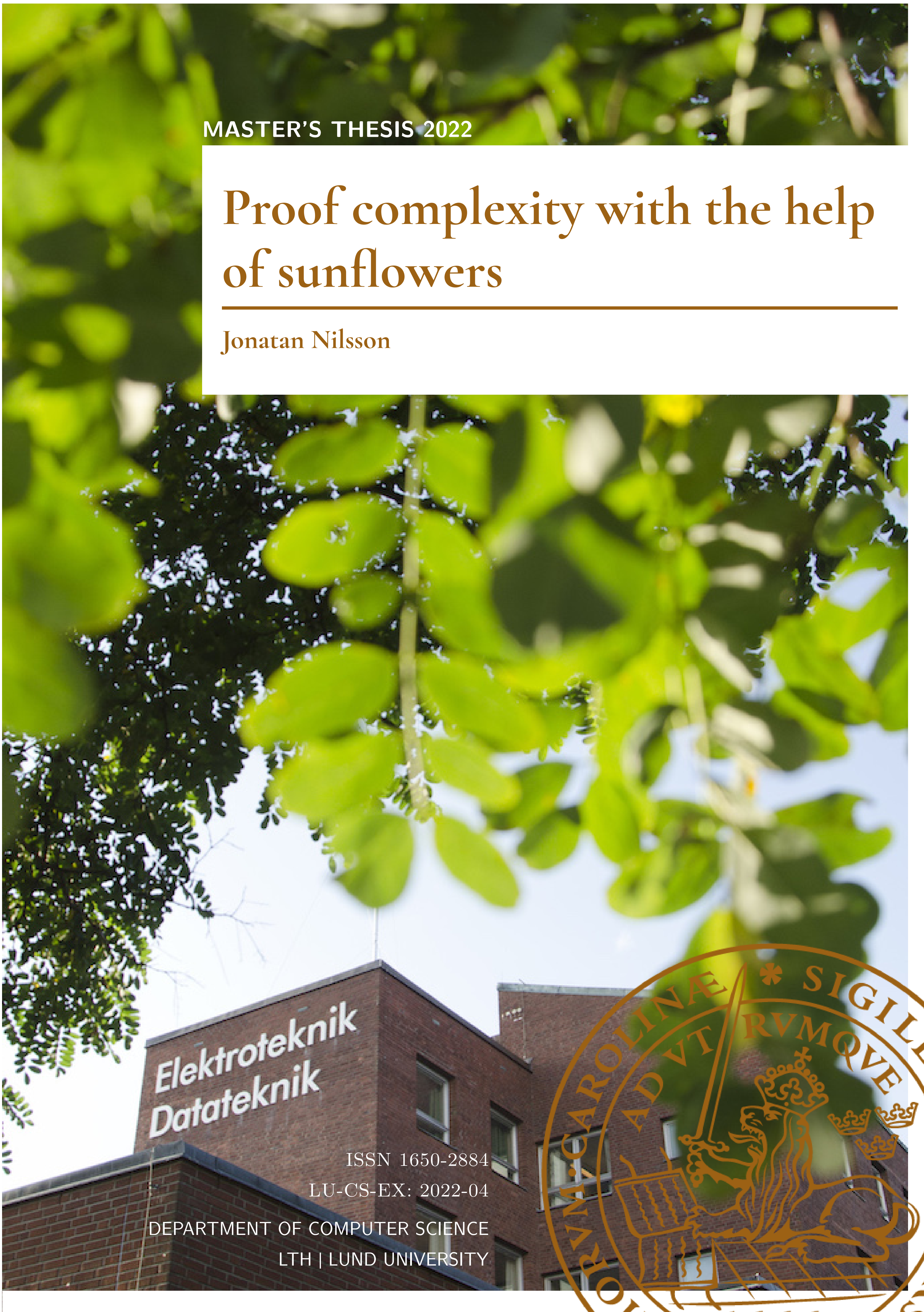
MASTER'S THESIS 2022

Proof complexity with the help of sunflowers

Jonatan Nilsson



ISSN 1650-2884
LU-CS-EX: 2022-04

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-04

Proof complexity with the help of sunflowers

Beviskomplexitet med hjälp av solrosor

Jonatan Nilsson

Proof complexity with the help of sunflowers

Jonatan Nilsson
nilsson.jonatan98@gmail.com

February 24, 2022

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Jakob Nordström, jakob.nordstrom@cs.lth.se

Examiner: Jonas Skeppstedt, jonas.skeppstedt@cs.lth.se

Abstract

Finding whether a boolean formula can be satisfied or not is a very fundamental problem. A related problem is to, for an unsatisfiable formula, find a proof that no solutions exist. In proof-complexity we study how large these proofs must be for different formulas. In this thesis we will look at two different types of proofs, resolution and cutting planes, with regards to their length and space. We will present a proof that for certain formulas there is a trade-off between length and space, i.e. all short proofs must use a lot of space and all space-efficient proofs must be long.

The aim of this thesis is to explore different possibilities to improve the trade-off result by using an area of combinatorics called sunflowers. We did not however manage to improve the trade-off compared to what was previously known.

Keywords: Proof complexity, sunflower, communication complexity, cutting planes, Dymond-Tompa

Contents

Chapter 1

Introduction

The problem of finding whether a boolean formula can be satisfied (SAT) is fundamental in computer science. As SAT is NP-complete no efficient algorithm is known. A related question is if, for an unsatisfiable formula, there exists small proofs showing that the formula is unsatisfiable. These proofs are called refutations. The field of proof-complexity is the field in which one studies these type of proofs.

