zero set $Z(h) = H$. Then f and hg have the same zero set, and hence are equal. □

On the other hand, for $f \in \mathbb{B}[x_0, \ldots, x_{n-1}]$ we may write f on the form $f = \sum_{\alpha \in A} \prod_{i=0}^{n-1} x_i^{\alpha_i}$, where $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$ is a binary vector of length n and $A \subset \mathbb{F}_2^n$ specifies the monomials occurring in f. Any Boolean function

can be written in this form, which gives a different one-to-one correspondence between Boolean functions and subsets of $\mathbb{F}_2^n$. This establishes the following relationship

$$A \leftrightarrow f \leftrightarrow Z.$$

For a given function f, the relationship between the sets A and Z is however far from trivial.

For a set of polynomial equations (1), let $F = \{f_1, \ldots, f_m\}$. The polynomials in F generate an ideal $I = (f_1, \ldots, f_m) = I(F)$ in the ring $\mathbb{B}[x_0, \ldots, x_{n-1}]$. Let $Z(I)$ denote the zero set of this ideal, i.e, the set of points

$$Z(I) = \{\mathbf{a} \in \mathbb{F}_2^n | f(\mathbf{a}) = 0 \text{ for every } f \in I\}.$$

Lemma 3 *Let f, g be Boolean functions in $\mathbb{B}[x_0, \ldots, x_{n-1}]$. Then the following ideals are equal:*

$$(f, g) = (fg + f + g).$$

Proof Note that it is easy to verify that they have the same zero set. But it is not a priori clear that they are equal as ideals of Boolean polynomials. Clearly $(f, g) \supseteq (fg + f + g)$. We also have $Z(f, g) = Z(fg + f + g)$, since it is easy to check that $f(a) = g(a) = 0$ if and only if $f(a)g(a) + f(a) + g(a) = 0$. Thus the zero set $Z(f) \supseteq Z(fg + f + g)$. By Lemma 2 the Boolean function f is a multiple $h(fg + f + g)$ for some other Boolean function h, and similarly for g. Thus

$$(f, g) \supseteq (fg + f + g) \supseteq (f, g),$$

which shows that these ideals are equal. □

Corollary 4 *Any ideal $I = (f_1, \ldots, f_m)$ in $\mathbb{B}[x_0, \ldots, x_{n-1}]$ is a principal ideal. More precisely $I = (f)$ where*

$$f = 1 + \prod_{i=1}^{m} (f_i + 1).$$

Proof Let $I = (f_1, \ldots, f_m)$. By Lemma 3 this is equal to the ideal $(f_1 f_2 + f_1 + f_2, f_3, \ldots, f_m)$, with one generator less. We may continue the process for the remaining generators providing us in the end with $I = (f)$, where

$$f = 1 + \prod_{i=1}^{m} (f_i + 1).$$

□

Corollary 5 *For two ideals in $\mathbb{B}[x_0, \ldots, x_{n-1}]$ we have $I \supseteq J$ if and only if $Z(I) \subseteq Z(J)$. In particular $I = J$ if and only if $Z(I) = Z(J)$.*

Proof By Corollary 4 we have $I = (f)$ and $J = (g)$, where f and g are the respective principal generators. Clearly if $(f) \supseteq (g)$ then $Z(g) \supseteq Z(f)$. If the zero set of g contains the zero set of f, then $g = fh$ for some polynomial h. Hence $(f) \supseteq (g)$. □

Given an ideal $I \subset \mathbb{B}[x_0, \ldots, x_{n-1}]$, our aim is to find the ideal $I_1 \subset \mathbb{B}[x_1, \ldots, x_{n-1}]$ such that $Z(I_1) = \pi_0(Z(I))$. More generally, when eliminating more variables we aim to find the ideal $I_k \subset \mathbb{B}[x_k, \ldots, x_{n-1}]$, such that $Z(I_k) = \pi_{k-1} \circ (\cdots \circ (\pi_0(Z(I))))$. Since the degree of the polynomials can grow very quickly when eliminating variables, in practice we have to settle for computing an ideal J, as large as possible given computational restrictions, which is contained in I_k.

Let us first describe precisely the ideal I_k whose zero set is the sequence of projections $\pi_{k-1} \circ (\cdots \circ (\pi_0(Z(I))))$. This corresponds to what is known as the *elimination ideal* $I \cap \mathbb{B}[x_k, \ldots, x_{n-1}]$.

Lemma 6 *Let $I_k \subseteq \mathbb{B}[x_k, \ldots, x_{n-1}]$ be the ideal of all Boolean functions vanishing on $\pi_{k-1} \circ (\cdots \circ (\pi_0(Z(I))))$. Then $I_k = I \cap \mathbb{B}[x_k, \ldots, x_{n-1}]$.*

Proof We show this for the case when eliminating one variable, the general case follows in a similar manner. Clearly $I_1 \supseteq I \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$. Conversely let $f \in \mathbb{B}[x_1, \ldots, x_{n-1}]$ vanish on $\pi_0(Z(I))$. Then f must also vanish on $Z(I)$, where f is regarded as a member of the extended ring $\mathbb{B}[x_0, \ldots, x_{n-1}]$. Therefore $f \in I$ by Corollary 5. □

A standard technique for computing elimination ideals is to use Gröbner basis computation, which iteratively eliminates one *monomial* at the time. In fact, to compute elimination ideals via Gröbner bases one has to compute the full Gröbner basis before performing elimination. Computing Gröbner bases is computationally heavy because the degrees of the polynomials grow rapidly over the iterations. To deal with this problem, we propose new tools which also restrict the degree of the polynomials.

Our solution is to not use all polynomials during elimination but only compute with those that do not produce new polynomials of high degree. We denote an ideal where the degree is restricted to some d by I^d, whereas $I^{\leq n}$ means that we allow all degrees. The benefit from our approach is that the elimination process has much lower complexity, at the cost of the following disadvantage.

Discarding polynomials of degree $> d$ gives an ideal $I^{\leq d}$ that is only contained in the elimination ideal $I^{\leq n} = I \cap \mathbb{B}[x_j, \ldots, x_{n-1}]$. It follows that $Z(I^{\leq d})$ of the eliminated system contains all the projected solutions of the original set of equations, but it will also contain "false" solutions which will not fit the ideal I when lifted back to $\mathbb{F}_2^n$, regardless of which values we assign to the eliminated variables. Since the proposed procedure expands the solution space to include false solutions, the worst case scenario is when we end up with an empty set of polynomials after eliminating a sequence of variables. This means that all constraints given by the initial I have been removed, and we end up with the complete $\mathbb{F}_2^{n-k}$ as a solution space.

It is important to note that not discarding any polynomials will provide, by Lemma 6, only the true solutions to the set where variables have been eliminated. Hence the solutions can then be lifted back to the solutions of

the initial ideal I. The drawback of this approach is that we must be able to handle arbitrarily large polynomials, i.e., high computational complexity.

Thus there is a tradeoff between the maximum degree d allowed, and the proximity between the "practical" ideal $I^{\leq d}$ and the true elimination ideal $I^{\leq n}$. In the following, we are going to dig deeper into this trade-off.

In the remainder of the paper we adopt the following non-standard notation: For a polynomial f^i, the superscript i indicates that f has degree i, and does not mean f raised to the power i. The reason for this is that since we bound the degree of polynomials we work with, it is important to keep in mind which degree the various polynomials and monomials have. Therefore we indicate this with a superscript. Similarly, F^i for a set of polynomials indicates that all polynomials in F have degree i.

We can split the initial system F into d sets according to degree:

$$F^d = \{f_1^d, \ldots, f_{r_d}^d\}, F^{d-1}, \ldots, F^2, F^1 = \{f_1^1, \ldots, f_{r_1}^1\},$$

The polynomials in $F^d, F^{d-1}, \ldots, F^1$ together generate the ideal

$$I = (F^d, F^{d-1}, \ldots, F^1)$$

. The focus in this paper is to eliminate variables from a system of Boolean equations. From here on we therefore always assume that $F^1 = \emptyset$ and omit it from further analysis, since otherwise we could just use any (linear) polynomial in F^1 to eliminate a variable without changing the degree of the system.

For a set of polynomials F, we use the notation F_{x_0} to mean that all polynomials in F depend on the variable x_0. Similarly, $F_{\overline{x_0}}$ indicates that x_0 does not appear in any of the polynomials in F.

4 Elimination Techniques

Our approach to solve the system (1) is to eliminate variables so that we find degree $\leq d$ polynomials in I_k, in smaller and smaller Boolean rings $\mathbb{B}[x_k, \ldots, x_{n-1}]$. Our objective is to find as many polynomials in the ideal $I(F)$ as possible *computing only with polynomials of degrees* $\leq d$. This limits both storage and computational complexity.

Let $F = (f_1, \ldots, f_m)$ be a set of Boolean equations in $\mathbb{B}[x_0, \ldots, x_{n-1}]$ of degree $\leq d$, and denote by $\langle F \rangle$ the vector space spanned by the polynomials in F, where each monomial is regarded as a coordinate. Let $L^{\leq i} = L^i \cup \ldots \cup L^1 \cup L^0$ be the set of monomials of degree $\leq i$, that is, $L^0 = \{1\}, L^1 = \{x_0, \ldots, x_{n-1}\}$, etc. Then $\langle L^{\leq i} \rangle$ is the vector space spanned by the Boolean polynomials of degree $\leq i$.

For the set of polynomials F^i, $(i = 2, \ldots, d-1)$ we consider the sets $L^{\leq j} F^i$ of all products lf where $l \in L^{\leq j}$ and $f \in F^i$. Since we are bounding the maximal degree to be d, we can be certain to form any such product provided that $i + j \leq d$. With this constraint, the total set of polynomials we can construct in our analysis is the vector space generated from the following basis:

$$F^d \cup L^{\leq 1}F^{d-1} \cup L^{\leq 2}F^{d-2} \cup \cdots \cup L^{\leq d-2}F^2.$$

For the purpose of this paper, we shall need two types of total orders on the monomials in a Boolean ring.

1. An x_0-elimination order: If t_1 and t_2 are monomials where t_1 contains x_0 while t_2 does not, then $t_1 > t_2$. An example is the lexicographic order.
2. A degree order: If t_1 has degree larger than t_2 then $t_1 > t_2$.

As part of the variable elimination process, it will be necessary to split a set of polynomials according to dependency on x_0 and possibly according to degree. The procedures for this can be implemented in terms of row reduction on the *Macaulay matrix* of the given set: Its columns are indexed by the monomials according to the total order we use, and the rows are indexed by the polynomials in our set. The entries are the coefficients of the monomials of the polynomials. A basic procedure is:

$SplitVariable(F, x_0)$: We use an x_0-elimination order. We perform Gaussian elimination on the Macaulay matrix of F on all the columns indexed by monomials containing x_0 (which form an initial segment of columns) to get these columns in row-reduced echelon form. The rest of the columns are not of concern in this procedure. The outputs are sets of polynomials F_{x_0} consisting of polynomials which containing x_0-terms (the upper rows of the resulting matrix), and $F_{\overline{x_0}}$ consisting of those polynomials not containing x_0-terms (the remaining lower rows of the resulting matrix).

4.1 Normalization

The purpose of normalization is to eliminate many monomials containing x_0 from a given polynomial, using a set of lower-degree polynomials depending on x_0 as a basis. Suppose we have an x_0-elimination ordering. Let $f^i \in F^i_{x_0}$ be given, and let $G_{x_0} = \{g_1, \ldots, g_r\}$ be a set of polynomials of degree $\leq i$ which all contain x_0. In general we say that f^i is normalized with respect to G_{x_0} if no term in f^i is divisible by any leading term (with respect to the x_0-elimination order) of the polynomials in G_{x_0}, and we write $f^{i,norm}$ when we need to stress that f^i is normalized (with respect to some basis).

Without restriction on the degree, it is easy to create $f^{i,norm}$ from f^i and G_{x_0}. Simply run through the polynomials in G_{x_0} and check if some monomial m_f in f^i is divisible by a leading term of some $g_j \in G_{x_0}$. If $m_f = q * in(g_j)$, eliminate m_f by adding qg_j to f^i. Doing this successively for all polynomials in G_{x_0} will produce $f^{i,norm}$.

Since we need to restrict the degree of the polynomials we work with to be at most d, extra care has to be taken. The leading term $in(g_j)$ of some g_j may not be of the highest degree among the monomials in g_j, for instance if $g_j = x_0x_3 + x_1x_3x_4$. When this happens we can only eliminate the m_f in f^i where $deg(q) + deg(g_j) \leq d$, since we do not want to possibly introduce terms

of degree larger than d. We, therefore, give the following definition of what normalization means in this paper.

Definition 7 *Let $f^i \in F^i_{x_0}$ and $d \geq i$ be given, and let G_{x_0} be a set of polynomials all depending on x_0 with an x_0-elimination ordering. We say that f^i is* normalized *with respect to G_{x_0} if no term m_f in f^i can be written as $m_f = q * in(g)$ for any $g \in G_{x_0}$ where $deg(q) + deg(g) \leq d$. When writing $F^{i,norm}$ for a set of polynomials we mean that every polynomial in F^i is normalized with respect to some basis, and that all the polynomials have distinct initial terms.*

In our algorithm, we combine normalization and Gaussian reduction to get polynomials with distinct initial terms. We start with $F^2_{x_0}$ and perform Gaussian reduction. Denote the new set of polynomials $F^{2,norm}_{x_0}$. The general procedure is as follows.

NORMALIZE($F^i_{x_0}$)

1. The polynomials of $F^i_{x_0}$ are put into the rows of a Macaulay matrix. Perform *SplitVariable*($F^i_{x_0}, x_0$) to get new set $F^i_{x_0}$ where the x_0-part is in row-reduced echelon form. If there are polynomials without x_0-terms, put these into $F^i_{\overline{x_0}}$.
2. For each f^i in $F^i_{x_0}$ normalize it with respect to $\cup_{j=2}^{i-1} F^{j,norm}_{x_0}$.
3. Again perform $SplitVariable(F^i_{x_0}, x_0)$ to get the x_0-part in row-reduced echelon form. The polynomials containing x_0 form the set $F^{i,norm}_{x_0}$. If any polynomials do not contain x_0, then add these polynomials to the set $F^i_{\overline{x_0}}$.

4.2 Resultants

The second tool for elimination we use is resultants, which eliminates x_0 from a pair of polynomials. Let $f_1 = a_1 x_0 + b_1$ and $f_2 = a_2 x_0 + b_2$ be two polynomials in $\mathbb{B}[x_0, \ldots, x_{n-1}]$. The variable x_0 has been factored out so the polynomials a_i and b_i are in $\mathbb{B}[x_1, \ldots, x_{n-1}]$. In order to find the resultant, form the 2×2 Sylvester matrix of f_1 and f_2 with respect to x_0

$$\mathrm{Syl}(f_1, f_2, x_0) = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix}$$

The resultant of f_1 and f_2 with respect to x_0 is then simply the determinant of this matrix, and hence a polynomial in $\mathbb{B}[x_1, \ldots, x_{n-1}]$:

$$\mathrm{Res}(f_1, f_2, x_0) = \det(\mathrm{Syl}(f_1, f_2, x_0)) = a_1 b_2 + a_2 b_1$$

We note that $\mathrm{Res}(f_1, f_2, x_0) = a_2 f_1 + a_1 f_2$, too, which means that the resultant is indeed in the ideal generated by f_1 and f_2. Moreover, $\mathrm{Res}(f_1, f_2, x_0)$ is in the elimination ideal I_1.

