Degree Complexity for a Modified Pigeonhole Principle*

Maria Luisa Bonet†
Department de Llenguatges i Sistemes Informatics
Universitat Politecnica de Catalunya
Jordi Girona Salgado 1-3 Barcelona Spain
e-mail bonet@lsi.upc.es

Nicola Galesi‡
School of Mathematics
Institute for Advanced Study
Princeton 08540
New Jersey USA
e-mail: galesi@ias.edu

Abstract

We consider a modification of the pigeonhole principle, $MPHP$, introduced by Goerdt in [7]. Using a technique of Razborov [9] and simplified by Impagliazzo, Pudlák and Sgall [8], we prove that any Polynomial Calculus refutation of a set of polynomials encoding the $MPHP$, requires degree $\Omega(\log n)$. We also prove that the this lower bound is tight, giving Polynomial Calculus refutations of $MPHP$ of optimal degree.

Finally we prove a simple Lemma giving a simulation of Resolution by Polynomial Calculus.

1 Introduction

Razborov in [9] proved that any Polynomial Calculus (PC) refutation of a polynomial encoding of the pigeonhole principle (PHP^m_n) requires degree at least $\Omega(n)$. Here the number of pigeons m is any integer greater than n. While other techniques were developed to prove degree lower bounds in PC [5, 2, 1] for other combinatorial principles, the technique introduced by Razborov and simplified by [8] wasn't succesfully applied to other principles different from the PHP.

The core of Razborov's technique is to produce an explicit characterization of the vector space of all polynomials derivable from the polynomial encoding of PHP, using low degree Polynomial Calculus refutations. This way one can study what is the minimal degree d for which refutations of PHP of degree d exist.

The main contribution of our result is extending Razborov's technique to a combinatorial principle somewhat different from the pigeonhole principle. We consider a modification of the pigeonhole principle ($MPHP$), introduced by Goerdt [7]. $MPHP$ is defined over n pigeons and $\log n$ holes, and differs from PHP since it allows many holes to hold more than one pigeon (see Definition 2.1 for further details). In fact, $MPHP$ is minimally unsatisfiable. Note that the PC degree lower bound of Razborov doesn't give lower bounds for this principle.

In Section 3 we introduce a polynomial formulation of the $MPHP$ principle, `Poly`-$MPHP$, and we prove that any Polynomial Calculus refutation of this set of polynomials requires degree

*A prelimianry version appeared as part of the paper *A Study of Proof Search Algorithms for Resolution and Polynomial Calculus* published in the *Proceedings of the 40-th IEEE Conference on Foundations of Computer Science, 1999*

†Partly supported by Project CICYT, TIC 98-0410-C02-01 and Promoción General del Conocimiento PB98-0937-C04-03.

‡This work was done while the author was at the Universitat Politecnica de Catalunya partly Supported by an European Community grant under the TMR project.

$O(\log n)$ over any field. Following Impagliazzo Pudlák and Sgall [8], we define a new pigeon dance tailored for the $MPHP$ principle and we prove those properties that we need to define an explicit characterization of the vector space of all polynomials derivable from the `Poly`-$MPHP$ using degree d PC refutations. As a consequence we prove that the minimal degree for refuting `Poly`-$MPHP$ in PC over any field is of the order of the number of holes in $MPHP$ (i.e $\Omega(\log n)$).

In section 4 we introduce the Resolution system. Following [3], we consider the *width* (i.e. the size of the largest clause used in a refutation) as a complexity measure and we show a Polynomial Calculus simulation of Resolution. Under a fixed standard translation of CNF formulas to polynomials our simulation produces PC refutations of degree bounded by the width plus 1. The consequences of this result are discussed in the last Section where we argue that our simulation implies that the width based proof-search algorithm of [3] cannot have a better performance than the Groebner Basis proof-search algorithm of [6].

In Section 5, we approach the problem of proving tight degree upper bounds for PC refutations of `Poly`-$MPHP$. In fact using the simulation result of the previous section we give PC refutations for `Poly`-$MPHP$ of optimal degree (i.e. $O(\log n)$).

In the final section, we propose an alternative approach to obtain the upper bound of section 5 and we discuss some consequences of our results. Also we pose some problems arising from our work that could be interesting to study.

2 Preliminaries

The POLYNOMIAL CALCULUS (PC) is a refutation system for formulas in CNF. We express a CNF formula F as a sequence of polynomials $p_1, = 0, \ldots, p_m = 0$ over some field K. To force 0-1 solutions we always add among the initial polynomials the axioms $x^2 - x = 0$ for all variables x. A PC REFUTATION is a sequence of polynomials ending with $1 = 0$ such that each line in the sequence is either an initial polynomial or is obtained from two previous polynomials in the sequence by the following rules: (1) $\frac{f \quad g}{\alpha f + \beta g}$ for $\alpha, \beta \in K$; and (2) $\frac{f}{xf}$, for any variable$x$. The DEGREE of a refutation is the maximal degree of a polynomial used in the proof. The complexity of a PC refutation is given by its degree.