Why is it interesting to study proof complexity? From a theory perspective, studying proof-complexity might help in understanding problems like P *vs* NP and NP *vs* $coNP$ since NP consists of the problems whose solutions can be proven to be solutions in polynomial time and $coNP$ consists of the problems whose non-solutions can be refuted in polynomial time.

Another reason to study proof complexity is to increase our understanding of the algorithms that underlie SAT-solvers. Even though SAT is a hard problem in theory it is often quite tractable in practice and SAT-solvers can often solve large instances fast. Algorithms that solve the SAT problem can be seen as searching for a proof whether the formula has a solution or is unsatisfiable. So if we can prove a lower bound for the length of refutations of a certain type (e.g. resolution, cutting planes) that would imply a lower bound for the running time of algorithms searching for this type of refutation. If we for example can prove that all resolution-refutations of a formula are very long then that would imply that the Davis-Putnam algorithm (which uses resolution [DP60]) has a large running time.

There is a rich literature in proof complexity on hardness results for different proof systems, showing that different families of formulas require long proofs or proofs of high space complexity. There have also been a number of papers establishing length-space trade-offs, meaning that there are formulas that have short proofs and also space-efficient proofs, but where every short proof has to use a lot of space and every space-efficient proof has to be very long. However, for cutting planes such trade-off results have had fairly weak parameters.

In the paper "How Limited Interaction Hinders Real Communication" [DRNV16], here denoted "Limited Interaction", such a length-space trade-off for cutting planes is proven. The proof goes via something called communication complexity and one important part of the proof is a "lifting" lemma that gives a connection between communication complexity and certain decision trees. This lifting lemma is the most difficult part of the proof in "Limited Interaction" and is also a bottleneck in getting stronger results. Recently a paper called "Lifting with Sunflowers" [LMM+20] came out which proves a different, but similar, lifting lemma with better parameters than the lemma in "Limited Interaction". The ideas in the paper "Lifting with Sunflowers" could possibly be used to improve the parameters in the "Limited Interaction"-paper and give stronger trade-off results for cutting planes. The problem is that the lemma in "Lifting with Sunflowers" is not general enough to work in the settings needed for the "Limited Interaction". In this thesis we will look at techniques used in "Lifting with Sunflowers" and see if it is possible to extend them to work in the setting needed for "Limited Interaction". This would give stronger trade-off results for cutting planes than what is currently known.

In Chapter 2 we go through the necessary background. Section 2.6 in this chapter contains the background needed to understand the "Lifting with Sunflowers" article, but it is not necessary to have read this section in order to follow Chapters 3 and 4. In Chapter 3 we present a proof of a length-space trade-off for cutting planes. This will be based on the proof in "Limited Interaction" and the overall structure will be the same, but details can of course be different. Potential ways of improving the length-space trade-off are discussed in Chapter 4 and in Chapter 5 we prove an alternative lifting lemma using the same techniques as "Lifting with Sunflowers". Section 5.4 is about the attempts to generalise this alternative lifting lemma. Lastly we will end the thesis with some final words in Chapter 6.

Chapter 2
Background

2.1 Basics and notation

The formulas we study will be in conjunctive normal form (CNF), i.e., a big *and* of clauses where each clause is an *or* of literals and a literal is a variable or a negated variable. A simple example of a CNF-formula would be

$$(x_1 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4)$$

As can be seen in the example we denote *or* with $\vee$, *and* with $\wedge$ and negation of x with $\overline{x}$.

In the following list let m and n be natural numbers and let A and B be sets.

- We define $[n]$ to be the set $\{1, 2, \ldots, n\}$.
- $A \times B$ is the Cartesian product, i.e., all pairs (a, b) with $a \in A$ and $b \in B$. In set builder notation this is written as $A \times B = \{(a, b) : a \in A, b \in B\}$.
- A^n is the set of all sequences of length n with elements from A.
- If $x \in [m]^n$ is a sequence and $i \in [n]$ is a number we let x_i denote the ith element in the sequence. Sometimes we will also use $x[i]$ to denote the ith element.
- If $x \in [m]^n$ is a sequence and $I \subseteq [n]$ is a subset of coordinates we let x_I (and $x[I]$) be the sequence of length $|I|$ with the elements from x at the coordinates specified by I.
- If $A = [m]^n$ is a set of sequences we let A_I be the set $\{x_I : x \in A\}$.
- We will use $\log$ to denote the logarithm in base 2 and $\ln$ to denote the natural logarithm.
- The symbol $\forall$ means *for all* and the symbol $\exists$ means *exists*.
- We will use the standard big O notations O, o, Ω, ω and Θ.

2.2 Proof systems

We start by a general description of proof systems and then look more closely at two concrete proof systems: resolution and cutting planes.

A proof will start with the constraints of the given CNF formula (these will be called the *axioms*). The proof will repeatedly derive new constraints from the axioms and previously derived constraints until a contradiction has been reached. In this project we will be interested in the *length* and the (*formula*) *space* complexity of a proof. To define length and space we can think of the proof having a “memory area” with constraints. Then the proof consists of a sequence of the following 3 operations:

1. “Axiom download”. Write one of the axioms to the memory space.
2. “Inference”. Derive a new constraint from the ones in memory.
3. “Erasure”. Delete one constraint from memory.