When both f_1 and f_2 are quadratic, then the a_i's are linear so the degree of the resultant $\mathrm{Res}(f_1, f_2, x_0)$ is at most 3. More generally, since we are

restricting the degree to at most d we can not take the resultants between all pairs of polynomials from F. Say $f_1 \in F^i$ and $f_2 \in F^j$. Then we have $\deg(a_1) \leq i-1$, $\deg(a_2) \leq j-1$, $\deg(b_1) \leq i$ and $\deg(b_2) \leq j$. Hence it follows that $\deg(\mathrm{Res}(f_1, f_2, x_0)) \leq i+j-1$, which is certain to respect the degree bound only when $i+j \leq d+1$. Hence we can take resultants of all polynomial pairs from the sets F^i, F^j as long as $i+j \leq d+1$:

$$\mathrm{Res}(F^i, F^j, x_0) = \{\mathrm{Res}(f, g, x_0) | f \in F^i, g \in F^j, i+j \leq d+1\}.$$

For a set F of polynomials of differing degree, we split F into $F^d, \ldots, F^2$ and denote

$$Res(F, x_0) = \cup_{i+j \leq d+1} Res(F^i, F^j, x_0).$$

It is easy to see that $Res(F, x_0)$ is contained in the elimination ideal $I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$, but this inclusion is in general strict.

Relation to Gröbner bases: Computing a resultant may be realized in terms of Gröbner bases, by running a slightly modified Buchberger's algorithm. We show this in the case of quadratic polynomials for ease of exposition, and the general case follows the same reasoning. Assume we have a lexicographic order on the monomials, where $x_0 > x_1 > x_2$ and all other variables are smaller than x_2. Let

$$\begin{aligned} f_1 &= (x_1 + a_1)x_0 + b_1 \\ f_2 &= (x_2 + a_2)x_0 + b_2 \end{aligned},$$

where a_1 and a_2 are linear combinations where neither x_1 nor x_2 appears, and b_1, b_2 do not depend on x_0. The first step of Buchberger's algorithm is to compute the S-polynomial of f_1 and f_2, and then reduce it modulo the polynomials already in the basis. Here we only consider reduction modulo the sub-basis consisting of (f_1, f_2). The leading monomial of f_1 is x_0x_1 and the leading monomial of f_2 is x_0x_2. The S-polynomial is then

$$S = x_2 f_1 + x_1 f_2 = x_0x_1a_2 + x_0x_2a_1 + x_1b_2 + x_2b_1.$$

The next step is to divide S on the two polynomials (f_1, f_2) to find the remainder that should be added to the sub-basis. The highest monomials in S are $x_0x_1x_k$, for the various x_k appearing in a_2. All of these monomials are divisible by the leading monomial of f_1. After a number of steps in the division we arrive at

$$S = a_2 f_1 + x_0x_2a_1 + x_0a_1a_2 + x_1b_2 + (x_2 + a_2)b_1.$$

The remainder now has $x_0x_2x_k$ as the highest monomials, for the different x_k appearing in a_1. All of these terms are divisible by the leading monomial of f_2. When continuing the division algorithm the $x_0a_1a_2$-terms will appear twice and cancel, so we arrive at

$$S = a_2 f_1 + a_1 f_2 + (x_1 + a_1)b_2 + (x_2 + a_2)b_1.$$

We see that the remainder (the last two terms) when reducing the S-polynomial only modulo the sub-basis (f_1, f_2) is exactly the resultant.

In a full implementation of an algorithm computing Gröbner bases the resultant would be reduced against all other polynomials in the basis. We are only interested in eliminating the terms containing x_0, so computing resultants are more efficient and straight to the point for our purpose than computing S-polynomials followed by a full reduction.

4.3 Coefficient constraints

The next object we introduce is what is exactly required to close the gap that the resultants leave in the elimination ideal.

Definition 8 *Let $I(F) = (f_1, \ldots, f_m) \subseteq \mathbb{B}[x_0, \ldots, x_{n-1}]$, and write each f_i as $f_i = a_i x_0 + b_i$, where neither a_i nor b_i depends on x_0. We define the* coefficient constraint *ideal as*

$$Co(F, x_0) = (b_1(a_1 + 1), b_2(a_2 + 1), \ldots, b_m(a_m + 1)).$$

We note that the degrees of the generators of $\mathrm{Co}(F, x_0)$ have the same degrees as the generators of $Res(F^i, x_0)$, and it follows that we can form the coefficient constraints for F^i as long as $2i \leq d + 1 \iff i \leq (d + 1)/2$. In the case when $I(F)$ consists of quadratic polynomials, the generators of $\mathrm{Co}(F, x_0)$ will be polynomials of degree ≤ 3.

Similarly to resultants, the coefficient constraints can also be realized through familiar Gröbner basis computations when we make use of the field polynomials $x_i^2 + x_i$. We show this for a quadratic polynomial. Consider when we have the lexicographic order where $x_0 > x_1$ and x_1 is bigger than all other variables. Let

$$f = (x_1 + a)x_0 + b,$$

where all terms in a are smaller than x_1. Compute the S-polynomial of f and $x_1^2 + x_1$:

$$S = x_1 f + x_0(x_1^2 + x_1) = x_0 x_1 a + x_0 x_1 + x_1 b.$$

The highest monomials in S are $x_0 x_1 x_j$ for the different x_j appearing in a, all of which are divisible by the leading monomial in f. We also have the quadratic monomial $x_0 x_1$ in S. Dividing S by f then gives

$$S = (a + 1)f + x_0(a^2 + a) + (x_1 + a + 1)b.$$

All cross-terms $x_j x_k$ in the squared linear combination a^2 will cancel since our base field has characteristic 2. Hence $x_0(a^2 + a)$ will be a sum of terms $x_0 x_j^2 + x_0 x_j$ where the highest monomials are the $x_0 x_j^2$ given by the elimination order. These are all divisible by the leading monomials of the field polynomials

$x_j^2+x_j$. Continuing the division algorithm using the field equations will remove all terms in $x_0(a^2+a)$. Assuming a starts with $x_j + x_k + \ldots$ we get

$$S = (a+1)f + x_0(x_j^2 + x_j) + x_0(x_k^2 + x_k) + \ldots + (x_1 + a + 1)b.$$

This gives a remainder $(x_1 + a + 1)b$ that does not depend on x_0, and it is exactly the coefficient constraint. In the same way as with resultants, coefficient constraints allow us to jump straight to the answer of reducing an S-polynomial modulo a given basis, instead of going through all the steps of the division in a Gröbner basis algorithm.

An important fact is that the zero set of $Co(F, x_0)$ contains the projection of the zero set of $I(F)$ onto $\mathbb{F}_2^{n-1}$.

Lemma 9 *$Z(Co(F, x_0)) \supseteq \pi_0(Z(I(F)))$.*

Proof A point $\mathbf{p} \in \mathbb{F}_2^{n-1}$ is not in the zero set $Z(\mathrm{Co}(F, x_0))$ when for some i we have $a_i(\mathbf{p}) = 0$ and $b_i(\mathbf{p}) = 1$. For both the two liftings of $\mathbf{p}$ to $\mathbb{F}_2^n$: $\mathbf{p}_0 = (0, \mathbf{p})$ and $\mathbf{p}_1 = (1, \mathbf{p})$ we have $f_i(\mathbf{p}_j) = 1$. Therefore $\mathbf{p} \notin \pi_0(Z(I(F)))$, and so we must have $Z(\mathrm{Co}(F, x_0)) \supseteq \pi_0(Z(I(F)))$. □

By the Lemmas 6 and 9, it follows that the coefficient constraint ideal lies in the elimination ideal. We can now use this ideal to describe the full elimination ideal, which turns out to be generated precisely by $\mathrm{Res}(F, x_0)$ and $\mathrm{Co}(F, x_0)$.

Theorem 10 *Let $I(F) = (f_1, \ldots, f_m) \subseteq \mathbb{B}[x_0, \ldots, x_{n-1}]$ be an ideal generated by a set F of Boolean polynomials. Then*

$$I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}] = I(Res(F, x_0), Co(F, x_0)).$$

Proof By Lemma 6 we have

$$\pi_0(Z(I(F))) = Z(I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}]).$$

We know that

$$I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}] \supseteq I(\mathrm{Res}(F, x_0), \mathrm{Co}(F, x_0)),$$

which implies that

$$\pi_0(Z(I(F))) = Z(I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}]) \subseteq Z(\mathrm{Res}(F, x_0)) \cap Z(\mathrm{Co}(F, x_0)).$$

Conversely, let a point $\mathbf{p} \in Z(\mathrm{Res}(F, x_0)) \cap Z(\mathrm{Co}(F, x_0))$ be given. Then $\mathbf{p}$ has two liftings to points in $\mathbb{F}_2^n$: $\mathbf{p}_0 = (0, \mathbf{p})$ and $\mathbf{p}_1 = (1, \mathbf{p})$. We will show that at least one of $\mathbf{p}_0$ or $\mathbf{p}_1$ is contained in $Z(I(F))$. Let $f_i = x_0 a_i + b_i$ be an element in F. Since $\mathbf{p}$ vanishes on $\mathrm{Co}(F, x_0)$, the following are the possible values for the terms in f_i.

$a_i(\mathbf{p})$	$b_i(\mathbf{p})$
0	0
1	0
1	1

Note that $\mathrm{Co}(F, x_0)$ excludes $(a_i(\mathbf{p}), b_i(\mathbf{p}))$ from taking the value $(0, 1)$. Since $\mathbf{p}$ also vanishes on the resultant ideal, there cannot be two f_i and f_j such that $(a_i(\mathbf{p}), b_i(\mathbf{p}))$ takes the value $(1, 0)$ and $(a_j(\mathbf{p}), b_j(\mathbf{p}))$ takes the value $(1, 1)$, since in that case the resultant $a_i b_j + a_j b_i$ would not vanish. This means that either 1) : $\{(a_i(\mathbf{p}), b_i(\mathbf{p}))|1 \leq i \leq m\} \subseteq \{(0, 0), (1, 0)\}$, or 2) : $\{(a_i(\mathbf{p}), b_i(\mathbf{p}))|1 \leq i \leq m\} \subseteq \{(0, 0), (1, 1)\}$. In case 1), the lifting $\mathbf{p}_0$ is in the zero set $Z(I(F))$. In case 2) the lifting $\mathbf{p}_1$ is in the zero set of $Z(I(F))$. This shows that $Z(\mathrm{Res}(F, x_0)) \cap Z(\mathrm{Co}(F, x_0))$ lifts to $Z(I(F))$, which means that

$$\pi_0(Z(I(F))) = Z(I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}]) \supseteq Z(\mathrm{Res}(F, x_0)) \cap Z(\mathrm{Co}(F, , x_0))$$

as desired. □

In general, this process can be iterated eliminating more variables. We denote by $\mathrm{Res}(F, x_0, \ldots, x_{k-1})$ and $Co(F, x_0, \ldots, x_{k-1})$ the iterative application of the resultant and the coefficient constraint ideal with respect to a sequence $x_0, \ldots, x_{k-1}$ of variables to be eliminated. Note that both Lemma 9 and Proposition 10 easily generalize to this case. We can also generalize Theorem 10 as follows.

Corollary 11 *For $I(F) = (f_1, \ldots, f_m)$ in $\mathbb{B}[x_0, \ldots, x_{n-1}]$, then*

$$I(F) \cap \mathbb{B}[x_k, \ldots, x_{n-1}] = I(Res(F, x_0, \ldots, x_{k-1}), Co(F, x_0, \ldots, x_{k-1})).$$

This enables us to actually compute the elimination ideal *independent of monomial order*, in contrast to approaches using Gröbner bases (see [2,3]). Moreover, one could find the elimination ideal by successively eliminating $x_0, \ldots, x_{k-1}$ using Corollary 11 by the following algorithm:

0. $F_0 = F$,
1. $F_1 =$ generators of $\mathrm{Res}(F_0, x_0) + Co(F_0, x_0)$,
2. $F_2 =$ generators of $\mathrm{Res}(F_1, x_1) + Co(F_1, x_1)$,
   ...
i. $F_i =$ generators of $\mathrm{Res}(F_{i-1}, x_{i-1}) + Co(F_{i-1}, x_{i-1})$,
   ...

We show that this simple solving algorithm already gives a complexity that is better than the trivial $\mathcal{O}(2^n)$.

Lemma 12 *If the degree of the polynomials in F_0 is 2, then the maximal degree of the polynomials in F_i is at most $2^i + 1$.*

Proof We show this by induction. The statement is clearly true for $i = 0$. Assume the statement is true for $i = j$, so the maximal degree of the polynomials in F_j is $2^j + 1$. The generators for both $\mathrm{Res}(F_j, x_j)$ and $Co(F_j, x_j)$ are computed by multiplying two polynomials, where the first has degree at most 2^j (because x_j has been factored out), and the other has degree at most $2^j + 1$. The generators of $Res(F_j, x_j)$ and $Co(F_j, x_j)$ then get maximal degree $2 \cdot 2^j + 1 = 2^{j+1} + 1$, as desired. □

Theorem 13 *The maximal degree of any polynomial in any F_i when running the elimination algorithm above is upper bounded by $n - \log_2(n) + 1$ and F_0 only has quadratic polynomials.*

Proof Let D be the maximal degree encountered during the elimination algorithm. From Lemma 12 we know that $D = max_{i:0 \le i \le n}(min\{2^i + 1, n - i\})$. For small i, $2^i + 1 < n - i$, but the function $2^i + 1$ is increasing and the function $n - i$ is decreasing when i increases, so there will be an integer i_0 such that $2^{i_0} + 1 \le n - i_0$ and $2^{i_0+1} + 1 > n - i_0 - 1$. Then $D = \max\{2^{i_0} + 1, n - i_0 - 1\}$. It is easy to see that $2^{\log_2(n)-1} + 1 = n/2 + 1 \le n - \log_2(n) + 1$ and $2^{\log_2(n)} + 1 = n + 1 > n - \log_2(n)$ when $n \ge 1$, so $i_0 = \log_2(n) - 1$. The theorem then follows since $D \le \max\{2^{\log_2(n)-1} + 1, n - \log_2(n) + 1\} = n - \log_2(n) + 1$. □

In [15, p. 315] it is stated that the maximal degree of polynomials occurring in the computation of Gröbner bases over $\mathbb{B}[x_0, \ldots, x_{n-1}]$ is n. Theorem 13 improves on this, in the sense that a solving algorithm using elimination of variables only needs to consider polynomials of degree up to $n - \log_2(n) + 1$.

Applying the straightforward elimination of variables in practice leads to problems, however, due to the initial exponential growth of degrees from the resultants and coefficient constraints with each elimination. With many variables, the number of monomials quickly becomes too large for a computer to work with, which is why we suggest in this paper to only compute polynomials of degree $\le d$, where d is a free parameter. Note that Corollary 11 is only valid if we do not restrict the maximum degree allowed.

4.4 Syzygies

The last concept we will define is that of *syzygies*. In fact, syzygies will give a more general view on resultants and coefficient constraints. The reason for this extension is that for degree d, in general both $\deg(\mathrm{Res}(F^d, F^d, x_0)) = 2d - 1$ and $\deg(\mathrm{Co}(F^d, x_0)) = 2d - 1$. This means that computation of the generators for $I(\cup_{i=2}^{d} F^i) \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$ produces polynomials of degree $\le 2d - 1$. When restricting the maximum allowed degree, resultants and coefficient constraints are not sufficient to generate the full elimination ideal. nontrivial syzygies fill this gap.

