A Lower Bound for the Pigeonhole Principle in Tree-like Resolution by Asymmetric Prover-Delayer Games

Olaf Beyersdorff [1] * Nicola Galesi [2] † Massimo Lauria [2]

[1] *Institut für Theoretische Informatik, Leibniz Universität Hannover, Germany*
[2] *Dipartimento di Informatica, Sapienza Università di Roma, Italy*

Abstract

In this note we show that the asymmetric Prover-Delayer game developed in [3] for Parameterized Resolution is also applicable to other tree-like proof systems. In particular, we use this asymmetric Prover-Delayer game to show a lower bound of the form $2^{\Omega(n \log n)}$ for the pigeonhole principle in tree-like Resolution. This gives a new and simpler proof of the same lower bound established by Dantchev and Riis [5].

1 Introduction

Proving lower bounds by games is a very fruitful technique in proof complexity [1, 8–10]. In particular, the Prover-Delayer game of Pudlák and Impagliazzo [10] is one of the canonical tools to study lower bounds in tree-like Resolution [2,10] and tree-like $Res(k)$ [6]. The Prover-Delayer game of Pudlák and Impagliazzo arises from the well-known fact [7] that a tree-like Resolution proof for a formula F can be viewed as a decision tree which solves the search problem of finding a clause of F falsified by a given assignment. In the game, Prover queries a variable and Delayer either gives it a value or leaves the decision to Prover and receives *one* point. The number of Delayer's points at the end of the game is then proportional to the height of the proof tree. It is easy to argue that showing lower bounds by this game only works if (the graph of) every tree-like Resolution refutation contains a balanced sub-tree as a minor, and the height of that sub-tree then gives the size lower bound.

In [3] we developed a new asymmetric Prover-Delayer game which extends the game of Pudlák and Impagliazzo to make it applicable to obtain lower bounds to tree-like proofs when the proof trees are very unbalanced. In [3] we used the new asymmetric game to obtain lower bounds in tree-like Parameterized Resolution, a proof system in the context of parameterized proof complexity recently introduced by Dantchev, Martin, and Szeider [4]. The lower bounds we obtain in [3] for tree-like Parameterized Resolution are of the form $\Omega(n^k)$ (n is the formula size and k the parameter), but the tree-like Parameterized Resolution refutations of the formulas in question only contain balanced sub-trees of height k.

The aim of this note is to show that the asymmetric Prover-Delayer game is also applicable to other (non-parameterized) tree-like proof systems. One of the best studied principles is the pigeonhole principle. Dantchev and Riis [5] show that the pigeonhole principle requires tree-like Resolution refutations of size roughly $n!$ while its tree-like Resolution proofs only contain

*This work was done while the first author was visiting Sapienza University of Rome under support of DFG grant KO 1053/5–2.

†Supported by grant "Limiti di compressione in combinatoria e complessità computazionale" by Sapienza University Rome.

balanced sub-trees of height n. Therefore the game of Pudlák and Impagliazzo only yields a $2^{\Omega(n)}$ lower bound which is weaker than the optimal bound $2^{\Omega(n \log n)}$ established by Dantchev and Riis. Here we provide a new and easier proof of this lower bound by our asymmetric Prover-Delayer game.

2 Preliminaries

A *literal* is a positive or negated propositional variable and a *clause* is a set of literals. A clause is interpreted as the disjunctions of its literals and a set of clauses as the conjunction of the clauses. Hence clause sets correspond to formulas in CNF. The *Resolution system* is a refutation system for the set of all unsatisfiable CNF. Resolution uses as its only rule the *Resolution rule*

$$\frac{\{x\} \cup C \qquad \{\neg x\} \cup D}{C \cup D}$$

for clauses C, D and a variable x. The aim in Resolution is to demonstrate unsatisfiability of a clause set by deriving the empty clause. If in a derivation every derived clause is used at most once as a prerequisite of the Resolution rule, then the derivation is called *tree-like*, otherwise it is *dag-like*. The *size* of a Resolution proof is the number of its clauses. Undoubtedly, Resolution is the most studied and best-understood propositional proof system (cf. [11]).