We define a *standard mapping* tr from formulas in CNF to sets of polynomials in the following way: (1) $tr(x) = 1 - x$; (2) $tr(\bar{x}) = x$; (3) $tr(x \vee y) = tr(x) \cdot tr(y)$. We denote by $[n]$ the set $\{1, 2, \ldots, n\}$.

We will use a tautology encoding a modification of the pigeon hole principle defined in [7]. Let n be a natural number of the form 2^k, for some k, and let $m = \log_2 n$. For each $j = 1, \ldots, m$, let $Part(j)$ be the partition of $[n]$ induced by j the following way:
$Part(j) := \{\{i, i+1, \ldots, i + (2^j - 1)\} \mid i = 1, 1 + 2^j, 1 + 2 \cdot 2^j, \ldots, 1 + (\frac{n}{2^j} - 1) \cdot 2^j\}$
If we consider $[n]$ as a set of pigeons and $[m]$ a set of holes, then for each hole $j \in [m]$, $Part(j)$ contains sets of pigeons, e.g.

$$\begin{aligned}
&Part(1) = \{\{1,2\}, \{3,4\}, \ldots \{n-1, n\}\} \\
&Part(2) = \{\{1,2,3,4\}, \ldots, \{n-3, n-2, n-1, n\}\} \\
&\vdots \\
&Part(\log_2 n) = \{\{1, 2, \ldots, n\}\}
\end{aligned}$$

Consider the following definition:

Definition 2.1 *For all $i, i' \in [n]$, i and i' are j-COMPATIBLE if and only if they are in different sets of $Part(j)$.*

We consider the following property for n pigeons and $\log_2 n$ holes. If each pigeon is sitting in some hole, then there must exist an hole j and two pigeons i and i' that are not j-compatible sitting in hole j. Our CNF formula $MPHP_n$ expresses the negation of the previous property with the further restriction that each pigeon must sit in exactly one hole.

$$
\begin{array}{lll}
(1) & \bigvee_{j=1}^{m} x_{i,j} & i \in [n] \\
(2) & \bar{x}_{i,j} \vee \bar{x}_{i',j} & j \in [m], i \neq i' \in [n] \text{ , not } j\text{-compatible} \\
(3) & \bar{x}_{i,j} \vee \bar{x}_{i,k} & i \in [n], j \neq k \in [m]
\end{array}
$$

Notice that the set of clauses defining our $MPHP_n$ subsumes the set of clauses defining the $MPHP_n$ of [7]. First, we add clauses encoding the restriction that each pigeon must sit in exactly one hole. Second, Goerdt considered the restricted, but more complicated, definition of *not compatibility* between pigeons, allowing the clauses in (2) only for not compatible piegeons i and i' lying in different half sets of $Part(j)$ (see [7]).

3 Degree Lower Bounds for the Modified PHP

In this section we show that any polynomial calculus refutation of the $MPHP_n$ requires degree $\Omega(\log n)$. We will use the same technique of [9, 8]. Recall that $m = \log n$, the definition of j-compatible pigeons and the definition of the set $Part(j)$, for all $j \in [m]$ (see Section 2). Given $Q_i := 1 - \sum_{j\in[m]} x_{i,j}$ we adopt the following polynomial formulation of the $MPHP_n$, that we call `Poly`-$MPHP_n$:

$$
\begin{array}{lll}
(1) & Q_i = 0 & i \in [n] \\
(2) & x_{i,j}x_{i,k} = 0 & i \in [n],\ j,k \in [m] \\
(3) & x_{i,j}x_{k,j} = 0 & j \in [m],\ i,k \in [n] \text{ not } j\text{-compatible} \\
(4) & x_{i,j}^2 - x_{i,j} = 0 & i \in [n],\ j \in [m]
\end{array}
$$

For a polynomial x which is a product of $x_{i,j}$, let $Pigeons(x, j)$ be the set of i's such that $x_{i,j}$ is a factor in x.

Definition 3.1 *T is the set of the monomials $x = x_{i_1,j_1} \dots x_{i_l,j_l}$ such that all i_k are distinct and for all $j_k \in [m]$ and for all i and i' in $Pigeons(x, j_k)$ i and i' are j_k-compatible. T_d is the set of monomials in T of degree at most d.*

Using the identities (2), (3) and (4) any polynomial can be represented, without increasing its degree, as a linear combination of monomials in T. Therefore any polynomial calculus refutation carried on modulo the ideal I generated by the polynomials (2), (3) and (4), is in the vector space $Span(T)$ generated from the monomials in T. From now on we assume that all the computations are modulo the ideal I.