The length of the proof is the number of steps and the space of the proof is the maximum number of constraints in memory at any given time. There exists a trade-off between the length of the proof and the space of the proof. If we only want small length then the proof should not do any “Erasures” but if we want small space then we might want to do many “Erasures” (maybe erase a constraint and rederive it later).

2.2.1 Resolution

One of the most basic proof systems is resolution. In resolution the constraints are just clauses and we only have a single inference rule:

$$\frac{a \vee B \quad \overline{a} \vee C}{B \vee C}$$

Here a is a variable and B and C are the rest of the clauses. The notation should be understood as if it is known that $a \vee B$ is true and that $\overline{a} \vee C$ is true then we can derive that $B \vee C$ must be true.

Resolution is a *sound* proof system, meaning that if all premises are true then resolution will never derive any contradiction. Since the single inference rule is sound the proof system will also be sound.

Resolution is also a *refutation complete* proof system. This means that for any unsatisfiable formula there exists a resolution proof that derives a contradiction. See [Gal06] for a proof.

2.2.2 Cutting planes

The other proof system we will be looking at is cutting planes. In cutting planes each constraint is represented by a linear inequality $\sum_j a_j x_j \geq c$ where a_j and c are integers and x_j are 0 or 1.

How can then a clause be represented as a linear inequality? Negated variables are substituted with $\overline{x}_j = 1 - x_j$ and OR is substituted with addition. If we for example have a formula with four variables and it has the clause $x_1 \lor \overline{x}_3 \lor \overline{x}_4$ this clause would be transformed to

$$x_1 + (1 - x_3) + (1 - x_4) \geq 1 \iff 1x_1 + 0x_2 + (-1)x_3 + (-1)x_4 \geq -1$$

Cutting planes has three inference rules: addition, multiplication and division.

1. Addition
$$\frac{\sum_j a_j x_j \geq c \quad \sum_j b_j x_j \geq d}{\sum_j (a_j + b_j) x_j \geq (c + d)}$$
2. Multiplication
$$\frac{\sum_j a_j x_j \geq d}{\sum_j c a_j x_j \geq cd}$$
3. Division
$$\frac{\sum_j c a_j x_j \geq d}{\sum_j a_j x_j \geq \lceil d/c \rceil}$$

For multiplication and division c must be a positive integer. Since all three inference rules are sound it is easy to see that cutting planes is sound.

Cutting planes is stronger than resolution in the sense that the proofs are generally shorter. A resolution proof can thus never be shorter than the shortest cutting planes proof for a specific formula [CCT87]. Since resolution is refutation complete this shows that cutting planes also is refutation complete.

2.3 Communication complexity

In a communication problem we have two players Alice and Bob that would like to solve a search problem by communicating with each other. More formally a communication problem is a relation $S \subseteq X \times Y \times O$. Alice is given some value $x \in X$ and Bob is given $y \in Y$ and they want to find a solution $o \in O$ such that $(x, y, o) \in S$. We assume that the search problem is total, i.e. $\forall x \in X, y \in Y : \exists o \in O : (x, y, o) \in S$.

We will consider two variants of communication.

2.3.1 Deterministic communication

A (deterministic) communication protocol is a rooted binary tree in which every leaf is labeled by some $o \in O$ and each internal node specifies if Alice or Bob speaks. The two output edges correspond to if a 1 or a 0 is spoken. The cost of the communication protocol is the total number of bits sent, which is the same as the height of the communication protocol tree.

One round of communication consists of Alice sending Bob some number of bits and then Bob replying with some number of bits. Alice and Bob want to minimize both the cost and the number of rounds.

There is a simple protocol which only uses a single round: Alice sends over all her input and Bob computes the solution and replies with $o \in O$. Note that only bits are communicated, so Alice will have to encode her input x in some way.

2.3.2 Real communication

The other communication model is real communication. In real communication Alice and Bob communicate by comparing real numbers. The cost of a real communication protocol is the number of comparisons made by Alice and Bob. We also have the concept of rounds. In one round Alice and Bob can choose many numbers and compare them simultaneously, i.e. Alice chooses $a_1, ..a_k$ and Bob chooses $b_1, ..b_k$ and they get the result $a_1 \leq b_1, ..., a_k \leq b_k$.

One easy observation is that real communication can simulate deterministic communication. If for example Alice wants to send bits 10010 in the deterministic protocol Alice can choose the real numbers $1, 0, 0, 1, 0$ while Bob chooses $1/2$ in all positions.

2.4 Decision Trees

Let's say we have a search problem $S \subseteq \{0, 1\}^n \times O$ where we are given a bit-string, z, of length n and we want to find a solution $o \in O$ such that $(z, o) \in S$. A decision tree is a rooted tree in which each internal node is labeled by some z_i and has two outgoing edges corresponding to 0 and 1. If the input is $z_1, z_2, ...z_n$ the decision tree starts in the root and if the root is labeled with z_i the decision tree query the value of z_i (looks at z_i). If the value is 1 we move along the output edge corresponding to 1 and otherwise the 0 edge. The decision tree continues to query values and move down the tree until we reach a leaf node. Each leaf node is labeled by some $o \in O$.