Definition 14 *Let $a_1, \ldots, a_\ell$ be boolean polynomials in $\mathbb{B} = \mathbb{B}[x_0, \ldots, x_{n-1}]$. Let*

$$\mathbb{B}^\ell = \mathbb{B}\epsilon_1 \oplus \cdots \oplus \mathbb{B}\epsilon_\ell$$

be the free $\mathbb{B}$-module of rank ℓ, where $\epsilon_1, \ldots, \epsilon_\ell$ is an (abstract) basis. The syzygy module S for polynomials a_i is the submodule of $\mathbb{B}^\ell$ consisting of all $r_1\epsilon_1 + \cdots + r_\ell\epsilon_\ell \in \mathbb{B}^\ell$ which form a relation

$$r_1 a_1 + \cdots + r_\ell a_\ell = 0.$$

We may also specify natural numbers $d_1, \ldots d_\ell$ such that $\deg a_i \leq d_i$. We let the module $\mathbb{B}^\ell$ be graded by letting ϵ_i have degree d_i. Then the syzygies of degree $\leq d$, $S^{\leq d}$ consists of the syzygies $\sum_i r_i\epsilon_i$ such that $\deg(r_i) + d_i \leq d$.

Syzygies are connected to Gröbner bases in the literature concerning optimizations of Buchberger's algorithm. In fact, most known approaches to Gröbner bases (for example [8,9]), reduce to computing the module of syzygies over the polynomial ring $K[x_0, \ldots, x_{n-1}]$ for some field K.

In [10, Ch.3] it is shown that whenever we get a reduction to 0 for an S-polynomial in Buchberger's algorithm, this reduction corresponds to a syzygy. One way to find the syzygies of some polynomials $(f_1, \ldots, f_m)$ is therefore to save the ones encountered when reducing S-polynomials in Buchberger's algorithm for computing Gröbner bases. In our work we also consider syzygies, but we generate a basis for them directly, without going through reductions of S-polynomials. Note that it is known that one can compute the elimination ideal by using Gröbner bases ([2,3]). Our approach is related but differs from the Gröbner bases approaches considered in the literature: We attempt to find more efficient ways of computing the vector space

$$\langle F^d \cup L^1F^{d-1} \cup L^2F^{d-2} \cup \cdots \cup L^{d-2}F^2 \rangle \cap \mathbb{B}[x_0, \ldots, x_{n-1}].$$

Let us indicate how syzygies enter here. Given a set of polynomials

$$F : x_0a_1 + b_1, x_0a_2 + b_2, \ldots, x_0a_m + b_m,$$

let V be the linear space of all expressions

$$l_1(x_0a_1 + b_1) + \cdots + l_m(x_0a_m + b_m)$$

with certain degree restrictions $\deg(l_i) \leq d_i$ or equivalently $l_i \in \langle L^{d_i} \rangle$. We want to eliminate x_0 and compute the space $V \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$. By a suitable reduction process we shall obtain a set of polynomials

$$F' : x_0a'_1 + b'_1, x_0a'_2 + b'_2, \ldots, x_0a'_k + b'_k.$$

Let V' be the linear space of all expressions

$$l'_1(x_0a'_1 + b'_1) + \cdots + l'_l(x_0a'_k + b'_k) \tag{2}$$

where l'_i now is a polynomial in $\langle L^{\frac{e_i}{x_0}} \rangle$ (for appropriate e_i). The essential step in computing the elimination space $V \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$ turns out to be to

compute $V' \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$. But we see that if an element (2) is in the latter intersection, we must have

$$l'_1 a'_1 + \cdots + l'_k a'_k = 0,$$

so $l'_1\epsilon_1 + \cdots + l'_k\epsilon_k$ is a syzygy for the a'_i. For each such syzygy we get by (2) a corresponding element in the intersection $V' \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$:

$$l'_1 b'_1 + \cdots + l'_k b'_k$$

and this intersection is the space of all such elements as we vary over the syzygies of the a'_i.

The following illustrates how the approach in this paper differs from Gröbner bases.

1. We compute the syzygies directly in the Boolean ring $\mathbb{B}[x_1, \ldots, x_{n-1}]$, which means that the field equations are encoded into our computations in contrary to Gröbner bases which are computed over the polynomial ring $\mathbb{F}_2[x_1, \ldots, x_{n-1}]$, thus needing the field equations to be added to the system of equations.
2. In the approach of this paper we only need to compute syzygies on the a_j^{i-1} terms. These *have degree one less* than the f^i's.
3. The approach here avoids the chain of reductions as done in Buchberger's algorithm since syzygies are computed directly. The trivial syzygies are Koszul syzygies $a_l^{j-1}\epsilon_k^{i-1} - a_k^{i-1}\epsilon_l^{j-1}$, and Boolean syzygies$(a_k^{i-1} + 1)\epsilon_k^{i-1}$ and any multiples of these allowed by the degree restriction d.
4. The approach is "straight to the point" meaning that we eliminate variables in a straightforward way avoiding precise term orderings apart from the elimination order $x_0 > x_1 > \ldots$, which can be arbitrary.

The Boolean syzygies are unique because they only occur for characteristic 2. In the following, we show that these two types of trivial syzygies give the resultants and coefficient constraints when applied to the f^i's.

4.5 Syzygies between linear polynomials

Let $a_1, \cdots, a_\ell$ be Boolean polynomials in $\langle L_{\overline{x_0}}^{\leq 1} \rangle$. After suitable Gaussian elimination we may assume that they are ordered so that the intial terms (in this case a variable or simply the constant 1)

$$\mathrm{in}(a_1) > \mathrm{in}(a_2) > \cdots > \mathrm{in}(a_\ell). \tag{3}$$

Proposition 15 *Let $S \subseteq \mathbb{B}_{\epsilon_1} \oplus \cdots \oplus \mathbb{B}_{\epsilon_\ell}$ be the syzygy module for $a_1, \ldots, a_\ell$, where each ϵ_i has degree 1. There are the following syzygies in S:*

1. *Koszul syzygies $a_j\epsilon_i + a_i\epsilon_j$,*
2. *Boolean syzygies $(a_i + 1)\epsilon_i$.*

Let K (resp. B) be the linear spaces generated by the Koszul (resp. Boolean) syzygies, and let $d \geq 2$. Then $S^{\leq d}$ is $\langle L_{x_0}^{d-2} \rangle K + \langle L_{x_0}^{d-2} \rangle B$.

Proof Let $\mathbf{r} = \sum_i r_i \epsilon_i$ be a syzygy of degree $\leq d$ so each $\deg(r_i) \leq d-1$. Suppose a term in r_i is $t \cdot \mathrm{in}(a_j)$ where $i \leq j$ and $\deg(t) + 1 \leq d-1$ (so $t \in \langle L_{x_0}^{d-2} \rangle$). If $i < j$ we subtract $t \cdot (a_i \epsilon_j + a_j \epsilon_i)$ from $\mathbf{r}$, and if $i = j$ we subtract $(a_i + 1)\epsilon_i$. Continuing this way we get a syzygy $\mathbf{r}' = \sum_i r_i' \epsilon_i$ such that r_i' contains no term $\mathrm{in}(a_j)$ where $j \leq i$. We show that $\mathbf{r}' = 0$. This will prove the proposition. So we have the relation $\sum_i r_i' a_i = 0$. Let $\mathrm{in}(a_1) = x_1$, say. Then no a_j for $j \geq 2$ contains the variable x_1 by the assumption (3). But also no r_j' contains the variable x_1 by construction. Hence the only terms in the relation above that contains x_1 are the terms in $r_1' x_1$. Hence we must have $r_1' = 0$. In this manner we may continue and we get that all $r_i' = 0$. □

When considering a system of independent quadratic polynomials $F^2 = \{f_1, \ldots, f_m\}$ where each $f_i = x_0 a_i + b_i$ or $f_i = b_i$ (if f does not contain x_0), it follows that the a_i are polynomials of degree ≤ 1. By Proposition 15 we know that the syzygy module is generated by Koszul and Boolean syzygies, which implies that if we map ϵ_i to b_i and ϵ_j to b_j in the Koszul syzygies we generate exactly the resultant $\mathrm{Res}(f_i, f_j, x_0) = a_j b_i + a_i b_j$. Similarly, mapping ϵ_i to b_i in the Boolean syzygies, we generate exactly the Coefficient constraint $\mathrm{Co}(f_i, x_0) = (a_i + 1) b_i$. This implies the following result.

Corollary 16 *Let $x_0 a_i + b_i$ for $i = 1, \ldots, m$ be quadratic polynomials in $\mathbb{B}[x_0, \ldots, x_{n-1}]$. The linear span of the resultants and coefficient constraints in $\mathbb{B}[x_1, \ldots, x_{n-1}]$ is precisely the image of the composition*

$$K + B \subseteq (\mathbb{B}[x_1, \ldots, x_{n-1}]^m)^2 \xrightarrow{\phi} \langle L_{x_0}^3 \rangle,$$

where ϕ sends $\epsilon_i \mapsto b_i$.

Note how this relates to computing the intersection

$$\langle L_{x_0}^{\leq 1} F_{x_0}^2 \rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}].$$

By Proposition 15 the above is equivalent to find all solutions $(l_1, \ldots, l_m)$, where $l_i \in \langle L_{x_0}^{\leq 1} \rangle$ that satisfy $l_1 a_1 + l_2 a_2 + \cdots + l_{r_2} a_{r_2} = 0$. This is the linear span of the Koszul and Boolean syzygies. Corollary 16 above gives that the intersection is generated by the resultants and the coefficient constraints.

Syzygies between polynomials of degree ≥ 2: When the degree of some a_i's are greater than 1, there may be other syzygies that are not generated by the Koszul and Boolean syzygies. We discuss these nontrivial syzygies in the Appendix.

5 Elimination of variables from systems of Boolean equations

Previous work on solving Boolean equation systems in ANF form include the XL and XSL algorithms [6,7]. In those approaches, one multiplies all equations with all monomials up to some fixed degree. If the degree is large enough, one could hope to have more equations than monomials in the system, and hence solve the system by re-linearization.

One can also use the approach of multiplying all polynomials with all monomials up to some degree to eliminate the variable x_0. Indeed, after generating the Macaulay matrix of the full set of polynomials one can perform Gaussian elimination on terms containing x_0. If we have more independent polynomials than x_0-terms we are certain to end up with some equations with no terms depending on x_0. This procedure can be iterated to eliminate a sequence of variables $x_0, x_1, \ldots$ just by using Gaussian elimination on the set of polynomials after multiplying with all allowed monomials.

When we have polynomials of low degree in F, the complexity of multiplying all of them with all monomials of allowed degree quickly becomes very large when d increases. We aim to eliminate x_0 for the sets $F^d, F^{d-1}, \ldots, F^2$ in a more efficient way motivated by the tools developed in the previous sections.

5.1 Bounding the degree to $d = 3$

The lowest degree possible to make meaningful use of the elimination techniques is to set the degree bound at $d = 3$. Here the input polynomials are cubic and quadratic, which means that we consider the sets F^3 and F^2. In the following procedure, these sets will be modified to only include polynomials respecting the degree constraint $d \leq 3$ while eliminating the variable x_0. The **elimination procedure** proceeds as follows:

1. We start by splitting the set F^2 into subsets $F^{2,norm}_{x_0}$ and $F^2_{\overline{x_0}}$ using the procedure $SplitVariable(F^2, x_0)$. We increase F^3, by adding $x_0 F^2_{\overline{x_0}}$ and $(x_0+1)F^{2,norm}_{x_0}$ to F^3. Note that we only multiply the sets $F^{2,norm}_{x_0}$ and $F^2_{\overline{x_0}}$ with (x_0+1), resp. x_0 and not with all linear polynomials.
2. With the new sets, we normalize F^3 with $F^{2,norm}_{x_0}$ as basis, producing $F^{3,norm}_{x_0}$ and $F^3_{\overline{x_0}}$. Since all initial terms in $F^{2,norm}_{x_0}$ depend on x_0, all terms removed in the normalization process will depend on x_0. Hence we get rid of many terms containing x_0 in this step. The Gaussian elimination in $SplitVariable$ allows us, however, to perform the reductions with these "all at once" in analogy with the F4 algorithm, [8]. The normalization process removes a lot of the degree ≤ 3 monomials depending on x_0 in $F^3_{x_0}$ and may eliminate x_0 completely from some polynomials.
3. The final step is to compute resultants and coefficient constraints from the set $F^{2,norm}_{x_0}$. We create $Res(F^{2,norm}_{x_0}, x_0)$ and $Co(F^{2,norm}_{x_0}, x_0)$ and join these sets with $F^3_{\overline{x_0}}$.

The outputs of the procedure are $F^3_{\overline{x_0}}$ and $F^2_{\overline{x_0}}$, sets of cubic and quadratic polynomials that do not depend on x_0. The following theorem shows that by

following this procedure we have not lost anything essential, in the sense that all polynomials in $F^2_{\overline{x_0}}$ multiplied with $L_{\overline{x_0}} = \{x_1, \ldots, x_{n-1}\}$ together with $F^3_{\overline{x_0}}$ still generate the whole $\langle F^3 \cup LF^2\rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$.