We want to build a basis B_d for the vector space $Span(T)$ such that the elements of B_d are products of the form $\prod_{i,j} x_{i,j} \prod_i Q_i$. As in [8] (and [9]) the definition of B_d is obtained from a process that maps partial assignments into partial assignments: the pigeon dance. We consider a dummy hole 0, and we represent elements of B_d as partial assignments according to the following definition:

Definition 3.2 *A is the set of the partial mappings a from $[n]$ to $[m] \cup \{0\}$ such that for all $i, i' \in [n]$, $i \neq i'$, if $a(i) = a(i') = j \neq 0$ then i and i' are j-compatible.*

Let $A_d := \{a \in A \ : \ |a| \leq d\}$. For $a \in A$ with $a = \{(i_1, j_1), \ldots (i_k, j_k), (i'_1, 0), \ldots, (i'_l, 0)\}$, let $\hat{a}$ denote the restriction $\{(i_1, j_1), \ldots (i_k, j_k)\}$ of a. Any element $a \in A$ defines a polynomial x_a the following way: $x_a = \prod_{a(i)=j, j\neq 0} x_{i,j} \prod_{a(i)=0} Q_i$. Therefore by definition of T any polynomial $x_{\hat{a}}$ associated to $\hat{a} \in A_d$ is in T_d.

Our pigeon dance differs from that of [9, 8] since sometimes a pigeon can be sent to an occupied hole. Consider the following definition:

Definition 3.3 *Given $a \in A$, we say that a hole j* IS GOOD FOR THE PIGEON i IN a, *and we write $j \in Good(i, a)$, if $j > a(i)$ and for all $i' \in a^{-1}(j)$, i and i' are j-compatible.*

Given $a \in A$, our pigeon dance works the following way: starting from the first pigeon in $dom(a)$ we try to move all the pigeons $i \in dom(a)$ into a hole j which is *good* for i in a.

Definition 3.4 (Dance) *Let $a \in A$ and consider $dom(a)$. A pigeon dance on a is a sequence of mappings $a_0, a_1, \ldots a_n$ in A with the same domain as a, defined the following way: $a_0 = a$ and for all $0 < t \leq n$, if $a(t)$ is undefined, then $a_t = a_{t-1}$, otherwise*

$$\begin{cases} a_t(j) = a_{t-1}(j) & j \neq t \\ a_t(t) \in Good(t, a_{t-1}) \end{cases}$$

Definition 3.5 (Minimal Dance) *Let $a \in A$ be given and let t be a pigeon index in $[n]$. By $D_t(a)$ we denote a mapping $b \in A$ such that $dom(b) = dom(a)$, and defined as follows:*

$$\begin{array}{l} b(i) = a(i) \quad i \in dom(a), i \neq t \\ b(t) = min_{j\in[m]}[j \in Good(t, a)] \end{array}$$

If $min_{j\in[m]}[j \in Good(t, a)]$ does not exists, then $D_t(a)$ is undefined. The MINIMAL PIGEON DANCE *$D_{min}(a)$ on a is: $D_{min}(A) = D_n(D_{n-1}(\cdots(D_1(a))\cdots)$*

The minimal dance has two main properties. It can be always defined whenever a dance is defined, and it defines a one-to-one mapping from partial assignments to partial assignments. We show these properties in the following lemmas.

Lemma 3.1 *If there exists a dance on a, then there always exists a minimal dance on a.*

Proof. We prove by induction on $t = 1, \ldots, n$ that there is dance $b = b_0, b_1, \ldots, b_n$ where $b_0 = a$ such that its first t steps correspond to the first t steps of the minimal dance on a. The lemma hence follows for $t = n$. Assume to have proved the claim for $t - 1$, and let $b = b_0, b_1, \ldots, b_n$ the correct dance having the first $t - 1$ steps as in the minimal dance. We show how to build a new correct dance $c = c_0, c_1, \ldots, c_n$ having its first t steps as in the minimal dance.

Let $j_{min} = \min_{j\in[m]}[j \in Good(t, b_{t-1})]$ and suppose $j = b_t(t)$. Observe that since b is a correct dance, then j_{min} always exists and moreover $j_{min} \leq j$. If $j = j_{min}$, then b is making the right choice at the t-th step. In this case we have no need to change b, so we define $c_i = b_i$ for all $i = 0, \ldots, n$. Assume instead that $j_{min} < j$. In this case we define $c = c_0, c_1, \ldots, c_n$ the following way: in the first $t - 1$ steps c and b are the same, that is, for all i, $i = 1, \ldots, t - 1$, $c_i = b_i$; at the t-th step, c_t is defined by:

$$c_t(i) = \begin{cases} j_{min} & \text{if } i = t \\ b_t(i) & \text{otherwise} \end{cases}$$

The definition of c_i for for $i > t$ is as follows:

$$c_i(j) = c_{i-1}(j) \quad \text{for } j \neq i$$
$$c_i(i) = \begin{cases} j & \text{if } b_i(i) = j_{min} \text{ and } i \text{ and } t \text{ are not } j_{min}\text{-compatible} \\ b_i(i) & \text{otherwise} \end{cases}$$

We have to prove that $c = c_0, c_1, \ldots, c_n$ is a properly defined dance and its first t steps are minimal. Observe that the first $t-1$ steps of c are correct and minimal since they are the same of b. The t-th step is correct and minimal by definition of j_{min}. Therefore it remains to prove that the steps strictly greater than t define a correct dance. By the definition of c_i, for $i > t$, we have to prove that for all $i > t$ $c_i(i) \in Good(i, c_{i-1})$.□