We say that a decision tree solves a search problem if it for any input $z \in \{0, 1\}^n$ moves down to a leaf node which is labeled by a solution $o \in O$ to the input z. The cost of a decision tree is the depth of the tree. We define the decision tree complexity of a problem to be the cost of the smallest decision tree that solves the problem.

Intuitively the decision tree complexity is the number of bits in the input we need to look at to solve the problem. So the decision tree complexity can never be more than n.

2.5 Dymond-Tompa games

A Dymond-Tompa game is played on a directed acyclic graph, G. If a node has no incoming edges we call it a source and if it has no outgoing edges we call it a sink and we say that node a is a direct predecessor of node b if there is an edge from a to b. For simplicity we will assume that G has exactly one sink node, this does not really matter but makes it a bit cleaner.

There are two players in this game, the pebbler and the challenger. The pebbler will pebble nodes and the challenger will "challenge" pebbled nodes. The game is played in rounds where a round consists of the following steps

1. The pebbler places pebbles on a non-empty subset of the nodes that hasn't been pebbled yet.
2. The challenger either challenges one of the newly pebbled nodes or decides to challenge the currently challenged node again.

The game ends when, at the end of a round, all the direct predecessors of the challenged node are pebbled. At all times there is only one challenged node and the game starts with the

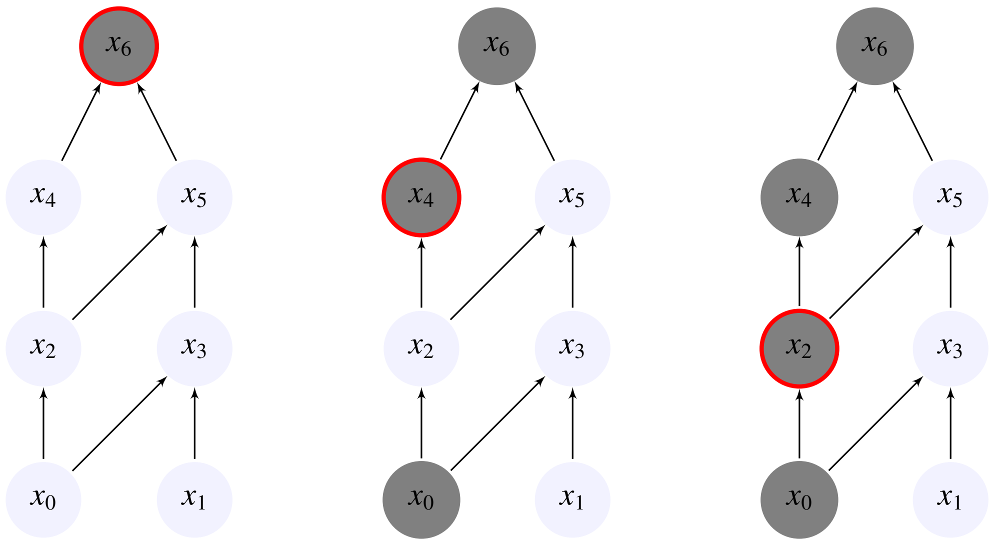


Figure 2.1: Example of a Dymond-Tompa game. Pebbled nodes have gray background and the challenged node have a red border. The leftmost picture is the starting position with the sink challenged and in the rightmost picture the game has ended since all direct predecessor to x_2 are pebbled.

sink being pebbled and challenged. The pebbler wants the game to end as quickly as possible and the challenger wants to play for as long as possible. We say a strategy for the pebbler is an r round strategy with cost c if the pebbler can guarantee that the game will end after at most r rounds by placing in total at most c pebbles no matter what the challenger does.

Let's consider the example in Figure 2.1. In the start of the game the sink is pebbled and challenged, the pebbler's first move is to place pebbles on node x_4 and x_0. The challenger then has the option to stay on x_6 or to move to x_4 or x_0, let's say the challenger decides to move to x_4. If the challenger instead had moved to x_0 the game would have ended. In the next round the pebbler chooses to pebble x_2, now the challenger's only options are to stay at x_4 or move to x_2, either way the game will end this round. This game had a cost of four since there were four nodes that were pebbled and two rounds. This example was just for illustration, both the pebbler and the challenger could have done better moves.

2.6 Sunflowers

Before we define what a sunflower is we will look at something called *min-entropy*. Let us define the min-entropy of a random variable $\mathbf{x} \in A$ as

$$\min_{x' \in A} \log(1/\Pr[\mathbf{x} = x'])$$

Now we will extend the notion of min-entropy to something called *blockwise min-entropy*. If A is a subset of B^n, for some set B, we can consider random variable $\mathbf{x} \in A$ and then look at the projection $\mathbf{x}_I$. The blockwise min-entropy is the normalised min-entropy of the random variable $\mathbf{x}$ projected to some set $I \subset [n]$. In other words we define the blockwise min-entropy of $\mathbf{x} \in A$ to be

$$\min_{I \subseteq [n], x'_I \in A_I} \frac{1}{|I|} \log\left(1/\Pr[\mathbf{x}_I = x'_I]\right)$$

In this thesis we will only consider uniform probability for $\mathbf{x} \in A$ and usually not specify the random variable but instead just say the blockwise min-entropy of A. Note that I is not allowed to be the empty set.