Theorem 17 *Let F^3 and F^2 be the input sets of polynomials, and $F^3_{\overline{x_0}}$ and $F^2_{\overline{x_0}}$ be the outputs of the elimination procedure.*

a. The linear span

$$\langle F^3 \cup L^{\leq 1}F^2\rangle = \langle F^{3,norm}_{x_0} \cup L^{\leq 1}_{\overline{x_0}} F^{2,norm}_{x_0} \cup F^3_{\overline{x_0}} \cup L^{\leq 1}_{\overline{x_0}} F^2_{\overline{x_0}}\rangle.$$

b. The elimination space

$$\langle F^3 \cup L^{\leq 1}F^2\rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}] = \langle F^3_{\overline{x_0}} \cup L^{\leq 1}_{\overline{x_0}} F^2_{\overline{x_0}}\rangle.$$

Proof a. It is clear that we have the inclusion $\supseteq$. Let us now prove that we have inclusion $\subseteq$. First note that we have the following decompositions

$$\langle F^2\rangle = \langle F^{2,norm}_{x_0} \cup F^2_{\overline{x_0}}\rangle$$

and

$$\langle L^{\leq 1}\rangle = \langle x_0\rangle + \langle L^{\leq 1}_{\overline{x_0}}\rangle$$

This gives

$$\begin{aligned}\langle L^{\leq 1}F^2\rangle &= \langle L^{\leq 1}F^{2,norm}_{x_0}\rangle + \langle L^{\leq 1}F^2_{\overline{x_0}}\rangle \\ &= (x_0+1)\langle F^{2,norm}_{x_0}\rangle + \langle L^{\leq 1}_{\overline{x_0}} F^{2,norm}_{x_0}\rangle + x_0\langle F^2_{\overline{x_0}}\rangle + \langle L^{\leq 1}_{\overline{x_0}} F^2_{\overline{x_0}}\rangle. \quad (4)\end{aligned}$$

Now by the normalization procedure in Part 2. above:

$$\langle F^3\rangle + (x_0+1)\langle F^{2,norm}_{x_0}\rangle + x_0\langle F^2_{\overline{x_0}}\rangle \subseteq \langle F^{3,norm}_{x_0} \cup F^3_{\overline{x_0}} \cup L^{\leq 1}_{\overline{x_0}} F^{2,norm}_{x_0}\rangle. \quad (5)$$

Hence putting (4) and (5) together we obtain

$$\langle F^3\rangle + \langle L^{\leq 1}F^2\rangle \subseteq \langle F^{3,norm}_{x_0}\rangle + \langle L^{\leq 1}_{\overline{x_0}} F^{2,norm}_{x_0}\rangle + \langle F^3_{\overline{x_0}}\rangle + \langle L^{\leq 1}_{\overline{x_0}} F^2_{\overline{x_0}}\rangle,$$

which gives part a.

b. By the identity in a. we see that

$$\begin{aligned}&\langle F^3 \cup L^{\leq 1}F^2\rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}] \quad (6)\\ &= \langle F^3_{\overline{x_0}} \cup L^{\leq 1}_{\overline{x_0}} F^2_{\overline{x_0}}\rangle + \langle F^{3,norm}_{x_0} \cup L^{\leq 1}_{\overline{x_0}} F^{2,norm}_{x_0}\rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}].\end{aligned}$$

Let $x_0a^1_i + b^2_i$ be the elements of $F^{2,norm}_{x_0}$ and $x_0a^2_i + b^3_i$ be the elements of $F^{3,norm}_{x_0}$. Then any element of the intersection on the right side of (6), comes from an expression

$$\sum_i l_i(x_0a^1_i + b^2_i) + \sum_i c_i(x_0a^2_i + b^3_i),$$

where the l_i are linear polynomials in $\mathbb{B}[x_1, \ldots, x_{n-1}]$ and the c_i constants. We must have the relation

$$\sum_i l_ia^1_i + \sum_i c_ia^2_i = 0.$$

But due to the elements of $F_{x_0}^{3,norm}$ being normalized with respect to $F_{x_0}^{2,norm}$, for any such relation we have each $c_i = 0$ and the syzygy $\sum_i l_i \epsilon_i^1$ is a linear combination of Koszul and Boolean syzygies. By Corollary 16 the last intersection in (6) is then a linear combination of resultants and coefficient constraints, and these have already been put into $F_{\overline{x_0}}^2$. This proves part b. □

Example: The following small example illustrates how the elimination procedure works. Let F^2 be the initial system with quadratic equations in the five variables x_0, x_1, x_2, x_3, x_4:

$$F^2 = \left\{ \begin{array}{l} x_2x_4 + x_1x_4 + x_0x_3 + x_0x_1 + x_4 + x_0 \\ x_2x_4 + x_0x_3 + x_1x_2 + x_0x_2 + x_2 + 1 \\ x_1x_4 + x_0x_4 + x_3 + x_2 \\ x_3x_4 + x_0x_2 + x_0x_1 + x_4 + x_3 + x_0 \end{array} \right\}$$

We will eliminate x_0 from F^2, by allowing the degree of polynomials to increase by one. First we split the set into $F_{x_0}^{2,norm}$ and $F_{\overline{x_0}}^2$, by doing Gauss elimination on terms depending on x_0. We then get

$$\begin{aligned} F_{x_0}^{2,norm} &= \left\{ \begin{array}{l} x_1x_4 + x_0x_4 + x_3 + x_2 \\ x_2x_4 + x_1x_4 + x_0x_3 + x_0x_1 + x_4 + x_0 \\ x_1x_4 + x_1x_2 + x_0x_2 + x_0x_1 + x_4 + x_2 + x_0 + 1 \end{array} \right\} \\ F_{\overline{x_0}}^2 &= \{x_3x_4 + x_1x_4 + x_1x_2 + x_3 + x_2 + 1\} \end{aligned}$$

The next step is to compute $(x_0+1)F_{x_0}^{2,norm}$ and $x_0F_{\overline{x_0}}^2$ and add to the complete F^2-set. We then get

$$F^2 \cup x_0F_{\overline{x_0}}^2 \cup (x_0+1)F_{x_0}^{2,norm} = \left\{ \begin{array}{l} x_0x_1x_4 + x_1x_4 + x_0x_3 + x_0x_2 + x_3 + x_2 \\ x_0x_2x_4 + x_0x_1x_4 + x_2x_4 + x_1x_4 + x_0x_4 + x_4 \\ x_0x_1x_4 + x_0x_1x_2 + x_1x_4 + x_0x_4 + x_1x_2 + x_0x_2 + x_4 + x_2 + x_0 + 1 \\ x_0x_3x_4 + x_0x_1x_4 + x_0x_1x_2 + x_0x_3 + x_0x_2 + x_0 \\ x_1x_4 + x_0x_4 + x_3 + x_2 \\ x_2x_4 + x_1x_4 + x_0x_3 + x_0x_1 + x_4 + x_0 \\ x_1x_4 + x_1x_2 + x_0x_2 + x_0x_1 + x_4 + x_2 + x_0 + 1 \\ x_3x_4 + x_1x_4 + x_1x_2 + x_3 + x_2 + 1 \end{array} \right\}$$

Normalizing this set with respect to $F_{x_0}^{2,norm}$ gives us $F_{x_0}^{3,norm}$ and $F_{\overline{x_0}}^3$:

$$\begin{aligned} F_{x_0}^{3,norm} &= \left\{ \begin{array}{l} x_1x_4 + x_0x_4 + x_3 + x_2 \\ x_2x_4 + x_1x_4 + x_0x_3 + x_0x_1 + x_4 + x_0 \\ x_1x_4 + x_1x_2 + x_0x_2 + x_0x_1 + x_4 + x_2 + x_0 + 1 \\ x_2x_4 + x_1x_2 + x_0x_1 + x_2 + x_1 + 1 \\ x_1x_2 + x_4 + x_3 + x_1 + x_0 + 1 \end{array} \right\} \\ F_{\overline{x_0}}^3 &= \left\{ \begin{array}{l} x_1x_2x_4 + x_4 \\ x_2x_4 + x_1x_3 + x_3 + 1 \\ x_3x_4 + x_1x_4 + x_1x_2 + x_3 + x_2 + 1 \end{array} \right\} \end{aligned}$$

The next step is to compute the resultants and coefficient constraints of the set $F_{x_0}^{2,norm}$. In this example we have three polynomials in $F_{x_0}^{2,norm}$ so we will get three polynomials in each of $Res(F_{x_0}^{2,norm}, x_0)$ and $Co(F_{x_0}^{2,norm}, x_0)$ These sets are

$$Res(F_{x_0}^{2,norm}, x_0) = \left\{ \begin{array}{l} x_1x_3x_4 + x_2x_4 + x_1x_4 + x_2x_3 + x_1x_3 + x_1x_2 + x_4 + x_2 \\ x_1x_2x_3 + x_3x_4 + x_1x_4 + x_1x_3 + x_4 + x_3 + x_1 + 1 \\ x_2x_4 + x_1x_4 + x_2x_3 + x_1x_3 + x_1x_2 + x_3 \end{array} \right\}$$

$$Co(F_{x_0}^{2,norm}, x_0) = \left\{ \begin{array}{l} x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_4 + x_3x_4 \\ x_1x_2x_4 + x_2x_4 + x_1x_2 + x_1 \\ x_3x_4 + x_2x_4 + x_3 + x_2 \end{array} \right\}$$

Finally we join $F_{\overline{x_0}}^3, Res(F_{x_0}^{2,norm}, x_0)$ and $Co(F_{x_0}^{2,norm}, x_0)$ together, and do Gaussian elimination with respect to degree to remove any linearly dependent polynomials. In this case there were no dependencies, and the final set returned from the elimination procedure is

$$F_{\overline{x_0}}^3 \cup Res(F_{x_0}^{2,norm}, x_0) \cup Co(F_{x_0}^{2,norm}, x_0) = \left\{ \begin{array}{l} x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_4 + x_3x_4 \\ x_1x_3x_4 + x_2x_4 + x_1x_4 + x_2x_3 + x_1x_3 + x_1x_2 + x_4 + x_2 \\ x_1x_2x_4 + x_4 \\ x_1x_2x_3 + x_3x_4 + x_1x_4 + x_1x_3 + x_4 + x_3 + x_1 + 1 \\ x_3x_4 + x_1x_4 + x_1x_2 + x_3 + x_2 + 1 \\ x_2x_4 + x_1x_2 + x_4 + x_1 \\ x_1x_4 + x_4 + x_1 + 1 \\ x_2x_3 + x_1x_3 + x_3 + 1 \\ x_1x_3 + x_1x_2 + x_4 + x_3 + x_1 + 1 \end{array} \right\}$$

With this small example we see that eliminating x_0 using our procedure we generate nine polynomials of degree 3, in four variables. The final set is Gaussian eliminated with respect to degree, and we also observe that we actually get five quadratic polynomials in x_1, x_2, x_3, x_4, one more polynomial than we started with, in one variable less. This gives a small taste of why this algorithm can be good for solving Boolean equation systems.

Optional addition to the algorithm.

The normalization process may produce polynomials in $F_{x_0}^{3,norm}$ of degree 2. We could test for this and if so, add this to $F_{x_0}^2$ and start the algorithm for eliminating x_0 over again. The new

$$\langle F^3 \cup LF^2 \rangle$$

will normally be larger than the original one, and so we normally get a larger elimination space than in Theorem 17b. This allows us to "compute with terms of degree 4 by only computing with terms of degree 3": Suppose we produce a polynomial of degree 2

$$h = \sum_i c_i f_i^3 + \sum_i l_i f_i^2$$

where the c_i are constants and the l_i are linear. Putting h in F^2, we can then multiply it with a linear polynomial l' to produce

$$h \cdot l' = \sum_i c_i l' f_i^3 + \sum_i l_i l' f_i^2,$$

and we see that the terms in the right expression will generally be of degree 4.

In the example above, we see that the $F_{x_0}^{3,norm}$ we produced contains *only* quadratic polynomials. Adding these to $F_{x_0}^{2,norm}$ before computing resultants and coefficient constraints yields all five x_0-terms as initial monomials, and hence $\binom{5}{2} = 10$ polynomials in $Res(F_{x_0}^{2,norm}, x_0)$:

$$Res(F_{x_0}^{2,norm}, x_0) = \left\{ \begin{array}{l} \\ x_2x_3x_4 + x_1x_2x_3 + x_2x_4 + x_2x_3 + x_1x_3 + x_1x_2 + x_3 + x_2 + x_1 + 1 \\ x_1x_3x_4 + x_1x_2x_3 + x_3x_4 + x_2x_4 + x_2x_3 + x_1x_2 + x_3 + x_2 + x_1 + 1 \\ x_1x_2x_4 + x_1x_3 + x_1x_2 + x_4 \\ x_1x_2x_3 + x_3x_4 + x_1x_4 + x_1x_3 + x_4 + x_3 + x_1 + 1 \\ x_3x_4 + x_1x_3 + x_1x_2 + x_4 + x_3 + x_2 \\ x_2x_4 + x_1x_4 + x_2x_3 + x_1x_3 + x_1x_2 + x_3 \\ x_1x_4 + x_1 \\ x_2x_3 + x_1x_2 + x_4 + x_3 \\ x_1x_3 + x_4 + x_2 + 1 \\ x_4 + x_2 + x_1 + 1 \end{array} \right\}$$

For the coefficient constraints, one of the polynomials in $F_{x_0}^{2,norm}$ is of the form $x_0 + b$, where $b \in \mathbb{B}[x_1, \ldots, x_4]$. For this particular polynomial $a = 1$, and hence the coefficient constraint $(a+1)b = 0$, so we only get four non-zero polynomials in $Co(F_{x_0}^{2,norm}, x_0)$. These are

$$Co(F_{x_0}^{2,norm}, x_0) = \left\{ \begin{array}{l} x_3x_4 + x_2x_4 + x_3 + x_2 \\ x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_4 + x_3x_4 \\ x_1x_2x_4 + x_2x_4 + x_1x_2 + x_1 \\ x_1x_2x_4 + x_2x_4 + x_1x_2 + x_2 + x_1 + 1 \end{array} \right\}$$

Joining these two sets with $F_{\overline{x_0}}^{3,norm}$ and doing Gauss reduction with respect to degree gives the following set.

$$F^3_{\overline{x_0}} \cup Res(F^{2,norm}_{x_0}, x_0) \cup Co(F^{2,norm}_{x_0}, x_0) =$$
$$\left\{\begin{array}{l} x_2x_3x_4 + x_1x_2x_3 + x_2x_4 + x_2x_3 + x_1x_3 + x_1x_2 + x_3 + x_2 + x_1 + 1 \\ x_1x_3x_4 + x_1x_2x_3 + x_3x_4 + x_2x_4 + x_2x_3 + x_1x_2 + x_3 + x_2 + x_1 + 1 \\ x_1x_2x_4 + x_4 \\ x_1x_2x_3 + x_3x_4 + x_1x_4 + x_1x_3 + x_4 + x_3 + x_1 + 1 \\ x_3x_4 + x_1x_4 + x_1x_2 + x_3 + x_2 + 1 \\ x_2x_4 + x_1x_3 + x_3 + 1 \\ x_1x_4 + x_1x_3 + x_4 + 1 \\ x_2x_3 + x_1x_3 + x_1x_2 + x_4 \\ x_1x_3 + x_1x_2 \\ x_1x_2 + x_4 + x_1 + 1 \\ x_4 + x_3 + x_1 + 1 \\ x_3 + x_1 \\ x_2 + x_1 \\ x_1 + 1 \end{array}\right\}$$

Using the additional optimization, we are able to "squeeze" out additional low-degree polynomials compared to the initial $F^{2,norm}_{x_0}$. In this example we get four independent linear polynomials after Gauss elimination in the four remaining variables, and therefore easily solve the system. The solution in the example is $x_1 = x_2 = x_3 = x_4 = 1$, and substituting these values into the initial system immediately also gives the value for x_0, which is also equal to 1.

Theorem 17 generalizes easily to eliminating several variables using the elimination procedure.

Corollary 18 *Let $L^{\leq 1}_{\overline{x_0,\ldots,x_{k-1}}} = \{1, x_k, \ldots x_{n-1}\}$ be the subset of $L^{\leq 1}$ not containing the variables $x_0, \ldots, x_{k-1}$. Let $F^2_{\overline{x_0,\ldots,x_{k-1}}}$ and $F^3_{\overline{x_0,\ldots,x_{k-1}}}$ be the result of applying the elimination procedure above k times to the input sets F^3, F^2, eliminating one variable at the time in the sequence $x_0, \ldots, x_{k-1}$. Then*

$$\langle F^3_{\overline{x_0,\ldots,x_{k-1}}} \cup L^{\leq 1}_{\overline{x_0,\ldots,x_{k-1}}} F^2_{\overline{x_0,\ldots,x_{k-1}}} \rangle = \langle F^3 \cup L^{\leq 1} F^2 \rangle \cap \mathbb{B}[x_k, \ldots, x_{n-1}].$$

Proof The output after eliminating $x_0, \ldots, x_{i-1}$ are the sets $F^3_{\overline{x_0,\ldots,x_{i-1}}}$ and $F^2_{\overline{x_0,\ldots,x_{i-1}}}$. These sets form the input for eliminating x_i, and applying Theorem 17 on these input sets gives

$$\langle F^3_{\overline{x_0,\ldots,x_i}} \cup L^{\leq 1}_{\overline{x_0,\ldots,x_i}} F^2_{\overline{x_0,\ldots,x_i}} \rangle = \langle F^3_{\overline{x_0,\ldots,x_{i-1}}} \cup L^{\leq 1}_{\overline{x_0,\ldots,x_{i-1}}} F^2_{\overline{x_0,\ldots,x_{i-1}}} \rangle \cap \mathbb{B}[x_{i+1}, \ldots, x_{n-1}],$$

for each i in $\{0, \ldots, k-1\}$. Substituting these equations into each other creates the stated relation between the original F^3, F^2 and $F^3_{\overline{x_0,\ldots,x_{k-1}}}, F^2_{\overline{x_0,\ldots,x_{k-1}}}$. □

5.2 Degree bound $d \geq 4$

We explain here how the elimination procedure needs to be generalized when we increase the degree bound to $d = 4$. The construction is essentially the same as in the previous section, but there will be more syzygies than Koszul and Boolean syzygies, as pointed out in Theorem 26 in the Appendix.