It is well known (cf. [7]) that a tree-like refutation of F can equivalently be described as a *boolean decision tree*. A boolean decision tree for F is a binary tree where inner nodes are labeled with variables from F and leafs are labeled with clauses from F. Each path in the tree corresponds to a partial assignment where a variable x gets value 0 or 1 according to whether the path branches left or right at the node labeled with x. The condition on the decision tree is that each path α must lead to a clause which is falsified by the assignment corresponding to α. Therefore, a boolean decision tree solves the *search problem* for F which, given an assignment α, asks for a clause from F falsified by α. It is easy to verify that each tree-like Resolution refutation of F yields a boolean decision tree for F and vice versa, where the size of the Resolution proof equals the number of nodes in the decision tree. In the sequel, we will therefore concentrate on boolean decision trees to prove our lower bound to tree-like Resolution.

3 Tree-like Lower Bounds via Asymmetric Prover-Delayer Games

We review the asymmetric Prover-Delayer game from [3]. Let F be a set of clauses in n variables $x_1, \dots, x_n$. In the asymmetric game, Prover and Delayer build a (partial) assignment to $x_1, \dots, x_n$. The game is over as soon as the partial assignment falsifies a clause from F. The game proceeds in rounds. In each round, Prover suggests a variable x_i, and Delayer either chooses a value 0 or 1 for x_i or leaves the choice to the Prover. In this last case, if the Prover sets the value, then the Delayer gets some points. The number of points Delayer earns depends on the variable x_i, the assignment α constructed so far in the game, and two functions $c_0(x_i, \alpha)$ and $c_1(x_i, \alpha)$. More precisely, the number of points that Delayer will get is

$$\begin{array}{cl} 0 & \text{if Delayer chooses the value,} \\ \log c_0(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 0, and} \\ \log c_1(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 1.} \end{array}$$

Moreover, the functions $c_0(x, \alpha)$ and $c_1(x, \alpha)$ are chosen in such a way that for each variable x and assignment α

$$\frac{1}{c_0(x, \alpha)} + \frac{1}{c_1(x, \alpha)} = 1 \tag{1}$$

holds. Let us call this game the (c_0, c_1)-game on F.

The connection of this game to size of proofs in tree-like Resolution is given by Theorem 1. The theorem is essentially contained in [3], but for completeness we include the full proof.

Theorem 1 ([3]). *Let F be unsatisfiable formula in CNF and let c_0 and c_1 be two functions satisfying (1) for all partial assignments α to the variables of F. If F has a tree-like Resolution refutation of size at most S, then the Delayer gets at most $\log S$ points in each (c_0, c_1)-game played on F.*

Proof. Let F be an unsatisfiable CNF in variables $x_1, \ldots, x_n$ and let Π be a tree-like Resolution refutation of F. Assume now that Prover and Delayer play a game on F where they successively construct an assignment α. Let α_i be the partial assignment constructed after i rounds of the game, i. e., α_i assigns i variables a value 0 or 1. By p_i we denote the number of points that Delayer has earned after i rounds, and by Π_{α_i} we denote the sub-tree of the decision tree of Π which has as its root the node reached in Π along the path specified by α_i.

We use induction on the number of rounds in the game to prove the following claim:

$$|\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} \text{ for any round } i.$$

To see that the theorem follows from this claim, let α be an assignment constructed during the game yielding p_α points to the Delayer. As a contradiction has been reached in the game, the size of Π_α is 1, and therefore by the inductive claim

$$1 \leq \frac{|\Pi|}{2^{p_\alpha}} \ ,$$

yielding $p_\alpha \leq \log |\Pi|$ as desired.

In the beginning of the game, Π_{α_0} is the full tree and the Delayer has 0 points. Therefore the claim holds.