We say that $\mathcal{F}$ is a *set system* if all elements of $\mathcal{F}$ are subsets of some universe $\mathcal{U}$. If m, n are natural numbers then elements of the set $[mn]$ might be viewed as a pair (i, j) with $i \in [n]$ and $j \in [m]$. Let's call a set $\gamma \subseteq [mn]$ *block-respecting* if each number (i, j) in γ has a unique i. If $\mathcal{F}$ is a set system with universe $[mn]$ we say that $\mathcal{F}$ is block-respecting if every $\gamma \in \mathcal{F}$ is block-respecting.

A *sunflower* is a set system such that the intersection between all sets (the *core*) is the same as the intersection between any pair of sets. In other words $\mathcal{F}$ is a sunflower if and only if

$$\bigcap_{\gamma \in \mathcal{F}} \gamma = \gamma_i \cap \gamma_j$$

holds for all $i \neq j$. A *petal* of a sunflower is a set that can be written as $\gamma_i \setminus (\gamma_i \cap \gamma_j)$ for some $\gamma_i, \gamma_j \in \mathcal{F}$. The famous sunflower lemma by Erdős and Rado in 1960 [ER60] says roughly that a large set system must contain a large sunflower.

Lemma 2.1 (Sunflower Lemma)**.** *Let s and k be natural numbers and $\mathcal{F}$ be a set system of size at least $s!(k-1)^s$ over universe $\mathcal{U}$ such that any $\gamma \in \mathcal{F}$ satisfy $|\gamma| \leq s$ then $\mathcal{F}$ contains a sunflower with k petals.*

We will actually not use sunflowers as defined above, but instead use a slightly different version. If $\mathcal{F}$ is a set system over $\mathcal{U}$ and $0 \leq \kappa \leq 1$ we say that $\mathcal{F}$ is *κ-satisfying* if

$$\Pr[\forall \gamma \in \mathcal{F} : \gamma \nsubseteq \mathbf{U}] < \kappa$$

holds. The probability is taken over the uniformly random variable $\mathbf{U} \subseteq \mathcal{U}$.

We will now state a version of the sunflower lemma using blockwise min-entropy together with this κ-satisfying sunflower definition. The lemma comes from a recent result by Alweiss et al. [ALWZ20], but the formulation of the lemma is taken from "Lifting with Sunflowers" [LMM+20].

Lemma 2.2 (Blockwise Robust Sunflower Lemma)**.** *There exists an absolute constant K such that the following holds: let $s, m, n \in \mathbb{N}$ and $0 \leq \kappa \leq 1$. Let F be a block-respecting set system with universe $[mn]$ such that*

1. *$|\gamma| \leq s$ for all $\gamma \in F$*

2. *F has blockwise min-entropy at least $\log(K \log(s/\kappa))$.*

Then F is κ-satisfying.

Chapter 3
Proof trade-offs

In this chapter we will prove the following theorem

Theorem 3.1 (Main theorem)**.** *There exist a family of CNF-formulas, with $O(N)$ clauses, such that no cutting planes-refutation with space $O(N^{1/11})$ and length $O(N^{\log N})$ exists.*

3.1 Overview of proof

We start with an overview of the proof of 3.1. The overall structure will be the same as in the "How Limited Interaction Hinders Real Communication"-article [DRNV16]. The proof will be given in multiple steps.

1. Assume there is a refutation for the CNF-formula $Lift_\ell(Peb_G)$ with small space and length.
2. Use the refutation to create a communication protocol for $Search(Lift_\ell(Peb_G))$ that has low cost and is round-efficient.
3. Use the communication protocol to create a round-efficient parallel decision tree for $Search(Peb_G)$ with low cost.
4. Use the parallel decision tree to create a round-efficient strategy for the pebbler in the Dymond-Tompa game on G with low cost.
5. Finally, we will prove that such a strategy cannot exist and conclude that no short and round-efficient refutation for $Lift_\ell(Peb_G)$ exists.

Since we are interested in the complexity of cutting planes we will consider a whole family of CNF-formulas and a whole family of graphs rather than just a single CNF-formula and a single graph.

In Section 3.2 we will define CNF-formulas $\boldsymbol{Peb_G}$ and $\boldsymbol{Lift_\ell(Peb_G)}$. The following sections each give one step of the proof and in Section 3.7 we complete the proof by putting all parts together.

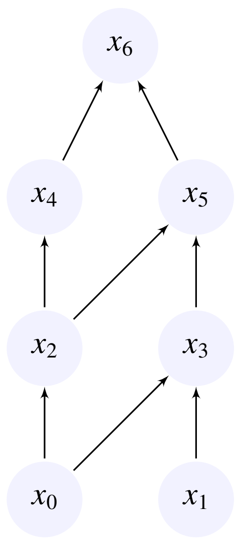
3.2 Hard formula

In this section we will define the CNF-formula $\boldsymbol{Lift_\ell(Peb_G)}$ which will be used as an example of an unsatisfiable CNF-formula for which no cutting planes-refutation with small space and length exists.

3.2.1 Pebbling contradiction

We start by defining the pebbling contradiction $\boldsymbol{Peb_G}$ for a directed acyclic graph (DAG), $\boldsymbol{G}$. Given a DAG $\boldsymbol{G}$ we create a variable x_v for each node v in $\boldsymbol{G}$. For each source node, v, we add the clause (*source axiom*) x_v, for each non-source node, v, with direct predecessors $u_1, \ldots, u_k$ we add the clause (*pebbling axiom*) $\overline{x}_{u_1} \lor \ldots \lor \overline{x}_{u_k} \lor x_v$. Finally for each sink node, v, we add the clause (*sink axiom*) $\overline{x}_v$. We note that the formula $\boldsymbol{Peb_G}$ will be unsatisfiable. Informal argument: All sources must be true and the pebbling axiom will progressively make all other nodes be true but the sink must be false according to the sink axiom.