Claim 3.1 *For all $i > t$, $c_i(i) \in Good(i, c_{i-1})$.*

Proof. (**of Claim3.1**)
By the definition of $c_i(i)$ for $i > t$, it is easy to see that we have to prove that for all $i > t$ such that $b_i(i) = j_{min}$ and i and t are not j_{min}-compatible, then $j \in Good(i, c_{i-1})$. We obtain the Claim showing that: (1) there is at most one i such that $b_i(i) = j_{min}$ and i and t are not j_{min}-compatible; and (2) for this i we have that $j \in Good(i, c_{i-1})$.

The first property easily follows since if there exist two different pigeons i and i' both not j_{min}-compatible with t, then i, i' and t are in the same set $D \in Part(j_{min})$. But this is not possible since b is a correct dance and therefore i and i' must be j_{min}-compatible.

For the second point, assume we have an i such that $b_i(i) = j_{min}$ and i and t are not j_{min}-compatible. We prove that i is j-compatible with all elements in $c_{i-1}^{-1}(j)$, which proves that $j \in Good(i, c_{i-1})$. If $c_{i-1}^{-1}(j) = \emptyset$, the result is trivial. Assume, for the sake of contradiction, that there is a $i' \in c_{i-1}^{-1}(j)$, which is not j-compatible with i. Therefore i and i' are in the same set $B \in Part(j)$. Since i is the only pigeon on which we have modified the dance b (except for t), then it follows that i' was already sent to j in b, that is $b_{i'}(i') = j$. Observe that, since i and t are not j_{min}-compatible, then i and t are in the same set $C \in Part(j_{min})$. But since $j_{min} < j$, then $C \subset B$ and therefore $t \in B$. This is a contradiction since b is a correct dance and we cannot have that $b_t(t) = j$ and $b(i) = j$, for two pigeons i and t not j-compatible. □

Lemma 3.2 *The minimal dance is a one-to-one mapping.*

Proof. We show that for all $t = 1, \ldots, n$, $D_t(\cdot)$ is a 1-1 mapping. The result then follows since the minimal dance is a composition of the D_t mappings. We show that if $D_t(a) = D_t(a')$ then $a = a'$. Suppose $D_t(a) = D_t(a')$. Then $dom(a) = dom(a')$ and $a(i) = a'(i)$ for all $i \in dom(a), i \neq t$. It remains to show that $a(t) = a'(t)$. We show that neither $a(t) < a'(t)$ nor $a'(t) < a(t)$. Suppose the former. We show the following two inequalities:

$$D_t(a)(t) \leq a'(t) \quad a'(t) < D_t(a')(t)$$

This leads to a contradiction since $D_t(a')(t) = D_t(a)(t)$ and by previous two inequalities we have that $D_t(a)(t) < D_t(a)(t)$. The second inequality just follows from the definition of minimal dance. To obtain the first inequality, observe that since for all $i < t$, $a(i) = a'(i)$, then either $Good(t, a) \subseteq Good(t, a')$ or $Good(t, a') \subseteq Good(t, a)$ and moreover $Good(t, a')$ and $Good(t, a)$ share the same minimal element $D_t(a)(t) = D_t(a')(t)$. Since $a'(t) \in Good(t, a')$, then $a'(t) \geq D_t(a)(t)$. The other case $a'(t) < a(t)$ is completely symmetric. □

Consider the following fact:

Fact

If a pigeon dance ends successfully on an $a \in A$, then the polynomial associated to the dance is in T (this is because we are moving to always strictly greater holes and therefore at the end the dummy hole 0 has disappeared).

Lemma 3.3 *If $d \leq \frac{\log n}{2}$ and $a \in A_d$, then there exists a dance on a if and only if there exists a dance on $\hat{a}$.*

Proof. If there is a dance for a then obviously there is a dance for $\hat{a}$, so one implication is easy. For the other implication, assume that the number of Q factors in x_a is different from 0, since otherwise there is nothing to prove. Assume that $\hat{a} = \{(i_1, j_1), \ldots, (i_k, j_k)\}$ and that l is the number of Q's factors in a, so that $k + l < d = \frac{\log n}{2}$. Note that after excuting the dance on $\hat{a}$ we remain with at least l holes unused. We will use these unused holes to define a dance on the whole a. That is, if the pigeon i is in $dom(\hat{a})$, then $a(i) = \hat{a}(i)$. If the pigeon $i \in dom(a) - dom(\hat{a})$, then we assign one of the unused l holes to move i in $a(i)$. Since these are completely new holes and since $|dom(a)| - |dom(\hat{a})| \leq l$, then the dance on a is well defined. □

We can now proceed to the definition of the basis B_d.