Elimination procedure:

1. We split the set F^2 into subsets $F^{2,norm}_{x_0}$ containing x_0 and $F^2_{\overline{x_0}}$ not containing x_0 by using $\mathit{SplitVariable}(F^2, x_0)$. These sets can be used to increase F^3, by adding $(x_0+1)F^{2,norm}_{x_0}$ and $x_0F^2_{\overline{x_0}}$ to F^3.
2. Normalize F^3 with respect to $F^{2,norm}_{x_0}$, producing $F^{3,norm}_{x_0}$ and $F^3_{\overline{x_0}}$.
3. Compute the resultants and coefficient constraints from $F^{2,norm}_{x_0}$ and add to $F^3_{\overline{x_0}}$
4. Add the sets $(x_0+1)F^{3,norm}_{x_0}$ and $x_0F^3_{\overline{x_0}}$ to F^4.
5. Normalize F^4 with respect to $F^{3,norm}_{x_0} \cup F^{2,norm}_{x_0}$, giving the sets $F^{4,norm}_{x_0}$ and $F^4_{\overline{x_0}}$.
6. Compute resultants and coefficient constraints of the sets $F^{3,norm}_{x_0}$ and $F^{2,norm}_{x_0}$ that respect the degree bound $d = 4$, and add these to $F^4_{\overline{x_0}}$ or $F^3_{\overline{x_0}}$, according to degree.
7. Compute the syzygies R^3 of degree 3 from $F^{4,norm}_{x_0} \cup F^{3,norm}_{x_0}$. These generate a set T^4 of polynomials of degree ≤ 4 in $\mathbb{B}[x_1, \ldots, x_{n-1}]$. Add T_4 to $F^4_{\overline{x_0}}$. This step is the main difference with respect to the case $d = 3$, and finding the nontrivial syzygies is not straightforward.

The outputs of the procedure are the sets $F^2_{\overline{x_0}}, F^3_{\overline{x_0}}$ and $F^4_{\overline{x_0}}$.

If we increase to $d = 5$ or higher, the elimination procedure follows the same lines as for $d = 3$ and $d = 4$. We normalize the $F^i_{x_0}$ using the already normalized sets of lower-degree polynomials as a basis, compute resultants and coefficient constraints, as well as any nontrivial syzygies needed for eliminating x_0. When d grows, the space for possible nontrivial syzygies increases. A topic for further research is to investigate how large fraction of all syzygies that are covered by Boolean and Koszul syzygies when d increases.

6 An alternative approach to eliminating variables from quadratic Boolean equations

In this section we develop an alternative algorithm for solving systems of Boolean equations by eliminating variables, where the degree of polynomials produced does not grow too fast. To guarantee that no false solutions are added during the elimination steps, certain conditions (stated in Theorem 20) must be satisfied.

6.1 Eliminating variables with no cross-terms

We intend to eliminate the k variables $x_0, \ldots, x_{k-1}$ from F. We shall assume that F does not contain any quadratic terms between the variables $x_0, \ldots, x_{k-1}$, i.e. no terms $x_i x_j$ where $0 \leq i < j < k$. Then any polynomial in F may be written uniquely as

$$x_0 a_0 + \cdots + x_{k-1} a_{k-1} + b$$

where each a_i and b are in $\mathbb{B}[x_k, \ldots, x_{n-1}]$.

Let

$$f_i = x_0 a_{0,i} + \cdots + x_{k-1} a_{k-1,i} + b_i, \quad i = 1, \ldots, m$$

be the polynomials of F. We have $L^{\leq 1} = \{1, x_0, \ldots, x_{n-1}\}$ and more generally, $L^{\leq k}$ denotes the set of all monomials of degree less than or equal to k. Moreover, $L^{\leq 1}_{\overline{x_0,\ldots,x_{j-1}}} = \{1, x_j, \ldots, x_{n-1}\}$. Define the following elimination sets, realized by Gauss elimination on the Macaulay matrix of the polynomials:

1. $F^{\leq k+2}_{\overline{x_0,\ldots,x_{k-1}}} = (L^{\leq k} \cdot F) \cap \mathbb{B}[x_k, \ldots, x_{n-1}]$.
2. $G^{\leq 2} = F$ and inductively for $j = 0, \ldots, k-1$ define $G^{\leq j+3} = (L^{\leq 1}_{\overline{x_0,\ldots,x_{j-1}}} \cdot G^{\leq j+2}) \cap \mathbb{B}[x_j, \ldots, x_{n-1}]$.
3. $\tilde{G}^{\leq 2} = F$ and inductively for $j = 0, \ldots, k-1$ define $\tilde{G}^{\leq j+3} = (L^{\leq 1}_{\overline{x_0,\ldots,x_{k-1}}} \cdot \tilde{G}^{\leq j+2}) \cap \mathbb{B}[x_j, \ldots, x_{n-1}]$.

Let A be the $m \times k$-matrix with entries $a_{i,j}$ for $j = 0, \ldots, k-1$ and $i = 0, \ldots, m-1$. The column vectors of A will be denoted by $\mathbf{a}_j$, for $j = 0, \ldots, k-1$. Let $\mathbf{b}$ be the $m \times 1$ column vector consisting of the b_i's, and denote by $A|\mathbf{b}$ the concatenation of A and $\mathbf{b}$.

4. Let $\mathbf{c}$ be a binary $1 \times m$ vector with Hamming weight (= number of 1's) $w_H(\mathbf{c}) = w$ (for some w), and let

$$(\sigma_0, \sigma_1, \ldots, \sigma_{k-1}, \tau) = \mathbf{c} \cdot A|\mathbf{b}. \tag{7}$$

This is a sum of w of the rows of the matrix $A|\mathbf{b}$. Now define $H^{\leq k+2}$ to be the set of polynomials consisting of all products

$$(\sigma_0 + 1)(\sigma_1 + 1) \cdots (\sigma_{k-1} + 1)\tau$$

as we range over all vectors $\mathbf{c}$ with $w_H(\mathbf{c}) \leq k + 1$. Note that the above product is nonzero at a point p iff

$$\sigma_0(p) = \cdots = \sigma_{k-1}(p) = 0, \quad \tau(p) = 1, \tag{8}$$

Example 19 *Suppose $k = 1$. Then F consists of polynomials $x_0 a_{0,i} + b_i$, and $\langle H^{\leq 3} \rangle$ is generated by the following polynomials*

1. $(a_{0,i} + 1)b_i, \quad 1 \leq i \leq m,$
2. $(a_{0,i} + a_{0,j} + 1)(b_i + b_j), \quad 1 \leq i < j \leq m.$

From this we see easily that $\langle H^{\leq 3} \rangle$ is equivalently generated by:

1. $(a_{0,i}+1)b_i, \quad 1 \le i \le m,$
2. $a_{0,i}b_j + a_{0,j}b_i, \quad 1 \le i < j \le m.$

We note that 1. are the coefficient constraints and 2. are the resultants.

Theorem 20 *a. We have*

$$\langle F^{\le k+2}_{\overline{x_0},\ldots,\overline{x_{k-1}}} \rangle \supseteq \langle G^{\le k+2} \rangle \supseteq \langle \tilde{G}^{\le k+2} \rangle \supseteq \langle H^{\le k+2} \rangle.$$

b. The zero sets of the above sets of polynomials are all equal, or equivalently $Z(F^{\le k+2}_{\overline{x_0},\ldots,\overline{x_{k-1}}}) = Z(H^{\le k+2})$.

c. The projection $\pi_{k-1}(Z(F)) = Z(F^{\le k+2}_{\overline{x_0},\ldots,\overline{x_{k-1}}})$.

Proof a) Suppose by induction that $L^{\le j}F \supseteq F^{\le j+2}_{\overline{x_0},\ldots,\overline{x_{j-1}}} \supseteq G^{\le j+2}$ for some $0 \le j < k$. Then

$$L^{\le j+1}F \supseteq L^{\le 1}G^{j+2} \supseteq L^{\le 1}_{\overline{x_0},\ldots,\overline{x_{j-1}}} G^{\le j+2}$$

which implies $F^{\le j+2}_{\overline{x_0},\ldots,\overline{x_{j-1}}} \supseteq G^{\le j+2}$.

If $G^{\le j+2} \supseteq \tilde{G}^{\le j+2}$ then

$$L^{\le 1}_{\overline{x_0},\ldots,\overline{x_{j-1}}} G^{\le j+2} \supseteq L_{\overline{x_0},\ldots,\overline{x_{k-1}}} \tilde{G}^{\le j+2}$$

and so $G^{\le j+3} \supseteq \tilde{G}^{\le j+3}$.

Let $(\sigma_0, \sigma_1, \ldots, \sigma_{k-1}, \tau)$ be given by $\mathbf{c} \cdot A|\mathbf{b}$ as in (7). Then $f = x_0\sigma_0 + x_1\sigma_1 + \cdots + x_{k-1}\sigma_{k-1} + \tau$ is in $\langle F \rangle$, where we recall that $F = \tilde{G}^{\le 2} = H^{\le 2}$. Furthermore

$$(\sigma_0+1)(\sigma_1+1)\cdots(\sigma_{k-1}+1)\tau = (\sigma_0+1)(\sigma_1+1)\cdots(\sigma_{k-1}+1)f \in \tilde{G}^{\le k+2}.$$

As we vary over $\mathbf{c}$ with $w_H(\mathbf{c}) \le k+1$, we get all of $H^{\le k+2}$. We then see that $H^{\le k+2} \subseteq \tilde{G}^{\le k+2}$.

b) and c): We now prove that $\pi_{k-1}(Z(F)) = Z(H^{\le k+2})$. First, from a), we have that

$$\pi_{k-1}(Z(F)) \subseteq Z(F^{\le k+2}_{\overline{x_0},\ldots,\overline{x_{k-1}}}) \subseteq Z(H^{\le k+2}).$$

Given a point p in the zero set of $H^{\le k+2}$, we wish to show that it lifts to a point $\tilde{p}$ in $Z(F)$ such that $\pi_{k-1}(\tilde{p}) = p$. Denote by $A(p)$ the matrix A evaluated at p and let r be the rank of $A(p)$. Choose an $m \times r$-submatrix A' of A such that the rank of $A'(p)$ is equal to the rank of $A(p) = r \le k$.

Note. If $\mathbf{c}$ is an $1 \times m$-matrix with $\mathbf{c} \cdot A'(p) = 0$, then $\mathbf{c} \cdot A(p) = 0$, since each column of $A(p)$ is a linear combination of columns in $A'(p)$.

Now extend A to the $m \times (k+1)$-matrix $A|\mathbf{b}$ by concatenating the $m \times 1$ column vector $\mathbf{b}$.

Claim. The column vector $\mathbf{b}(p)$ is a linear combination of the column vectors in $A(p)$.

This claim will prove the statement in part c) since then the system of equations

$$x_0 a_{0,i}(p) + x_1 a_{1,i}(p) + \cdots + x_{k-1} a_{k-1,i}(p) + b_i(p) = 0, \quad 0 \leq i < m$$

has a solution for the unknowns $x_0, \ldots, x_{k-1}$, thus giving a lifting $\tilde{p}$ of p.

It is sufficient to prove that $\mathbf{b}(p)$ is a linear combination of the column vectors of $A'(p)$. We finish the proof by contradiction. So suppose $A'|\mathbf{b}(p)$ has rank $r+1$. Then some $(r+1)\times(r+1)$-minor is nonzero, say the one consisting of the top $(r+1)$ rows. So the determinant

$$\begin{vmatrix} a_{0,1}(p) & \cdots & a_{r-1,1}(p) & b_1(p) \\ a_{0,2}(p) & \cdots & a_{r-1,2}(p) & b_2(p) \\ \vdots & & \vdots & \\ a_{0,r+1}(p) & \cdots & a_{r-1,r+1}(p) & b_{r+1}(p) \end{vmatrix} \neq 0.$$

We now expand this along the last column. Let B_i be the cofactor of $b_i(p)$. Then

$$b_1(p)B_1 + b_2(p)B_2 + \cdots + b_{r+1}(p)B_{r+1} \neq 0, \text{ and so is } 1$$

while

$$a_{j,1}(p)B_1 + \cdots + a_{j,r+1}(p)B_{r+1} = 0,$$

for every $j = 0, \ldots, r-1$. The B_i's are either 0 or 1. Let $\mathbf{c}$ be the $1 \times m$ vector $\mathbf{c} = (B_1, \ldots, B_{r+1}, 0, \ldots, 0)$. Then $\mathbf{c} \cdot A'|\mathbf{b}(p) = (0, \ldots, 0, 1)$ and so also $\mathbf{c} \cdot A|\mathbf{b}(p) = (0, \ldots, 0, 1)$. But this contradicts the fact that p is a zero of all the polynomials generating $H^{\leq k+2}$, see Equation (8). Therefore a point $p \in Z(F^{\leq k+2}_{\overline{x_0,\ldots,x_{k-1}}})$ can always be lifted to a point $\tilde{p} \in \pi_{k-1}(Z(F))$. □

6.2 Solving algorithm

Theorem 20 gives us a condition on the initial set of polynomials F^2 for preserving the zero set during elimination, when increasing the degree by one for each variable eliminated. This is in contrast to the much faster growth in degree when using resultants and coefficient constraints.The condition from Theorem 20 is that no cross-terms between the variables $x_0, \ldots, x_{k-1}$ occur, i.e. no terms $x_i x_j$ where $0 \leq i < j < k$. If a cross-term occurs when eliminating variables, then Theorem 20 does not hold and it means that the resulting zero set after elimination may or may not increase, depending on the polynomials in the system.

We note that for the set F^2, when eliminating k variables there are maximally $\binom{k}{2}$ cross-terms. If all cross-terms occur, by performing Gaussian elimination on these terms we may write all but the $\binom{k}{2}$ first polynomials in F^2 as

$$x_0 a_0 + \cdots + x_{k-1} a_{k-1} + b,$$

where each a_i and b are in $\mathbb{B}[x_k, \ldots, x_{n-1}]$. One way of using Theorem 20 is then to eliminate k variables from the polynomials not containing any cross-terms, and simply leave out the up to $\binom{k}{2}$ polynomials containing cross-terms. The solution set of this subsystem can then be lifted back to the original system to see if it contains any false solutions due to the polynomials omitted.