For the inductive step, assume that the claim holds after i rounds and Prover asks for a value of the variable x in round $i+1$. If the Delayer chooses the value, then $p_{i+1} = p_i$ and hence

$$|\Pi_{\alpha_{i+1}}| \leq |\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} = \frac{|\Pi|}{2^{p_{i+1}}} \ .$$

If the Delayer defers the choice to the Prover, then the Prover uses the following strategy to set the value of x. Let $\alpha_i^{x=0}$ be the assignment extending α_i by setting x to 0, and let $\alpha_i^{x=1}$ be the assignment extending α_i by setting x to 1. Now, Prover sets $x = 0$ if $|\Pi_{\alpha_i^{x=0}}| \leq \frac{1}{c_0(x,\alpha_i)}|\Pi_{\alpha_i}|$, otherwise he sets $x = 1$. Because $\frac{1}{c_0(x,\alpha_i)} + \frac{1}{c_1(x,\alpha_i)} = 1$, we know that if Prover sets $x = 1$, then $|\Pi_{\alpha_i^{x=1}}| \leq \frac{1}{c_1(x,\alpha_i)}|\Pi_{\alpha_i}|$. Thus, if Prover's choice is $x = j$ with $j \in \{0, 1\}$, then we get

$$|\Pi_{\alpha_{i+1}}| = |\Pi_{\alpha_i^{x=j}}| \leq \frac{|\Pi_{\alpha_i}|}{c_j(x,\alpha_i)} \leq \frac{|\Pi|}{c_j(x,\alpha_i)2^{p_i}} = \frac{|\Pi|}{2^{p_i + \log c_j(x,\alpha_i)}} = \frac{|\Pi|}{2^{p_{i+1}}} \ .$$

This completes the proof of the induction. $\square$

As remarked in [3] we get the game of Pudlák and Impagliazzo [10] by setting $c_0(x, \alpha) = c_1(x, \alpha) = 2$ for all variables x and partial assignments α.

4 Tree-like Resolution Lower Bounds for the Pigeonhole Principle

The *weak pigeonhole principle* PHP_n^m with $m > n$ uses variables $x_{i,j}$ with $i \in [m]$ and $j \in [n]$, indicating that pigeon i goes into hole j. PHP_n^m consists of the clauses

$$\bigvee_{j\in[n]} x_{i,j} \quad \text{for all pigeons } i \in [m]$$

and $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ for all choices of distinct pigeons $i_1, i_2 \in [m]$ and holes $j \in [n]$. We prove that PHP_n^m is hard for tree-like Resolution. Showing the lower bound by the asymmetric game from the last section, requires a suitable choice of the functions c_0 and c_1 and then the definition of the Delayer-strategy for the (c_0, c_1)-game.

Theorem 2. *Any tree-like Resolution refutation of PHP_n^m has size $2^{\Omega(n \log n)}$.*

Proof. Let α be a partial assignment to the variables $\{ x_{i,j} \mid i \in [m], j \in [n] \}$. Let

$$p_i(\alpha) = |\{ j \in [n] \mid \alpha(x_{i,j}) = 0 \text{ and } \alpha(x_{i',j}) \neq 1 \text{ for all } i' \in [m] \}| \ .$$

Intuitively, $p_i(\alpha)$ corresponds to the number of holes which are still free but are explicitely excluded for pigeon i by α (we do not count the holes which are excluded because some other pigeon is sitting there). We define

$$c_0(x_{i,j}, \alpha) = \frac{\frac{n}{2} + 1 - p_i(\alpha)}{\frac{n}{2} - p_i(\alpha)} \quad \text{and} \quad c_1(x_{i,j}, \alpha) = \tfrac{n}{2} + 1 - p_i(\alpha) \ .$$

For simplicity we assume that n is divisible by 2. During the game it will never be the case that Prover gets the choice when $p_i(\alpha) \geq \frac{n}{2}$. Therefore the functions c_0 and c_1 are always greater than zero when the Delayer gets points, thus the score function is always well defined. Furthermore notice that this definition satisfies (1).