See Figure 3.1 for an example of a pebbling contradiction.



$$\left.\begin{array}{l} x_0 \\ x_1 \end{array}\right\}\text{Source axiom}$$
$$\left.\begin{array}{l} \overline{x}_0 \lor x_2 \\ \overline{x}_0 \lor \overline{x}_1 \lor x_3 \\ \overline{x}_2 \lor x_4 \\ \overline{x}_2 \lor \overline{x}_3 \lor x_5 \\ \overline{x}_4 \lor \overline{x}_5 \lor x_6 \end{array}\right\}\text{Pebbling axiom}$$
$$\left.\begin{array}{l} \overline{x}_6 \end{array}\right\}\text{Sink axiom}$$

Figure 3.1: Example of a pebbling contradiction.

3.2.2 Lifted formulas

Next we will describe a way to "lift" a CNF-formula.

Definition 3.1 (Lifted formula)**.** Given a CNF formula, $\boldsymbol{F}$, over $\boldsymbol{n}$ variables z_i and a positive integer ℓ. We will define the lifted formula of lift-length ℓ, $\boldsymbol{Lift_\ell(F)}$. For each variable z_i in $\boldsymbol{F}$ we create ℓ *selector variables* $x_{i,j}$ and ℓ *main variables* $y_{i,j}$. We then add clauses to $\boldsymbol{Lift_\ell(F)}$:

- For each $i \in [n]$ we require that at least one $x_{i,j}$ is true.

$$x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,\ell}$$

- For every clause $z_{i_1} \vee z_{i_2} \vee \ldots \vee z_{i_s} \vee \overline{z}_{i_{s+1}} \vee \ldots \vee \overline{z}_{i_t}$ in F and every tuple $(j_1, \ldots, j_t) \in [\ell]^t$ we create a main clause

$$(x_{i_1,j_1} \rightarrow y_{i_1,j_1}) \vee \cdots \vee (x_{i_t,j_t} \rightarrow \overline{y}_{i_t,j_t})$$

The notation $x \rightarrow y$ means that x implies y which is exactly $\overline{x} \vee y$. If F has m clauses and width k then $Lift_\ell(F)$ is an unsatisfiable CNF formula over $2\ell n$ variables with $m\ell^k + n$ clauses and width $\max(\ell, 2k)$.

3.3 Proofs to communication

In this section we prove the following two lemmas.

Lemma 3.1. *If an unsatisfiable sat formula, F, has a resolution refutation of size L and space S then there exists a deterministic communication protocol for the problem $Search(F)$ of cost $O(S\log(L))$ that uses $O(\log(L))$ rounds.*

And a variant for cutting planes proofs.

Lemma 3.2. *If an unsatisfiable sat formula, F, has a cutting planes refutation of size L and space S then there exists a real communication protocol for the problem $Search(F)$ of cost $O(S\log(L))$ that uses $O(\log(L))$ rounds.*

3.3.1 Defining $Search(F)$

We start with defining the falsified clause search problem.

Definition 3.2 (Falsified clause search problem)**.** Given an unsatisfiable formula, F, and an assignment α to the variables in F find a clause in F that is falsified by α. We denote this problem $Search(F)$.

$Search(F)$ is a search problem but it is not a communication problem. To turn $Search(F)$ into a communication protocol we give the value of some variables to Alice and the value for the rest of the variables to Bob. Note that we do not specify how we partition the variables between Alice and Bob, the proof will work for any partition. Now the goal for Alice and Bob is to communicate to find a falsified clause. Next we will show how one can, from a given refutation, construct a protocol for the communication problem with the right parameters.

3.3.2 Constructing the protocol

Given a resolution proof we can think of it as being presented on a blackboard. In the beginning there is no falsified constraint on the board (since there are no constraints at all). The proof ends when a contradiction has been derived. In at least one step the proof has gone from having no falsified constraints on the blackboard to having one. Since the proof system is sound this step must have been an "Axiom download" step.

The goal for Alice and Bob will be to find this "bad" step in which the proof downloads the falsified constraint. They will do this in a binary search fashion. By starting in the middle and evaluating the configuration to see if there is any falsified constraint. If the middle configuration has a falsified constraint then it must be that the falsified constraint has already been downloaded so we continue with the first half of the proof. See Algorithm 1 for pseudocode.

Algorithm 1 Constructing communication protocol by binary search over proof.

```
$low = 1, high = L$
while $low + 1 < high$ do
    $mid = \lfloor (low + high)/2 \rfloor$
    Evaluate configuration $mid$ in the proof.
    if configuration has falsified constraint then
        $high = mid$
    else
        $low = mid$
    end if
end while
return Constraint downloaded between $low$ and $high$.
```

Let's take a closer look at the evaluation of a configuration. How can Alice and Bob figure out if a configuration contains a falsified constraint? So far our construction works the same way for both resolution and cutting planes, but here it will be slightly different.