Definition 3.6

$$B_d = \{x_a : a \in A_d \textit{ there is a dance on } \hat{a}\}$$

It is easy to prove that the following monotonicity properties hold for B_d: (1) $B_{d-1} \subseteq B_d$; (2) $x_a \in B_{d-1}$ if and only if for all $i \notin dom(a)$, $x_a Q_i \in B_d$. In order to show that B_d is a basis for $Span(T_d)$ we need to define an order $\prec$ on polynomials in T_d. We will do it as in [8].

Definition 3.7 *Let x_a and x_b be two polynomials in T_d. then $x_a \prec x_b$ if and only if $deg(x_a) < deg(x_b)$, or if $deg(x_a) = deg(x_b)$, then for the largest pigeon i such that $a(i) \neq b(i)$, we have that $a(i) < b(i)$.*

Lemma 3.4 *B_d is a basis for $Span(T_d)$ for any $d \leq \frac{\log n}{2}$.*

Proof. Under the hypothesis that the degree d is less than $\frac{\log n}{2}$ we show: (1) that $|B_d| \leq |T_d|$ and (2) that any $x_a \in T_d$ can be expressed as a linear combination of elements of B_d, from which the Lemma follows. The first property follows because the minimal dance defines a 1-1 into mapping from B_d to T_d. More precisely, if $x_a \in B_d$ then we have a dance on $\hat{a}$ and since $d \leq \frac{\log n}{2}$, then by Lemma 3.3, there is dance on a and therefore by Lemma 3.1 there is a minimal dance on a that by Lemma 3.2 is a 1-1 mapping. Finally the observation in the previous Fact proves the first part. For the second part we work by induction on $\prec$. Assume that for all $x' \prec x_a$, $x' \in Span(B_d)$. We show that $x_a \in Span(B_d)$. If there is a dance on a then x_a is in B_d. Otherwise we show how to express x_a as a linear combination of the elements of B_d. Let P_t be the set of all possible correct first t steps of the dance on a. We prove that $x_a \in Span(B_d)$ iff $\sum_{b \in P_t} x_b \in Span(B_d)$ by induction on $t = 0, \ldots, n$. Since there is no dance on a, then $P_n = \emptyset$ and therefore the claim follows. The base of the induction $t = 0$ follows since $P_0 = a$. For the induction step observe that if $t \notin dom(a)$ then $P_t = P_{t-1}$ and so the claim follows by induction on t. Otherwise for any $b \in P_{t-1}$, x_b is of the form $x_{t,j} x_c$. We rewrite $x_{t,j}$ with respect to the relation Q_t, so that x_b can be rewritten as

$$(1) \quad x_c - x_c Q_t - \sum_{j' \neq j} x_c x_{t,j'}$$

equation 1 can be rewritten as:

$$x_c - x_c Q_t - \sum_{j'<j} x_c x_{t,j'} - \sum_{j'>j, j' \in Good(t,b)} x_c x_{t,j'} - \sum_{j'>j, j' \notin Good(t,b)} x_c x_{t,j'}$$

Each monomial in the last term is equal 0, therefore in $Span(B_d)$. The first three terms in the above sum are in $Span(B_b)$ by induction on $\prec$. The first by the base case of the definition of $\prec$. The second by the induction case of $\prec$ and by the monotonicity property of B_d. The third by (the second case of the definition) $\prec$. The fourth term corresponds exactly to all the possible correct first t steps of b. Therefore if we sum over all x_b for $b \in P_{t-1}$ we have that

$$\sum_{b \in P_t} x_b \in Span(B_d) \quad \text{iff} \quad \sum_{b \in P_{t-1}} x_b \in Span(B_d)$$

This concludes the proof of the Lemma. □

Theorem 3.1 *Any polynomial calculus refutation of $MPHP_n$ has degree not less than $\frac{\log n}{2}$.*

Proof. The proof is as in [8]. That is we prove by induction on the length of the proof that each polynomial derivable from the initial polynomials Q_i with at most degree d is a linear combinations of polynomials in $B_d - T_d$ (i.e a combination of the elements of B_d that are multiples of some axioms Q_i). Therefore since $1 \in T_d$ and it has a unique representation in each basis, we cannot derive the polynomial 1 with a proof of degree less than or equal to d.

Recall that we are considering refutation modulo the vector I. Therefore in the base case an axiom is always of the form Q_i for some $i \in [n]$, and the claim follows.