Let us try to provide an intuitive explanation of why we choose the functions c_0 and c_1 and thus the points for Delayer as above. As a first observation, Delayer always earns more when Prover is setting a variable $x_{i,j}$ to 1 instead of setting it to 0. This is intuitively correct as the amount of freedom for Delayer to continue the game is by far more diminished by sending pigeon i to some hole j than by just excluding that hole j for pigeon i. Thus, when Prover sets $x_{i,j}$ to 1, Delayer should earn a number of points approximately equal to the quantity of the information that is not anymore available for that pigeon i after the choice of Prover. How much is this information? We take as measure of information (approximately—hence the $\frac{n}{2}$) the number of possible holes where Prover can choose to send pigeon i and hence are still free for mapping that pigeon.

We now describe Delayer's strategy in a (c_0, c_1)-game played on PHP_n^m. If Prover asks for a value of $x_{i,j}$, then Delayer decides as follows:

set $\alpha(x_{i,j}) = 0$	if there exists $i' \in [m] \setminus \{i\}$ such that $\alpha(x_{i',j}) = 1$ or if there exists $j' \in [n] \setminus \{j\}$ such that $\alpha(x_{i,j'}) = 1$;
set $\alpha(x_{i,j}) = 1$	if $p_i(\alpha) \geq \frac{n}{2}$ and there is no $i' \in [m]$ with $\alpha(x_{i',j}) = 1$, and
let Prover decide	otherwise.

Intuitively, Delayer leaves the choice to Prover as long as pigeon i does not already sit in a hole, hole j is still free, and there are at most $\frac{n}{2}$ excluded free holes for pigeon i.

If Delayer uses this strategy, then the small clauses $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ from PHP^m_n will not be violated in the game. Therefore, a contradiction will always be reached on one of the big clauses $\bigvee_{j\in[n]} x_{i,j}$. Let us assume now that the game ends by violating $\bigvee_{j\in[n]} x_{i,j}$, i. e., for pigeon i all variables $x_{i,j}$ with $j \in [n]$ have been set to 0. As soon as the number $p_i(\alpha)$ of excluded free holes for pigeon i reaches the threshold $\frac{n}{2}$, Delayer will not leave the choice to Prover. Instead, Delayer will try to place pigeon i into some hole. If Delayer still answers 0 to $x_{i,j}$ even after $p_i(\alpha) > \frac{n}{2}$, it must be the case that some other pigeon already sits in hole j, i. e., for some $i' \neq i$, $\alpha(x_{i',j}) = 1$. Therefore, at the end of the game at least $\frac{n}{2}$ variables have been set to 1. W. l. o. g. we assume that these are the variables x_{i,j_i} for $i = 1, \ldots, \frac{n}{2}$.

Let us check how many points Delayer earns in this game. We calculate the points separately for each pigeon $i = 1, \ldots, \frac{n}{2}$ and distinguish two cases: whether x_{i,j_i} was set to 1 by Delayer or Prover. Let us first assume that Delayer sets the variable x_{i,j_i} to 1. Then pigeon i was not assigned to a hole yet and, moreover, there must be $\frac{n}{2}$ unoccupied holes which are already excluded for pigeon i by α, i. e., there is some $J \subseteq [n]$ with $|J| = \frac{n}{2}$, $\alpha(x_{i',j'}) \neq 1$ for $i' \in [m]$, $j' \in J$, and $\alpha(x_{i,j'}) = 0$ for all $j' \in J$. All of these 0's have been assigned by Prover, as Delayer has only assigned a 0 to $x_{i,j'}$ when some other pigeon was already sitting in hole j', and this is not the case for the holes from J (at the moment when Delayer assigns the 1 to x_{i,j_i}). Thus, before Delayer sets $\alpha(x_{i,j_i}) = 1$, she has already earned points for all $\frac{n}{2}$ variables $x_{i,j'}$, $j' \in J$, yielding