For resolution each constraint is a clause of the form $x_1 \lor \ldots \lor x_k \lor y_l \lor \ldots \lor y_m$ where Alice knows the value of all x_i and Bob knows the value of each y_j. Alice can send over the value of $x_1 \lor \ldots \lor x_k$ for each clause in the configuration. Bob can now see if there is any falsified clauses and answers with 1 if there is and 0 otherwise. To evaluate a configuration Alice needs to send 1 bit of information per clause and Bob only needs to answer with 1 bit. Since there is at most S clauses in the configuration the cost will be at most $S + 1$ and only a single round.

For cutting planes each constraint will be of the form $\sum_i a_i x_i + \sum_j b_j y_j \leq c$. Here a_i, b_j and c are just the coefficient of the inequality and x_i are the variables known by Alice and y_j are the variables known by Bob. If Alice and Bob would use deterministic communication it could possibly take a lot of communication to figure out if any inequality is falsified. Instead, Alice and Bob can use real communication where they choose one real number each. If Alice chooses the real number $\sum_i a_i x_i$ and Bob chooses the number $c - \sum_j b_j y_j$ they will know if the constraint is falsified. There will be one comparison for each inequality and since there is at most S inequalities the cost will be S. It will only take one round since we can do all comparisons in parallel.

Due to the binary search we will evaluate roughly $\log(L)$ configurations for both resolution and cutting planes hence the cost will be $O(S \log(L))$ and $O(\log(L))$ number of rounds.

3.3.3 $Search(Lift_\ell(F))$ and $Lift_\ell(Search(F))$

In this section we have defined and used the communication problem $Search(F)$ but in the next section we will use the communication problem $Lift_\ell(Search(F))$. Here we will define $Lift_\ell(Search(F))$ and talk about its relation to $Search(Lift_\ell(F))$.

To recap: F is an unsatisfiable CNF-formula over n variables and $Lift_\ell(F)$ is an unsatisfiable CNF-formula over $2\ell n$ variables. $Search(F)$ and $Search(Lift_\ell(F))$ are the falsified clause search problem for the CNF formulas F and $Lift_\ell(F)$, respectively. To turn $Search(F)$ into a communication problem we gave the values of some variables to Alice and the rest to Bob. When we turn $Search(Lift_\ell(F))$ into a communication problem we will do the choice of giving Alice all the values of the selector variables and Bob all the values of the main variables. So Alice and Bob want to find a clause from the CNF-formula $Lift_\ell(F)$ and Alice is given the values of the ℓn selector variables and Bob is given the values of the ℓn main variables.

In the communication problem $Lift_\ell(Search(F))$ the goal for Alice and Bob is to find a falsified clause from the CNF-formula F (instead of $Lift_\ell(F)$). For each variable z_i in F Alice is given a number x_i between 1 and ℓ, and Bob is given a bit string y_i of length ℓ. The value of z_i is then the x_ith bit in Bob's bit string y_i.

If there exists a protocol for the communication problem $Search(Lift_\ell(F))$ then there also exists a protocol for the communication problem $Lift_\ell(Search(F))$ with the same parameters. This is true since $Lift_\ell(Search(F))$ is actually equivalent to the special case of $Search(Lift_\ell(F))$ when exactly one selector variable $x_{i,j}$ is true for each variable z_i.

3.4 Communication to Decision Trees

In this section we prove

Lemma 3.3. *Let F be an unsatisfiable CNF-formula over n variables and let $\ell = n^{3+\epsilon}$ for some $\epsilon > 0$. If there exists a (deterministic/real) communication protocol for $Lift_\ell(Search(F))$ that uses r rounds with cost $c\log(\ell)$ then there exists a r-round parallel decision tree with cost $O(c)$.*

This is a long and complicated proof so we will first do an overview of it before we dive in.

3.4.1 Overview

Recall how the communication protocol works: Alice will be given a sequence of length n of integers modulo ℓ and Bob will be given a sequence containing n bit strings of length ℓ. We will call Alice's input $x \in [\ell]^n$ and Bob's input $y \in (\{0,1\}^\ell)^n$. For Bob's input we might look at it like n bit strings of length ℓ or one big bit string of length ℓn and we will shift perspective to whatever is most convenient at a given moment. As we know Alice and Bob communicates to find a falsified clause of F. We denote the variables of F as $z_1, z_2, \ldots, z_n$ and we let z be just the bit string of $z_1 z_2 \ldots z_n$ concatenated. The x_ith bit in the bit string y_i will be denoted by $y_i[x_i]$.

We want to show that there exists a decision tree for the falsified clause search problem, $Search(F)$, given that there exists a communication protocol for $Lift_\ell(Search(F))$. We will construct a decision tree that is capable of "simulating" the communication protocol. In other words we will start in the root of the communication protocol tree and progressively move down the protocol tree and finally let the decision tree answer the same clause as the protocol tree at the leaf. To know which child to move to in the protocol tree we will have the decision tree query the values of z. As we move down the protocol tree we call a possible input to the communication problem, $(x, y) \in [\ell]^n \times \{0,1\}^{n\ell}$ *compatible* if it satisfies

1. If Alice is given x and Bob is given y they will move down the protocol tree in the same way as we have done so far.

2. (x, y) is consistent with the values queried so far by the decision tree. In other words if $y_i[x_i] = z_i$ for all coordinates $i \in [n]$ that has been queried.

The goal will be to show that, when we reach a leaf of the protocol tree, there exists a compatible input with $y_i[x_i] = z_i$ for *all* $i \in [n]$. If we manage to do this the correctness of the protocol tree will imply that the decision tree is correct.