In Example 2 below we give an example of a system with a cross-term where the algorithm fails. This example was carefully crafted to force the elimination algorithm to increase the solution space after two variables were eliminated. In Section 7.4 we perform experiments where we run the algorithm on randomly generated quadratic systems of equations. Because of computational limitations, these experiments only cover systems for $n \leq 24$. The zero set was always preserved when solving these systems by elimination. Hence, among the experiments performed it seems like the occurrence of cross-terms only rarely has an impact on the solution space.

Based on this observation, we propose in Algorithm 1 a simple algorithm for solving systems of quadratic Boolean equations by eliminating a number of variables. To be certain that the algorithm does not introduce any false solutions, it is necessary that the polynomials in F^2 contain no cross-terms, but in practice we expect it to work very often even if some cross-terms are present. Algorithm 1 works by doing Gauss elimination on the variable x_{i-1} to be eliminated before multiplying with remaining $L^{\leq 1}$. The polynomials not containing x_{i-1} can be multiplied only with $\{1, x_{i-1}\}$ since multiplying with other variables will not give anything with respect to eliminating x_{i-1}. The polynomials containing x_{i-1} need to be multiplied with all remaining linear and constant terms.

Algorithm 1 Eliminate(F^2, k)

In: F^2 - set of Boolean equations in n variables $x_0, \ldots, x_{n-1}$ and k - number of variables to eliminate

Out: $F^{\leq k+2}_{\overline{x_0},\ldots,\overline{x_{k-1}}}$, set of Boolean equations in $n-k$ variables $x_k, \ldots, x_{n-1}$ of degree $\leq k+2$.

for i from 0 to $k-1$ **do**

$F^{\leq i+2}_{\overline{x_0},\ldots,\overline{x_{i-2}},x_{i-1}}, F^{\leq i+2}_{\overline{x_0},\ldots,\overline{x_{i-2}},\overline{x_{i-1}}} \leftarrow$ Gauss eliminate $F^{\leq i+2}_{\overline{x_0},\ldots,\overline{x_{i-2}}}$ w.r.t terms containing x_{i-1})

$F^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},} \leftarrow \{1, x_{i-1}\} F^{\leq i+2}_{\overline{x_0},\ldots,\overline{x_{i-2}},\overline{x_{i-1}}}$

$F^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},x_{i-1}}, F^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},\overline{x_{i-1}}} \leftarrow$ Gauss eliminate $F^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},}$ w.r.t. terms containing x_{i-1})

$G^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}}} \leftarrow L^{\leq 1}_{\overline{x_0},\ldots,\overline{x_{i-2}}} F^{\leq i+2}_{\overline{x_0},\ldots,\overline{x_{i-2}},x_{i-1}}$

$G^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},x_{i-1}}, G^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},\overline{x_{i-1}}} \leftarrow$ Gauss eliminate $G^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}}}$ w.r.t. terms containing x_{i-1})

$F^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},\overline{x_{i-1}}} \leftarrow F^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},\overline{x_{i-1}}} \cup G^{\leq i+3}_{\overline{x_0},\ldots,\overline{x_{i-2}},\overline{x_{i-1}}}$

end for

Return $F^{\leq k+2}_{\overline{x_0},\ldots,\overline{x_{k-1}}}$

The following examples will illustrate how this algorithm works, as well as when the algorithm fails due to the occurrence of cross-terms.

Example 1. What may happen if we start with cubic instead of quadratic equations: Let $F^3 = \{x_0x_1x_2 + x_0x_1 + x_0x_2 + 1\}$. Then the zero set of F^3 is

$$Z(F^3) = \{(1,0,1),(1,1,0),(1,1,1)\},$$

and $\pi_0(Z(F^3) = \{(0,1),(1,0),(1,1)\}$. If we multiply with $L^{\leq 1} = \{1, x_0, x_1, x_2\}$ to eliminate x_0, then the set $L^{\leq 1}F^3$ is given by:

$$\begin{gathered} x_0x_1x_2 + x_0x_1 + x_0x_2 + 1 \\ x_0x_1x_2 + x_0x_1 + x_0x_2 + x_0 \\ x_0x_1 + x_1 \\ x_0x_2 + x_2 \end{gathered}$$

It can readily be checked that the terms containing x_0 in the four polynomials are linearly independent, so $L^{\leq 1}F^3 \cap \mathbb{B}[x_1, x_2] = \emptyset$. This means that $Z(L^{\leq 1}F^3 \cap \mathbb{B}[x_1, x_2]) = \mathbb{F}_2^2$, which has one more point than $\pi_0(Z(F^3))$. This implies that $Z(L^{\leq 1}F^3 \cap \mathbb{B}[x_1, x_2]) \supsetneq \pi_0(Z(F^3))$ and shows that our algorithm does not work when starting with cubic polynomials.

If we eliminate x_2 instead of x_0 from $L^{\leq 1}F^3$, we obtain $L^{\leq 1}F^3 \cap \mathbb{B}[x_0, x_1] = \{x_0 + 1, x_0x_1 + x_1\}$ which has the zero set $\{(1,0),(1,1)\}$. The projection of $Z(F^3)$ on the x_2-coordinate is $\{(1,0),(1,1)\}$, which means that $Z(LF^3 \cap \mathbb{B}[x_1, x_2])$ is equal to the x_2-projection of $Z(F^3))$. This example shows that whether or not a zero set is preserved after projection and elimination depends on which variable is being eliminated.

Transforming cubic equations into quadratic equations: We can transform the system F^3 into a system of quadratic equations F^2 by introducing a third variable x_3 with the relation $x_3 = x_0x_2$:

$$F^2 = \{x_1x_3 + x_0x_1 + x_3 + 1, x_0x_2 + x_3\}$$

.

It can be checked that $Z(F^2) = \{(1,0,1,1),(1,1,0,0),(1,1,1,1)\}$. Next we multiply this set with $L^{\leq 1} = \{1, x_0, x_1, x_2, x_3\}$ and eliminate x_3 to get back to the setting above. Here the set $L^{\leq 1}F^2$ is given by

$$\begin{gathered} \{x_1x_3 + x_0x_1 + x_3 + 1, x_0x_1 + x_1, x_1x_2x_3 + x_0x_1x_2 + x_2x_3 + x_2, \\ x_0x_1x_3 + x_1x_3, x_0x_2 + x_3, x_0x_1x_2 + x_1x_3, x_0x_2 + x_2x_3, x_0x_2x_3 + x_3\} \end{gathered}$$

From this it can easily be verified that

$$L^{\leq 1}F^2 \cap \mathbb{B}[x_0, x_1, x_2] = \{x_0x_1x_2 + x_0x_2 + x_0x_1 + x_0, x_0x_1 + x_1, x_0 + 1\},$$

and that $Z(L^{\leq 1}F^2 \cap \mathbb{B}[x_0, x_1, x_2]) = \{(1,0,1),(1,1,0),(1,1,1)\}$ which equals the projection of $Z(F^2)$ onto the x_3-coordinate. Moving on to eliminate x_0 and following the algorithm we compute $L^{\leq 1}_{\overline{x_3}}F^3 = L^{\leq 1}_{\overline{x_3}}(L^{\leq 1}F^2 \cap \mathbb{B}[x_0, x_1, x_2]))$, providing us with the following polynomials

$$\{x_0x_1x_2 + x_0x_2 + x_0x_1 + x_0, x_0x_1x_2 + x_1x_2, x_0x_2 + x_2, \quad x_0x_1 + x_1, x_0 + 1\}.$$

If we do Gauss reduction on the x_0-terms to eliminate x_0 from this set we end up with $\{x_1x_2 + x_2 + x_1 + 1\}$. It follows that $Z(L^{\leq 2}F^2 \cap \mathbb{B}[x_1, x_2]) = \{(1,0),(0,1),(1,1)\}$ which equals the projection of $Z(F^2)$ onto the x_0- and x_3-coordinates.

Example 2. What may happen if we have cross-terms: Let $F^2 = \{x_0x_1 + 1, x_3x_0 + x_2x_1 + x_4x_5\}$. Note that the first polynomial immediately gives $x_0 = x_1 = 1$, so this actually corresponds to solving the equation $x_3 + x_2 + x_4x_5 = 0$. Then $Z(F^2)$ has 8 solutions, and it follows that the projection of $\pi_0(Z(F^2))$ also consists of 8 solutions. Following the algorithm we multiply the set of equations with $L^{\leq 1} = \{1, x_0, x_1, x_2, x_3, x_4, x_5\}$, and proceed to eliminate x_0. The set $L^{\leq 1}F^2 \cap \mathbb{B}[x_1, x_2, x_3, x_4, x_5]$ is given by.

$$\begin{gathered} x_1x_2x_3 + x_3x_4x_5 + x_1x_2 + x_4x_5 \\ x_1x_4x_5 + x_1x_2 + x_3 \\ x_1 + 1 \end{gathered}$$

It can easily be checked that $Z(L^{\leq 1}F^2 \cap \mathbb{B}[x_1, x_2, x_3, x_4, x_5]) = \pi_0(Z(F^2))$, consisting of the same 8 solutions without x_0. Next we multiply the set $L^{\leq 1}F^2 \cap \mathbb{B}[x_1, x_2, x_3, x_4, x_5]$ with $L^{\leq 1}_{\overline{x_0}} = \{1, x_1, x_2, x_3, x_4, x_5\}$, and continue by eliminating x_1. Then

$$L^{\leq 1}_{\overline{x_0}}(L^{\leq 1}F^2 \cap \mathbb{B}[x_1, x_2, x_3, x_4, x_5]) \cap \mathbb{B}[x_2, x_3, x_4, x_5] = \{x_2x_3x_4x_5 + x_3x_4x_5 + x_2x_4x_5 + x_4x_5\}.$$

This polynomial may be written as $x_4x_5(x_2+1)(x_3+1)$, and has only one value-assignment out of the 16 possible that does not fit, namely $x_4 = x_5 = 1$ and $x_2 = x_3 = 0$. Hence it follows that $Z(L^{\leq 2}F^2 \cap \mathbb{B}[x_2, x_3, x_4, x_5])$ consists of 15 solutions, while $\pi_1(Z(F^2))$ only has 8. This shows an example where the elimination algorithm fails to preserve the zero-sets due to the cross-term in the initial polynomial $x_0x_1 + 1$.

7 Complexity

The complexity of the elimination algorithm, and how many variables we need to eliminate before the system can be solved by re-linearization, depends on various parameters.

7.1 Bounds on the degrees of the elimination algorithm

The elimination procedures discussed in this paper eliminate variables iteratively. Hence, the maximum degree encountered will be less than n. For the procedures in Section 4.3, Theorem 13 showed that those procedures will never have to handle polynomials of degree higher than $n - log_2(n) + 1$.

Now consider Algorithm 1 in Section 6. With an input system consisting of quadratic polynomials, no cross-terms, and no restrictions on the degrees of

the polynomials produced, this method is certain to find all solutions to the input system. We proceed to show that the maximum degree that Algorithm 1 needs to deal with is typically much less than $n - log_2(n) + 1$. This is based on the observation that if F^2 is the input system to the elimination algorithm, then the maximal degree of the polynomials in $\langle F^{i+2}_{\overline{x_0},\ldots,\overline{x_{i-1}}}\rangle$ after i eliminations is $i + 2$. Note that we are not guaranteed that the zero set is preserved when running the algorithms *if* cross-terms should occur, but by choosing appropriate elimination orders we could potentially limit this problem. The following lemma shows how we can greatly improve the degree bound.

Lemma 21 *Let F^2 be the input system to Algorithm 1. The maximal degree of any polynomial in any $\langle F^{i+2}_{\overline{x_0},\ldots,\overline{x_{i-1}}}\rangle$ is upper bounded by $\lfloor (n+2)/2 \rfloor$.*

Proof Let D be the maximal degree encountered during the elimination algorithm. We know that $D = max_i(min\{i + 2, n - i\})$. Since $i_0 + 2 = n - i_0$ for $i_0 = (n - 2)/2$, it is easy to see that $D = \frac{n+2}{2}$ for even n, and $D = \frac{n+1}{2}$ for odd n. □

7.2 The number of columns in the Macaulay matrix

Algorithm 1 is applied to a set F^2 of quadratic equations in n variables. Let $\mu^*(F^2, t)$ be the number of columns of the Macaulay matrix observed after elimination of t variables, where t runs from 0 to $n - 1$. The number $\mu^*(F^2, t)$ depends not only on the exact form of F^2, but also on the exact implementation of the elimination process at each step, hence the exact value of $\mu^*(F^2, t)$ is hard to predict without actually running the program implementing the algorithm. However, an obvious upper bound on $\mu^*(F^2, t)$ is the total number $\mu(n, t)$ of monomials in $n - t$ variables of degree up to $t + 2$.

$\mu(n, t)$ can be computed directly and exactly as

$$\mu(n,t) = \sum_{i=0}^{\min\{2+t,n-t\}} \binom{n-t}{i}. \tag{9}$$

Figure 1 shows the typical behaviour of $\mu(n, t)$: As t increases, the number of not-yet-eliminated variables decreases but the degree increases. For $t \geq n/2 - 1$, $\min\{2 + t, n - t\} = n - t$ so $\mu(n, t + 1) < \mu(n, t)$, and hence $\mu(n, t)$ reaches a maximum for some t in the interval $[0, n/2 - 1]$. We want to determine the maximum $\mu(n) = max_t\{\mu(n, t)|0 \leq t < n/2\}$. Again, the maximum can easily be determined for moderate values of n. By numerical inspection, the value of t that maximizes (9) seems to approach a fraction $.276\cdots$ of n. Asymptotically, we have the following result.

Proposition 22 *For $n > 25$, $\mu(n) < 2^{\log_2(\Phi)n} = 2^{n \cdot 0.694241\cdots}$, where Φ is the golden ratio.*

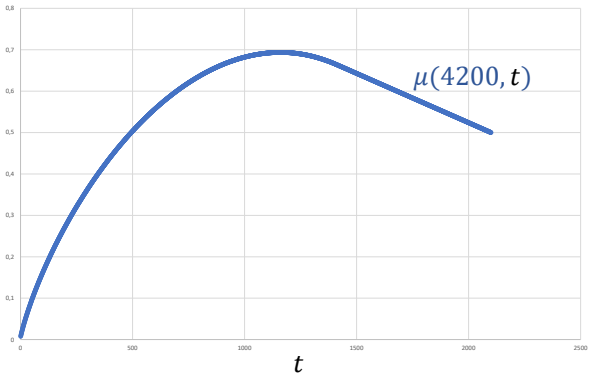

Fig. 1 Graph of the function $\log_2(\mu(4200,t))/4200$.