$$\sum_{p=0}^{\frac{n}{2}-1} \log \frac{\frac{n}{2}+1-p}{\frac{n}{2}-p} \;=\; \log \prod_{p=0}^{\frac{n}{2}-1} \frac{\frac{n}{2}+1-p}{\frac{n}{2}-p} \;=\; \log\left(\frac{n}{2}+1\right)$$

points for the Delayer. Let us note that because Delayer never allows a pigeon to go into more than one hole, she will really get the number of points calculated above for *every* of the variables which she set to 1.

If, conversely, Prover sets variable x_{i,j_i} to 1, then Delayer gets $\log(\frac{n}{2}+1-p_i(\alpha))$ points for this, but she also received points for the $p_i(\alpha)$ variables set to 0 before by Prover. Thus, in this case Delayer earns on pigeon i

$$\begin{aligned} & \log(\frac{n}{2}+1-p_i(\alpha)) + \sum_{p=0}^{p_i(\alpha)-1} \log \frac{\frac{n}{2}+1-p}{\frac{n}{2}-p} \\ = \; & \log(\frac{n}{2}+1-p_i(\alpha)) + \log \frac{\frac{n}{2}+1}{\frac{n}{2}-p_i(\alpha)+1} \\ = \; & \log\left(\frac{n}{2}+1\right) \end{aligned}$$

points. In total, Delayer gets at least

$$\frac{n}{2}\log\left(\frac{n}{2}+1\right)$$

points in the game. Applying Theorem 1, we obtain $2^{\frac{n}{2}\log(\frac{n}{2}+1)}$ as a lower bound to the size of each tree-like Resolution refutation of PHP^m_n. $\square$

By inspection of the above Delayer strategy it becomes clear that the lower bound from Theorem 2 also holds for the *functional pigeonhole principle* where in addition to the clauses from PHP^m_n we also include $\neg x_{i,j_1} \vee \neg x_{i,j_2}$ for all pigeons $i \in [m]$ and distinct holes $j_1, j_2 \in [n]$.

References

[1] A. Atserias and V. Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008.

[2] E. Ben-Sasson, R. Impagliazzo, and A. Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.

[3] O. Beyersdorff, N. Galesi, and M. Lauria. Hardness of parameterized resolution. Technical Report TR10-059, Electronic Colloquium on Computational Complexity, 2010.

[4] S. S. Dantchev, B. Martin, and S. Szeider. Parameterized proof complexity. In *Proc. 48th IEEE Symposium on the Foundations of Computer Science*, pages 150–160, 2007.

[5] S. S. Dantchev and S. Riis. Tree resolution proofs of the weak pigeon-hole principle. In *Proc. 16th Annual IEEE Conference on Computational Complexity*, pages 69–75, 2001.

[6] J. L. Esteban, N. Galesi, and J. Messner. On the complexity of resolution with bounded conjunctions. *Theoretical Computer Science*, 321(2–3):347–370, 2004.

[7] J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications.* Cambridge University Press, Cambridge, 1995.

[8] P. Pudlák. On the complexity of propositional calculus. In *Sets and Proofs, Invited papers from Logic Colloquium'97*, pages 197–218. Cambridge University Press, 1999.

[9] P. Pudlák and S. R. Buss. How to lie without being (easily) convicted and the length of proofs in propositional calculus. In *Proc. 8th Workshop on Computer Science Logic*, volume 933 of *Lecture Notes in Computer Science*, pages 151–162. Springer-Verlag, Berlin Heidelberg, 1994.

[10] P. Pudlák and R. Impagliazzo. A lower bound for DLL algorithms for SAT. In *Proc. 11th Symposium on Discrete Algorithms*, pages 128–136, 2000.

[11] N. Segerlind. The complexity of propositional proofs. *Bulletin of symbolic Logic*, 13(4):482–537, 2007.