In the inductive step, if a line is inferred by the sum rule the result is immediate. For the case of product, say we have $\frac{x_a}{x_a x_{i,j}}$, with $|a| \le d-1$. We want to prove that $x_{i,j} x_a \in Span(B_d - T_d)$. By induction, x_a can be written as a sum of elements in $B_d - T_d$ (i.e. multiples of Q_i). Therefore distributing $x_{i,j}$ along elements of this sum, we can write $x_a x_{i,j}$ as a sum of multiples of Q_i. By the monotonicity properties of B_d, it is easy to see that this is a sum of scalar multiples of Q_i and therefore in $Span(B_d - T_d)$. □

4 Resolution lower bounds via degree lower bounds

RESOLUTION is a refutation proof system for formulas in CNF form based on the following *resolution rule*: $\frac{C \vee x \quad \bar{x} \vee D}{C \vee D}$ where if C and D have common literals, they appear only once in $C \vee D$. A resolution PROOF of a CNF formula F is a derivation of the empty clause from the clauses defining F, using the above inference rule. Following [3] the *width* $w(F)$ of a CNF formula F is defined to be the the number of literals of the largest clauses in F. The *width* $w(R)$ of a refutation R is defined as the size of the greatest clause appearing in R. The width $w(\vdash F)$ of refuting a formula F is defined as $\min_{R \vdash F} w(R)$

We prove that degree lower bounds imply width lower bounds as long as the initial polynomials of the PC proofs are obtained by the *standard mapping tr* of the initial clauses of the resolution proofs.

Lemma 4.1 *Given a set of unsatisfiable clauses F and a resolution refutation of F, there is a polynomial calculus refutation of $tr(F)$ of degree less than or equal to $w(\vdash F)+1$.*

Proof. Observe that given two clauses A and B, it is easy to obtain a PC derivation of

$$tr(A) = 0 \vdash tr(A)tr(B) = 0$$

with degree $w(A) + w(B)$.

We show that for each clause A in the resolution proof we find a PC derivation of $tr(A) = 0$ with degree at most the width of deriving A plus one. If A is an initial clause the result follows by definition of tr. Now assume that at a resolution step we are in the following situation

$$\frac{A \vee x \quad \bar{x} \vee B}{D}$$

We will simulate the resolution rule by a few PC steps. Assume that $A = A' \vee C$ and $B = B' \vee C$, i.e. C is the clause formed by the literals that belong to both A and B. $D = A' \vee B' \vee C$. By induction we have derived

$$tr(A)(1-x) = tr(A')tr(C)(1-x) = 0 \quad \text{and} \quad tr(B)x = tr(B')tr(C)x = 0$$

By the previous observation we can obtain the refutations

$$tr(A')tr(C)tr(B')(1-x) = 0 \quad \text{and} \quad tr(B')tr(C)tr(A')x = 0$$

An application of the sum rule gives $tr(D) = 0$. Note that the premises and conclusion of the resolution rule get translated by polynomials of the same degree as the width of the clauses. The steps added in the simulation can increment by 1 the degree respect to the width. □

As a consequence of the previous lemma and the width-size trade-off [3](see also last Section), a linear (in the number of variables) degree lower bound in polynomial calculus can give us an exponential lower bound in resolution size.

5 Upper bounds for the Modified Pigeonhole Principle

In this section we prove that the lower bound obtained in Section 3 is tight giving degree $O(\log n)$ PC refutations of Poly-$MPHP_n$. We use the simulation Lemma 4.1 of the previous section.

First consider the following definition and Lemma from [5].

Definition 5.1 ([5]) *Let $P(\vec{x})$ and $Q(\vec{y})$ be two sets of polynomials over a field F. Then P is (d_1, d_2)-reducible to Q if :*

1. *For ever y_i, there is a degree d_1 definition of y_i in terms of $\vec{x}$. That is for every i, there exist a degree d_1 polynomial r_i sucht that y_i can be viwed as defined by $y_i = r_i(\vec{x})$;*

2. *There exists a degree d_2 PC refutation of the polynomials $Q(\vec{r}(\vec{x}))$ from the polynomials $P(\vec{x})$*

Lemma 5.1 ([5]) *Suppose that $P(\vec{x})$ is (d_1, d_2)-reducible to $Q(\vec{y})$. Then if there is a degree d_3 PC refutation of $Q(\vec{y})$, then there is a degree* $\max(d_2, d_3 d_1)$ *PC refutation of $P(\vec{x})$.*

We'll use the previous Lemma to obtain PC refutations of Poly-$MPHP_n$ from PC refutations of the set of polynomials Poly-$3MPHP_n$ that we define next.

First we define the formula $3MPHP_n$ obtained from $MPHP_n$ as follows: for all $i \in [n]$, let $y_{i,0}, \ldots, y_{i,m}$ be $m+1$ new extension variables. Define $3MPHP$ by changing for all $i \in [n]$, the clauses $\bigvee_{i=1}^{m} x_{i,j}$ of $MPHP$ by the set of clauses:

$$\bar{y}_{i,0} \wedge \bigwedge_{j=1}^{m} (y_{i,j-1} \vee x_{i,j} \vee \bar{y}_{i,j}) \wedge y_{i,m}$$

where $y_{i,j}$ means that i goes to one of the first j holes.

Definition 5.2 `Poly`-$3MPHP_n$ *is obtained applying the mapping tr to the clauses defining* $3MPHP_n$.

Lemma 5.2 `Poly`-$MPHP_n$ *is (1,3)-reducible to* `Poly`-$3MPHP_n$

Proof. Define

$$y_{i,j} = 1 - \sum_{k>j}^{m} x_{i,k}$$

We prove that the initial polynomials of `Poly`-$3MPHP_n$ (with the y substituted by the definitions above) are derivable with a 3-degree polynomial calculus refutations from initial polynomials of `Poly`-$MPHP_n$.