Proof It can be shown by direct computation that for $n \leq 25$ the result does not hold. On the other hand,

$$\mu(n,t) = \sum_{i=0}^{\min\{t+2,n-t\}} \binom{n-t}{i} \tag{10}$$

$$\leq \sum_{i=0}^{\min\{t+2,n-t\}} \binom{n+2-i}{i} \tag{11}$$

$$\leq \sum_{i=0}^{\lfloor (n+2)/2 \rfloor} \binom{n+2-i}{i} = \mathcal{F}_{n+3}, \tag{12}$$

where the last inequality follows from Lemma 21 and where $\mathcal{F}_j$ is the j-th Fibonacci number [24]. For large n the inequalities above approaches equalities, so asymptotically we get

$$\lim_{n\to\infty} \frac{\log_2 \mu(n)}{n} = \lim_{n\to\infty} \frac{\log_2 \mathcal{F}_{n+3}}{n} = \lim_{n\to\infty} \frac{\log_2 \mathcal{F}_n}{n} = \log_2(\Phi) = 0.694241\cdots,$$

where we have used $\lim_{n\to\infty} \mathcal{F}_n/\mathcal{F}_{n-1} = \Phi$, a well-established relation between $\mathcal{F}_n$ and Φ. This proves the *asymptotic* result. This still leaves the potential gap between $\mathcal{F}_n$ and $\mathcal{F}_{n+3}$, but in fact, by direct computation it can be seen that for $25 < n < 64$, $\mu(n) > \mathcal{F}_n$ but $\log_2(\mu(n))/n < \log_2(\Phi)$, while for $n \geq 64$, $\mu(n) < \mathcal{F}_n$ so the arguments above can be tightened and the result follows. □

7.3 Solving complexity

Proposition 22 shows that solving a system of quadratic equations by repeatedly multiplying all equations with all linear terms and eliminating one vari-

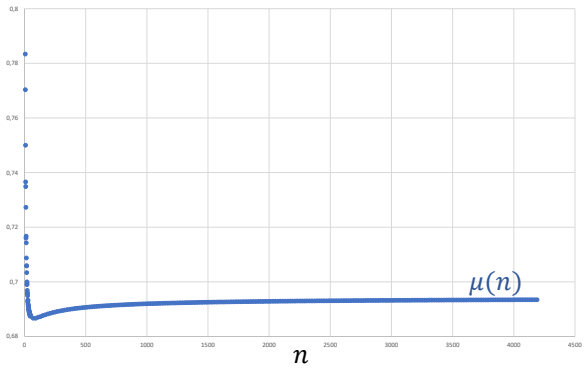

Fig. 2 Graph of $\log_2(\mu(n))/n$ for $n \leq 4200$.

able every time, we will never have to handle more than $2^{\log_2(\Phi)n}$ monomials in the Macaulay matrix. We also know that the system will be solved by re-linearization after elimination of at most $n/2$ variables. This follows from the fact that multiplying with all linear terms $n/2$ times is essentially the same as multiplying with all monomials of degree up to $n/2$, and from the fact that when $n/2$ variables have been eliminated only monomials of degree up to $n/2$ remain.

We now show that a system will be solved after eliminating much fewer variables than $n/2$. The output of the solving algorithm after elimination of the k variables $x_0, \ldots, x_{k-1}$ is the same as multiplying the initial system F^2 with all monomials of degree up to k, and then do Gaussian elimination on all monomials depending on any of $x_0, \ldots, x_{k-1}$.

In [25] and [26] the authors study how many linearly independent polynomials we get when multiplying a set of independent polynomials with all monomials up to some degree k. Under some mild condition regarding the dependencies that arise, the following formula is arrived at for computing I, the number of linearly independent polynomials after multiplying a system of m quadratic equations in n variables with all monomials of degree up to k:

$$I = I(n,k,m) = \sum_{i=0}^{\lfloor k/2 \rfloor} (-1)^i \binom{m+i}{i+1} \sum_{j=0}^{k-2i} \binom{n}{j}.$$

In [25] it is reported that the formula has been tested on many random systems, and gives a very accurate estimate on the number of independent equations produced.

After multiplication with all monomials up to degree k we get a system of polynomials of degree $\leq k+2$, and we proceed to eliminate $x_0 \ldots, x_{k-1}$ from it by doing Gaussian elimination on all monomials depending on any of the variables $x_0, \ldots, x_{k-1}$. There are $\sum_{i=0}^{k+2} \binom{n}{i}$ monomials in total, and there are $\sum_{i=0}^{k+2} \binom{n-k}{i}$ monomials that do not depend on any of $x_0, \ldots, x_{k-1}$. Hence there

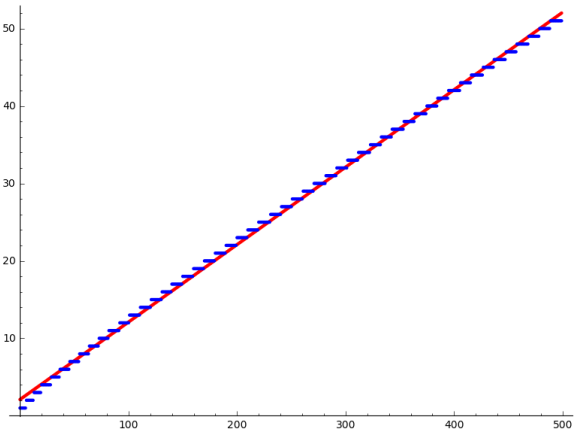

Fig. 3 $\delta(n,n)$ and $n/10+2$ for $1 \leq n \leq 500$. The blue step-wise function is $\delta(n,n)$ and the red straight line is $n/10+2$.

are $\sum_{i=0}^{k+2}\binom{n}{i} - \sum_{i=0}^{k+2}\binom{n-k}{i}$ monomials that touch at least one of the variables $x_0,\ldots,x_{k-1}$. In the worst case all of these monomials occur in the Macaulay matrix, and we lose the same number of polynomials in the Gauss reduction that eliminates the variables from the system. Subtracting this number from $I(n,k,m)$ means we will have at least

$$I(n,k,m) - \sum_{i=0}^{k+2}\binom{n}{i} + \sum_{i=0}^{k+2}\binom{n-k}{i}$$

independent polynomials left in $L^{\leq k}F^2 \cap \mathbb{B}[x_k,\ldots,x_{n-1}]$. When this number is bigger than $\sum_{i=0}^{k+2}\binom{n-k}{i}$, which is the total number of monomials remaining, we have more independent polynomials than monomials and can solve the system by re-linearization. Define $\epsilon(n,k,m)$ to be

$$\epsilon(n,k,m) = I(n,k,m) - \sum_{i=0}^{k+2}\binom{n}{i},$$

and let $\delta(n,m)$ be the first k that gives $\epsilon(n,k,m) > 0$. Then $\delta(n,m)$ gives the number of variables we need to eliminate before the system can be solved by re-linearization.

We have made plots of $\delta(n,m)$ for $n \leq 500$ and some values of m to see the behaviour of the function. Figure 3 shows the plot for $\delta(n,n)$, together with the function $n/10+2$. As we can see, $n/10+2$ gives a close approximation to $\delta(n,n)$, for $n \leq 500$. We therefore propose the following conjecture.

Conjecture 23 *For $m = n$, Algorithm 1 will succeed in solving the system after at most $n/10+2$ eliminations have been done.*

Unsurprisingly, when the initial system is overdetermined we can expect to solve it with fewer eliminations. Figure 4 shows the plot for $\delta(n,2n)$ when

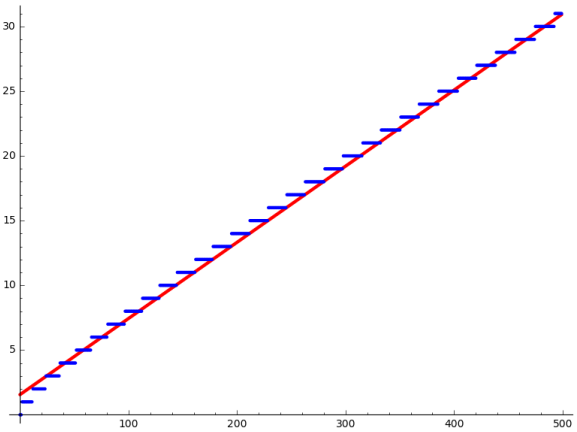

Fig. 4 $\delta(n, 2n)$ and $n/17 + 3/2$ for $1 \leq n \leq 500$. The blue step-wise function is $\delta(n, 2n)$ and the red straight line is $n/17 + 3/2$.

$n \leq 500$, together with $n/17 + 3/2$. As we can see, the function $n/17 + 3/2$ is a good approximation for $\delta(n, 2n)$, giving a similar conjecture.

Conjecture 24 *For $m = 2n$, Algorithm 1 will succeed in solving the system after at most $n/17 + 3/2$ eliminations have been done.*

We can try to generalize from these observations. Let $\alpha = m/n$ be the degree of overdeterminedness for a system of quadratic Boolean equations. For $\alpha > 2$, the relative required number of eliminations decreases further. We also know that for $\alpha = n/2$, we have $\binom{n}{2}$ equations in n variables in the initial system and therefore do not need to do any eliminations, but can solve the system by re-linearization directly.

We conjecture that for $\alpha \in [1, n/2]$, the function $\delta(n, \alpha n)$ can in general be approximated by a linear function $\frac{n}{g(\alpha)} + c$, where c is a small constant and $g(\alpha)$ is strictly increasing with $g(n/2) = \infty$ or $g(n/2) = n$. We have computed values of $g(\alpha)$ such that $g(\alpha)$ approximates $\delta(n, \alpha n)$ well, for $\alpha \in [1, 4]$ with increments of 0.25 and $n = 500$. The plot of g we get is shown in Figure 5.

On the range of α that we are considering we note that $g(\alpha)$ can be well approximated by the linear function $6\alpha + 9/2$. Using this approximation, we propose the following conjecture.

Conjecture 25 *A system of quadratic Boolean equations in $n \leq 500$ variables will be solved by Algorithm 1 after approximately $\frac{n}{6\alpha+9/2}$ eliminations.*

Going back to the function $\mu(n, t)$, we can estimate how big the Macaulay matrix will be when the system is ready to be solved by re-linearization. Setting $t = n/(6\alpha+9/2)$, this leads to a final complexity for solving a quadratic system with $m = \alpha n$ equations as

$$C(n, \alpha) = (\mu(n, n)/(6\alpha + 9/2)))^{\omega},$$

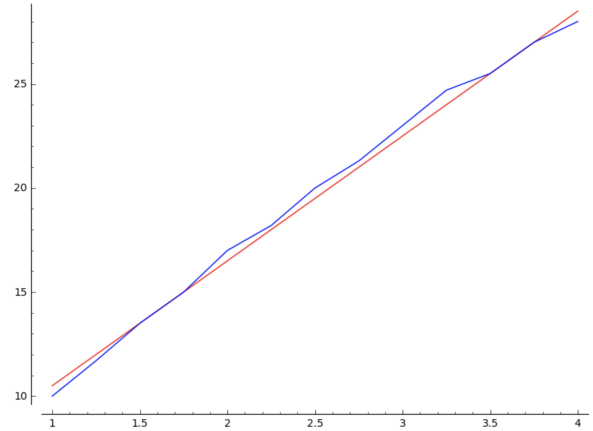

Fig. 5 The blue curve shows the observed $g(\alpha)$ such that $n/g(\alpha)$ approximates $\delta(n, \alpha n)$, computed with $n = 500$. The red line shows the linear function $6\alpha + 9/2$.

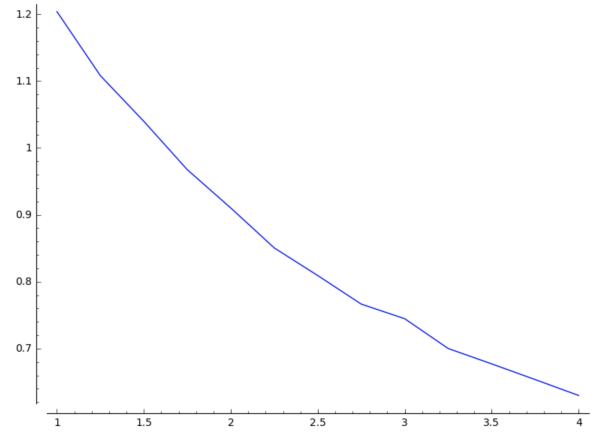

Fig. 6 $\log_2(C(500, \alpha))/500$, for $\alpha \in [1, 4]$ and $\omega = 2.7$.

where ω is the exponent for doing Gauss reduction. Figure 6 shows our last plot, where we have computed $\log_2(C(500, \alpha))/500$, for $\alpha \in [1, 4]$ using $\omega = 2.7$. This function is decreasing with increasing values of α, and when the values of $\log_2(C(500, \alpha))/500$ drops below 1 we can expect to solve the system faster than exhaustive search. This happens for $\alpha \approx 1.65$. For instance, for $\alpha = 3$ we get a solving complexity for Algorithm 1 of $\mathcal{O}(2^{0.745n})$.

7.4 Experimental results

We have implemented Algorithm 1 and tested it on some random quadratic equation systems. The tests were done on a MacBook Pro with a 2.3 GHz Intel Core i5 processor and 16GB RAM. Due to the limited computing resources

	$m = n$	$m = 1.5n$	$m = 2n$
$n = 10$	2	2	1
$n = 14$	3	2	2
$n = 17$	3	3	2
$n = 20$	3	3	2
$n = 24$	*	3	2

Table 1 Number of eliminations needed to solve random quadratic equation systems. For $m = n = 24$ the computer ran out of memory after three eliminations.

we were only able to test the method on systems with up to $n \leq 24$ variables. Our priority here is not to demonstrate the efficiency of the algorithm, but to observe the behaviour of the algorithm and to verify that the theoretical analysis discussed above is consistent with practice.

For each n we made systems with $m = n, m = 1.5n$ and $m = 2n$ equations in them. The polynomials in the systems were created by choosing the coefficient of each monomial uniformly at random, with one exception. We wanted every system to have a solution, so some random point in $\mathbb{F}_2^n$ was chosen to be a solution. This was realized by adjusting the constant term of each polynomial accordingly. Please note that the systems contain cross-terms of the variables to be eliminated with overwhelming probability, but we observed no impact of this on the ability of the algorithm to solve the systems.

Next, we ran Algorithm 1 on the systems and checked how many eliminations k were needed before the system could be solved by relinearization (i.e. when we get k independent linear polynomials in the set). The results are summarized in Table 1. As the table shows, the systems get solved after a few eliminations and the experimental results closely match the analytical estimates given in the previous subsection. Admittedly, the values of n of systems we solved are rather small, but it is reassuring to see that theory and practice give the same result.

8 Conclusions

In this paper, we have studied how to eliminate variables from Boolean equation systems when the degree of polynomials produced is upper bounded. The tools we use for elimination are the known techniques of normalization and resultants. Also, we introduced coefficient constraints, which only applies to characteristic 2.

The motivation for our study comes from solving non-linear Boolean equation systems. The best known (and best-known) algorithms to solve such systems are focused on Gröbner bases (F4/F5) or re-linearization after multiplying all polynomials with a set of monomials (XL). We relate the work in this paper to these approaches. What we found is that our elimination tools may be explained in terms of Gröbner basis algorithms, but where we only reduce an S-polynomial modulo a sub-basis and stop the reduction as soon as the variable in question has been eliminated. With regard to XL methods,

we found that our elimination procedure will compute the same output as multiplying all polynomials with all allowed monomials and doing Gaussian elimination to remove a variable from the system. The elimination procedure presented in this paper is a lot more efficient though, since we do not multiply with all monomials but only compute precisely what is needed to eliminate the desired variable.

Eliminating variables may be explained in terms of syzygies, where Koszul syzygies correspond to resultants, and Boolean syzygies correspond to coefficient constraints. We found that when the syzygies are computed for all linear polynomials, all syzygies can be generated by the syzygies of the Koszul and Boolean types.

As an alternative approach, we may eliminate k variables from the system at the expense of increasing the degree by the same value k, while the solution space will remain intact (by projection) provided the original system does not contain any cross-terms between any of the variables being eliminated.