Observe that $y_{i,0} = Q_i = 0$ and $y_{i,m} = 1$. So we can easily prove $y_{i,0} = 0$ and $1 - y_{i,m} = 0$. A generic initial polynomial of `Poly`-$3MPHP_n$ of the form

$$(1 - y_{i,j-1})(1 - x_{i,j})(y_{i,j}) \text{ for } 2 < j < n$$

is equivalent to

$$(\sum_{k>j-1}^{m} x_{i,k})(1 - x_{i,j})(\sum_{k>j}^{m} x_{i,k})$$

And using again $Q_i = 0$, the previous equation can be rewritten as

$$(1 - \sum_{k=1}^{j-1} x_{i,k})(1 - x_{i,j})(1 - \sum_{k>j}^{m} x_{i,k})$$

By simple calculations (using the initial axioms of `Poly`-$MPHP_n$) this is equal to

$$(1 - Q_j) + \sum_{k=1, k \neq j}^{n} x_{i,k} x_{i,j} + (1 - x_{i,j}) \sum_{k=1}^{j-1} x_{i,k} \; sum_{k=j+1}^{m} x_{i,k}$$

Using the initial axioms of `Poly`-$MPHP_n$ it is easy to see that each of the three terms of the above polynomial is equal to 0. □

In order to use lemma 4.1 to obtain $O(\log n)$ degree upper bounds for `Poly`-$3MPHP_n$ we need to prove $O(\log n)$ width upper bounds (in resolution) for $3MPHP_n$. We adapt a proof sketched by Goerdt from [7].

Lemma 5.3 *There are resolution refutations of* $3MPHP_n$ *of size* $n^{O(\log n)}$ *and width* $O(\log n)$.

Proof. First observe that there is an easy tree-like resolution refutation of size polynomial in n and of width $O(\log n)$ to obtain the initial clauses of $MPHP_n$ from the initial clauses of $3MPHP_n$. We show, following [7], that there is a resolution refutation of $MPHP_n$ of width $O(\log n)$ and size $n^{O(\log n)}$.

The proof goes by induction on $k = 4, \ldots, \log n$. The case $MPHP_4$ is easy and can be also found in [7]. Assume to have, by induction, a refutation of $MPHP_{2^k}$, for $k \geq 4$. We give a proof of $MPHP_{2^{k+1}}$.

Consider the intial clauses of $MPHP_{2^{k+1}}$ of the form

$$x_{i,1} \vee \ldots \vee x_{i,k+1} \qquad (1)$$

for all $i = 1, \ldots, 2^{k+1}$. Resolve all of these clauses in parallel with the clauses $\bar{x}_{i,k+1} \vee \bar{x}_{i',k+1}$ for all $i' \neq i$, such that i and i' are not $(k+1)$-compatibles. This leaves us with the following clauses

$$x_{i,1} \vee \ldots \vee x_{i,k} \vee \bar{x}_{i',k+1}$$

for all $i = 1, \ldots, 2^{k+1}$ and for all $i' \neq i$ not $(k+1)$-compatibles. Applying to these clauses the proof of $MPHP_{2^k}$ we have by induction, we produce the singleton clauses $\bar{x}_{i',k+1}$. Now we use these clauses and resolve with clauses in (1) to obtain for al $i = 1, \ldots, 2^k$ the clauses

$$x_{i,1} \vee \ldots \vee x_{i,k}$$

With another application of the proof of $MPHP_{2^k}$ applied to these clauses we obtain the empty clauses.

It is straigthforward to see that in the refutation we never use clauses of width greater than $O(\log n)$ and that the total number of clauses derived is at most quasipolynomial in n. □

Theorem 5.1 *There are $O(\log n)$ degree PC refutations of* `Poly`-$MPHP_n$.

Proof. Lemma 4.1 and the previous Lemma gives us $O(\log n)$ degree PC refutations of `Poly`-$3MPHP_n$. Then Lemma 5.2 implies the claim of the Theorem □

6 Discussion and Open Problems

Resolution and Polynomial Calculus: width, size and degree

Consider the following two Theorems proved respectively in [6] and [7].

Theorem 6.1 ([6]) *If a set of clauses F over n variables and of width at most k has a tree-like resolution refutation of size S, then the set of polynomials $tr(F)$ has a PC refutation of degree $k + O(\log S)$.*

Theorem 6.2 ([7]) *There are tree-like resolution refutations of $MPHP_n$ of size $n^{O(1)}$ and width $O(n)$.*

Following the strategy used in the previous section and the previos theorems we can obtain the same degree $O(\log n)$ upper bound for PC refutations of `Poly`-$MPHP_n$. The analysis of the Theorem 6.1 shows that the number of poynomials produced by the simulation is roughly of the order $n^{O(\log S)}$. Therefore using this theorem we cannot improve the $n^{O(\log n)}$ number of polynomials that also our simulation produces, given that we are using quasipolynomial size resolution refutation for $MPHP_n$.