This insight allows us to bound the number of monomials occurring in the systems as the elimination iterations proceed, and gives an estimate on the number of variables that need to be eliminated before the system can be solved by relinearization. This estimate is given by a linear function in n with a small coefficient. Numerical experiments on moderate size examples correspond well to this estimate. For sufficiently large and overdetermined systems, the total complexity of solving by re-linearization is faster than that of exhaustive search.

To limit the length of this paper, we have avoided the discussion of several closely related questions. One topic for further research is to investigate how much of the complete syzygy space is generated by the trivial Boolean and Koszul syzygies, or, in other words, how rare nontrivial syzygies are. Another line of investigation should examine and quantify the introduction of false solutions associated with the limitation of the degree. Yet another approach is to design a hybrid algorithm incorporating ideas by Joux and Vitse [19] or Lokshtanov et. al. [23].

Acknowledgement

We would like to thank the anonymous reviewers for helpful comments.

References

1. M. Brickenstein, A. Dreyer *A framework for Gröbner-basis computations with Boolean polynomials*, J. Symbolic Comput. 44, no.9, (2009), pp. 1326–1345. PolyBoRi Polynomials over Boolean Rings. http://polybori.sourceforge.net/.
2. D.Cox, J.Little, D.O'Shea, *Ideals, varieties and algorithms*, Third edition, 2007 Springer Science and Business Media.
3. D.Cox, J.Little, D.O'Shea *Using Algebraic Geometry* GTM 185, Springer Science and Business Media 2005.

4. W. Decker, G.-M. Greuel, G. Pfister, H. Schönemann, 2010. Singular 3-1-2 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de/.
5. A. Kandri-Rody, D. Kapur, P. Narendran, *An ideal-theoretic approach to word problems and unification problems over finitely presented commutative algebras*, In: Jouannaud JP. (eds) Rewriting Techniques and Applications. RTA 1985. Lecture Notes in Computer Science, vol 202. Springer, Berlin, Heidelberg
6. A. Shamir, J. Patarin, N. Courtois, A. Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, pp. 392 — 407, Springer 2000.
7. Courtois N.T., Pieprzyk J. *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Advances in Cryptology — ASIACRYPT 2002. ASIACRYPT 2002. Lecture Notes in Computer Science, vol 2501, pp. 267 – 287. Springer, Berlin, Heidelberg 2002.
8. J-C. Faugere. *A new efficient algorithm for computing Gröbner bases (F4). Effective methods in algebraic geometry* (Saint-Malo, 1998), J. Pure Appl. Algebra 139 (1999)
9. J-C. Faugere. *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)*, Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, 75–83, ACM, New York, 2002.
10. T. Stegers. *Faugere's F5 Algorithm Revisited*, Thesis For The Degree Of Diplom-Mathematiker, Department of Mathematics, Technische Universität Darmstadt, 2005. Available at http://sciencedocbox.com/Physics/68613748-Faugere-s-f5-algorithm-revisited.html
11. J. Horácek, M. Kreuzer, A.S.M. Ekossono. *Computing Boolean border bases*, Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2016 18th International Symposium on. IEEE, 2016, pp. 465–472.
12. M.Giusti. *Some effectivity problems in polynomial ideal theory* In Proc. Int. Symp. on Symbolic and Algebraic Computation EUROSAM 84, Cambridge (England), volume 174 of LNCS, pages 159–171. Springer, 1994.
13. T.M. Chan and R. Williams. *Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky.* In Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1246–1255, 2016.
14. K. Sakai, Y. Sato, *Boolean Gröbner bases*, ICOT Technical Momorandum 488 (1988). http://www.jipdec.or.jp/archives/icot/ARCHIVE/Museum/TRTM/tm0488.htm.
15. M. Sala, T. Mora, L. Perret, S. Sakata, C. Traverso, *Gröbner Bases, Coding and Cryptography*, Springer, 2009
16. Bardet,M., J.-C.Faugere and B.Salvy, *Complexity of Gröbner basis computation for semiregular overdetermined sequences over $\mathbb{F}_2$ with solutions in $\mathbb{F}_2$*, rapport de recherche 5049, Institut National de Recherche en Informatique et en Automatique, Lorraine, 2003.
17. M. Bardet, J.-C. Faugere, B. Salvy, and P.-J. Spaenlehauer. *On the complexity of solving quadratic Boolean systems.* Journal of Complexity, 29(1):53–75, 2013.
18. M. Bardet, J.-C. Faugere, B. Salvy, and B.-Y. Yang. *Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems.* Presented at MEGA'05, Eighth International Symposium on Effective Methods in Algebraic Geometry, 2005.
19. A. Joux and V. Vitse, *A crossbred algorithm for solving Boolean polynomial systems*, Cryptology ePrint Archive, Report 2017/372, 2017. https://eprint.iacr.org/2017/372.
20. C. Bouillaguet, H.-C. Chen, C.-M. Cheng, T. Chou, R. Niederhagen, A. Shamir and B.-Y. Yang. *Fast exhaustive search for polynomial systems in $\mathbb{F}_2$.* In Cryptographic hardware and embedded systems – CHES 2010. 12th international workshop, Santa Barbara, USA, August 17–20, 2010. Proceedings, pages 203–218. Berlin: Springer, 2010.
21. Rosen, K.H., *Handbook of Discrete and Combinatorial Mathematics.* In the Series *Discrete Mathematics and Its Applications*, Taylor & Francis, 1999.
22. D Lazard. *Gaussian elimination and resolution of systems of algebraic equations.* In proc. EUROCAL 1983, vol 162 of LNCS, pp. 146-157, 1983.
23. D. Lokshtanov, R. Paturi, S. Tamaki, R. Williams, and H. Yu. *Beating brute force for systems of polynomial equations over finite fields.* The 27th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2017.
24. Rosen, Kenneth H., *Handbook of Discrete and Combinatorial Mathematics, 2nd Edition*, CRC Press, 2017.

25. H. Raddum, S. Rønjom, *On the Number of Linearly Independent Equations Generated by XL*, Sequences and Their Applications (SETA) 2008, LNCS 5203, pp. 239 – 251, Springer, 2008.
26. Yang, B.-Y., Chen, J.-M. *Theoretical Analysis of XL over Small Fields*, Australasian Conference on Information Security and Privacy (ACISP) 2004, LNCS 3108, pp. 277 -– 288, Springer, 2004.
27. R. Smolensky. *Algebraic methods in the theory of lower bounds for Boolean circuit complexity.* In Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC), pp. 77—82, 1987
28. P.Zajac, *Upper bounds on the complexity of algebraic cryptanalysis of ciphers with a low multiplicative complexity*, Designs, Codes and Cryptography, Volume 82, Issue 1–2, pp. 43-–56, 2017

Appendix: Syzygies between polynomials of degrees ≥ 2

When the degree of some a_i's are greater than 1, there may be other syzygies that are not generated by the Koszul and Boolean syzygies. Let $a_1^1, \dots a_{\ell_1}^1$ be polynomials of degree ≤ 1. By suitable Gaussian elimination we may assume the initial terms are such that:

$$\text{in}(a_1^1) > \text{in}(a_2^1) > \cdots > \text{in}(a_{\ell_1}^1). \tag{13}$$

Let $a_1^d, \dots, a_{\ell_d}^d$ be polynomials of degree $\leq d$. We perform reduction operations as follows: If a term of a_i^d is of the form $t \cdot \text{in}(a_j^1)$ where t is a monomial of degree $\leq d-1$, we replace a_i^d by $a_i^d - t \cdot a_j^1$. We then eventually get:

$$\text{No term of } a_i^d \text{ is } t \cdot \text{in}(a_j^1) \text{ where } t \text{ is a monomial of degree} \leq d-1. \tag{14}$$

Secondly we may perform Gaussian elimination on the a_i^d such that:

$$\text{in}(a_1^d) > \text{in}(a_2^d) > \cdots > \text{in}(a_{\ell_d}^d). \tag{15}$$

Suppose we have given a_i^d as above for each $1 \leq d \leq D$ and $i = 1, \dots, \ell_d$. Let

$$\mathbb{B}^L = \mathbb{B}^{\ell_1} \oplus \cdots \oplus \mathbb{B}^{\ell_D}$$

where $\mathbb{B}^{\ell_d} = \mathbb{B}\epsilon_1^d \oplus \cdots \oplus \mathbb{B}\epsilon_{\ell_d}^d$ and we set ϵ_j^d to have degree d. There is a map

$$\mathbb{B}^L \to \mathbb{B}, \quad \epsilon_i^d \mapsto a_i^d$$

and the syzygy module $S \subseteq \mathbb{B}^L$ is the kernel of this map.

Suppose now we have a total order on the terms of $\mathbb{B}$. We make a term order on $\mathbb{B}^L$ by letting terms $s\epsilon_j^e < t\epsilon_i^d$ if:

- $e < d$, or
- $e = d$ and $j < i$, or
- $e = d, j = i$ and $s < t$

Theorem 26 *Given polynomials a_i^d of degree $\leq d$, for each $1 \leq d \leq D$ and suppose for each d they fulfill Condition* (15) *above. The following syzygies may exist:*

1. *Koszul syzygies* $a_j^d\epsilon_k^e + a_k^e\epsilon_j^d$ *where* $e < d$ *or* $e = d$ *and* $k < j$. *For given sum* $d + e$ *denote by* K^{d+e} *the linear space these syzygies generate.*
2. *Boolean syzygies* $(a_j^d + 1)\epsilon_j^d$. *For given* d *denote by* B^{2d} *the linear space these syzygies generate.*
3. *For each* $\delta \geq 2$ *syzygies*

$$\mathbf{r} = \sum_{\substack{d\,=\,1,\,\ldots\,,\,\delta \\ i\,=\,1,\,\ldots\,,\,\ell_d}} r_i^{\delta-d}\epsilon_i^d$$

where $r_i^{\delta-d}$ *has degree* $\leq \delta - d$ *and no term of* $\mathbf{r}$ *is* $\tau \cdot t$ *where* t *is the initial term of a syzygy in* K^e *or* B^e *and* $\deg(\tau) + e \leq \delta$.

For a given δ *in 3., denote by* $R^{\leq\delta}$ *the linear space of such syzygies.*

a. Then for $\delta \geq 2$ *we have:*

$$S^{\leq\delta} = \sum_{d=2}^{\delta} S^{\leq\delta-d}K^d + \sum_{d=2}^{\delta} S^{\leq\delta-d}B^d + R^{\leq\delta}. \tag{16}$$

b. Suppose in addition the a_i^d *fulfill the Condition* (14) *above. Then we may let* $R^{\leq\delta}$ *be the space of all syzygies of type 3. where the coefficient* $r_i^{\delta-1}$ *of the* a_i^1 *vanish, and we still have the above identity* (16).

Proof Given a syzygy of degree $\leq \delta$

$$\mathbf{s} = \sum_{\substack{d\,=\,1,\,\ldots\,,\,\delta \\ i\,=\,1,\,\ldots\,,\,\ell_d}} s_i^{\delta-d}\epsilon_i^d.$$

If a term in $\mathbf{s}$ is a product $n \cdot t$ where t is the initial term of a syzygy $\mathbf{s}'$ in K^p or B^p with $\deg(\tau) + p \leq \delta$, we replace $\mathbf{s}$ by $\mathbf{s} - \tau \cdot \mathbf{s}'$. In this way we continue and in the end we get syzygy as in 3. This proves the identity (16) above.

Suppose now the Condition (14) is also fulfilled. Let the following relation be of Type 3. :

$$\sum_{i=1}^{\ell_1} r_i^{\delta-1}a_i^1 + \sum_{\substack{d\,=\,2,\,\ldots\,,\,\delta \\ i\,=\,1,\,\ldots\,,\,\ell_d}} r_i^{\delta-d}\epsilon_i^d.$$

Let $x_1 = \mathrm{in}(a_1^1)$. Then no term of any other a_i^d contains x_1 and also no $r_i^{\delta-d}$ contains x_1. But then the relation above is only possible if $r_1^1 = 0$. In this way we may continue and get all $r_i^1 = 0$ except possibly if $\mathrm{in}(a_j^1)$ is the constant 1 (in which case we must have i the last index ℓ_1). But then by the reduction process using $a_{\ell_1}^1$, none of the a_i^d for $d \geq 2$ contains a term of degree $< d$ and similarly no term of the $r_j^{\delta-d}$ contains a term of degree $< \delta - d$. But then in the relation

$$r_{\ell_1}^{\delta-1} \cdot 1 + \sum_{\substack{d\,=\,2,\,\ldots\,,\,\delta \\ i\,=\,1,\,\ldots\,,\,\ell_d}} r_i^{\delta-d}a_i^d,$$

the left side has degree $\leq \delta - 1$ while the right side has all terms of degree δ. Hence $r_{\ell_1}^{\delta-1} = 0$. □

We now present the algorithm to compute $R^{\leq\delta}$ under the assumption of Conditions (15) and (14).

Algorithm to compute $R^{\leq\delta}$

1. Set $KB_{in}^{\leq 1}, R_{in}^{\leq 1}$ equal to 0. Let $\delta := 2$.
2. Let KB_{in}^{δ} consist of all pairs (t, δ) where t is the initial term of a Koszul syzygy in K^δ or a Boolean syzygy in B^δ.
3. $KB_{in}^{\leq\delta} = KB_{in}^{\leq\delta-1} \cup KB_{in}^{\delta}$.
4. If $\delta = 2$ let $R^2 = 0$. If $\delta \geq 3$ then R^δ consist of all syzygies

$$\mathbf{r} = \sum_{\substack{d\,=\,2,\,\ldots\,,\,\delta \\ i\,=\,1,\,\ldots\,,\,\ell_d}} r_i^{\delta-d}\epsilon_i^d$$

   where $r_i^{\delta-d}$ has degree $\leq \delta - d$ and no term of $\mathbf{r}$ is a product of monomials $\tau \cdot t$ where:
   - $(t, p) \in R_{in}^{\leq\delta-1} \cup KB_{in}^{\leq\delta-1}$ and τ is a monomial such that $\deg(\tau)+p \leq \delta$.
   - $\tau = 1$ and $(t, \delta) \in KB_{in}^{\delta}$
5. Perform Gaussian elimination on R^δ and let R_{in}^{δ} consists of all pairs (t, δ) where t is the initial term of a syzygy in R^δ.
6. $R_{in}^{\leq\delta} = R_{in}^{\leq\delta-1} \cup R_{in}^{\delta}$.
7. If δ is less than the stop bound then $\delta := \delta + 1$ and go to 2.

As for the actual computation of the syzygies in Step 4, this can be done by taking the $r_i^{\delta-d}$ to be linear combinations of the allowed terms (with unknown coefficients), and then solving a system of linear equations.

Proposition 27 *With the algorithm above, then*

$$R^{\leq\delta} = \sum_{d\geq 3} S^{\leq\delta-d} R^d.$$

Proof This is clear by construction. □

Our applications of Theorem 26 are typically for $\delta = 1$ or 2. (This occurs for sets $F^2, \ldots, F^d$ where $d = 3$ or 4.) We are then interested in the syzygies $S^{\leq 2}$ and $S^{\leq 3}$. These are given as follows:

$$S^{\leq 2} = K_2 + B_2$$
$$S^{\leq 3} = \langle L^1\rangle K_2 + K_3 + \langle L^1\rangle B_2 + R^3.$$