Consider the following resuming table for $MPHP_n$ and `Poly`-$MPHP_n$:

System	size	width	degree
tree-like Res	$n^{O(1)}$	$O(n)$	$*$
dag-like Res	$n^{O(\log n)}$	$O(\log n)$	$*$
PC	$n^{O(\log n)}$	$*$	$\Theta(\log n)$

Observe that there seems to be a trade-off between size and width for Resolution refutations of $MPHP_n$. To reduce the width from $O(n)$ to $O(\log n)$ the size should increase considerably. We don't know, for the case of $MPHP_n$, if there exist refutations of $O(\log n)$ width and polynomial size. We suspect that such proofs don't exists; i.e. that we cannot always obtain proofs of optimal size and width at the same time. It could be interesting to study such a kind of questions also for other tautologies or even in a general setting analyzing the relationships between size, width in Resolution and degree in Polynomial Calculus.

Resolution: width optimality and proof searching

The previous question is also motivated by the problem of searching for proofs or refutations. Ben-Sasson and Wigderson in [3], proposed an algorithm for searching for Resolution proofs based on the width measure:

Algorithm BW

C:= Clauses of F
$w := 0$
While $\Box \notin C$ **Do**
    $w := w + 1$
    apply resolution rule to clauses in C to derive all possible clauses of width $\leq w$.
    Add the clauses obtained to C
End

The running time of this algorithm is $n^{O(w(\vdash F))}$ when it searches for refutations of a formula F over n variables. The relationship between size and width was studied by [3] in order to reduce the problem of size lower bounds to that of width lower bounds. They proved the following theorem.

Theorem 6.3 *([3]) Let F be any unsatisfiable CNF formula over n variables and let $S(\vdash F)$ be the size of the shortest resolution refutation of F. Then:*

$$S(\vdash F) \geq \exp\left(\Omega\left(\frac{(w(\vdash F) - w(F))^2}{n}\right)\right)$$

We proved that this trade-off is optimal ([4]). It would be interesting to find some unsatisafible formula F over $O(n^2)$ variables with polynomial size resolution refutations but requiring degree $\Omega(n)$ in Polynomial Calculus for some polynomial encoding. This result would show that there is a tautology provable fast in Resolution, but requiring high degree in Polynomial Calculus. A possible good candidate could be the formula GT_n used by [4] to prove the optimality of the size-width trade off.

Resolution and Polynomial Calculus: simulations

Recall the simulation Theorem of Clegg,Edmonds and Impagliazzo [6]:

Theorem 6.4 *([6]) If a set of clauses F over n variables and of width at most k, has a deag-like resolution refutation of size S, then the set of polynomials $tr(F)$ has PC refutation of degree at most $3\sqrt{n \log_e S} + k + 1$.*

The size-degree trade-off of the previous Theorem is optimal, since there is a trivial resolution proof of size $O(2^n)$ for the PHP_n^{n+1}, and Razborov in [9] showed that PHP_n^m require $\Omega(n)$ degree PC refutations for all m. It could be interesting to prove the optimality of the previous simulation also in the case when S is "small" (i.e. polynomial in the size of the formula). We suspect that this result could be true. A way to obtain it is proving a $\Omega(n)$ degree lower bound for the GT_n principle.

Observe that our Lemma 4.1 is better (in the sense that it gives a smaller degree PC refutations) than the corresponding simulation result (Theorem 6.4) of [6] in the case we have constant width polynomial Resolution refutations of formulas having initial clauses of constant size. Moreover it implies that under the tr translation the width based algorithm of [3] cannot be better than the Grobner basis algorithm of [6].

Finally notice that there is no simulation of Polynomial Calculus by Resolution. Therefore it would be also interesting to obtain the opposite direction of Lemma 4.1.

Acknowledgment

The authors would like to thank the anonymous referees of the PhD Thesis of N. Galesi for many comments about the preliminary version of this paper. These comments contributed to improve the quality of the presentation.

References

[1] M. Aleckhnovitch and A. Razborov. Lower bounds for the polynomial calculus: non binomial case. Manuscript, September 2000.

[2] E. Ben-Sasson and R. Impagliazzo. Random CNF are hard for the Polynomial Calculus. *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 412–421, 1999.

[3] E. Ben-Sasson and A. Wigderson. Short proofs are narrow - resolution made simple. To appear in Journal of the ACM.

[4] M.L. Bonet and N. Galesi. Optimality of Size-Width Trade-Off for Resolution In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 422–431, 1999. Journal version submitted to *Computational Complexity*

[5] S. Buss, D. Grigoriev, R. Impagliazzo, and T. Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 547–556, 1999.

[6] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 174–183, Philadelphia, Pennsylvania, 22–24 May 1996.

[7] A. Goerdt. Unrestricted resolution versus N-resolution. *Theoretical Computer Science*, 93:159–167, 1992.

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

[9] A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998.