To show that there exists such an input we will maintain a rectangle $R = A \times B$ of compatible inputs where $A \subseteq [\ell]^n$ is a set of possible inputs for Alice and $B \subseteq \{0,1\}^{n\ell}$ is a set of possible inputs for Bob.

3.4.2 Notation and definitions

The set of coordinates we have not queried will be called $I \subseteq [n]$. Let us call the current node in the protocol tree v and let $R_v \subseteq [\ell]^n \times \{0,1\}^{n\ell}$ be all inputs that reach v. We let A_v denote Alice's inputs and B_v denote Bob's inputs such that $R_v = A_v \times B_v$. If v is a node in the communication protocol tree we say that v_l is the left child and v_r is the right child. The amount of bits sent by Alice and Bob in the simulation will be denoted as $cost_A$ and $cost_B$, respectively. Since the communication protocol has cost $c \log \ell$ it is clear that $cost_A + cost_B \leq c \log \ell$. In the beginning we will have $I = [n]$, $A = [\ell]^n$, $B = \{0,1\}^{n\ell}$, $cost_A = 0$, $cost_B = 0$, and $v = root$.

It will be helpful for us to view the set A_I as a graph. We define $\mathrm{Graph}_i(A_I)$ to be a bipartite graph in which the left nodes correspond to the value of $x_i \in [\ell]$ and the right nodes correspond to the value of $x_{I\setminus\{i\}} \in [\ell]^{|I|-1}$. So $\mathrm{Graph}_i(A_I)$ has ℓ left nodes and $\ell^{|I|-1}$ right nodes. There will be an edge in $\mathrm{Graph}_i(A_I)$ between a left node and a right node if and only if there exists an $x \in A_I$ such that x_i corresponds to the left node and $x_{I\setminus\{i\}}$ corresponds to the right node. Similarly, we define $\mathrm{Graph}_i(B_I)$ to be a bipartite graph with left nodes corresponding to value of $y_i \in \{0,1\}^\ell$ and right nodes corresponding to value of $y_{I\setminus\{i\}} \in \{0,1\}^{\ell(|I|-1)}$.

Now it is possible to define two useful measures using this graph perspective. $\mathrm{AvgDeg}_i(A_I)$ is the average degree of the right nodes (of $\mathrm{Graph}_i(A_I)$) that have non-zero degree. It should be clear from the definition that the number of edges in $\mathrm{Graph}_i(A_I)$ are $|A_I|$ and the number of right nodes with non-zero degree are $|A_{I\setminus\{i\}}|$ so the following holds: $\mathrm{AvgDeg}_i(A_I) = |A_I|/|A_{I\setminus\{i\}}|$. Apart from AvgDeg we also define $\mathrm{MinDeg}_i(A_I)$ to be the minimum degree of the right nodes of $\mathrm{Graph}_i(A_I)$ that have non-zero degree. We say that A_I is *thick* if $\mathrm{MinDeg}_i(A_I) \geq \ell^\mu$ (μ will be defined shortly) holds for all $i \in I$.

To measure how large A is it will sometimes be convenient to use *density loss* which is defined as $D_\infty(A_I) = |I| \log \ell - \log |A_I|$. Similarly, we define density loss for B as $D_\infty(B_I) = \ell|I| - \log |B_I|$. Note that density loss is always non-negative and that A and B are non-empty if and only if $D_\infty(A_I) < \infty$ and $D_\infty(B_I) < \infty$, respectively.

When we construct our decision tree we will need a way to denote those input to Bob that can, in a certain sense, be completed to any value of z. This is made formal in our definition of Complete.

Definition 3.3 (Complete(B, I, U, i))**.** Let B be a set of possible inputs for Bob, $I \subseteq [n]$, $U \subseteq [\ell]$ and $i \in I$. Now Complete(B, I, U, i) are those $y_{I\setminus\{i\}} \in B_{I\setminus\{i\}}$ such that for both $b = 0$ and $b = 1$ there exists a $y \in B$ such that for all $u \in U$ we have $y_i[u] = b$.

Lastly we will define a number of parameters that will be used throughout the proof of lemma 3.3. As stated in the lemma we have $\ell = n^{3+\epsilon}$. Let us define $\gamma = 1/(3+\epsilon)$ so that $\ell^\gamma = n$. We will define $\mu = 2/3$, this is the μ that is used in definition of thickness. Furthermore we define $0 < \lambda < 1$ and $\delta = 2/3$ such that the following holds:

1. $\mu + \gamma < \lambda$
2. $\delta + \mu - 1 > \gamma$
3. $\delta + \gamma < 1$

3.4.3 Deterministic communication

We will first prove Lemma 3.3 for deterministic communication. Let's state one important claim that will be used when we construct our decision tree. The proof of the Projection Claim will be explained further down.

Claim 3.1 (Projection Claim)**.** *If A_I thick and $D_\infty(B_I) \leq 2n \log \ell$. Then there exists a $U \subseteq [\ell]$ such that*

1. $\{x_{I\setminus\{i\}} : x \in A, x_i \in U\}$ *is thick.*
2. $D_\infty(\{x_{I\setminus\{i\}} : x \in A, x_i \in U\}) \leq D_\infty(A_I) - \log \ell + \log \mathrm{AvgDeg}_i(A_I)$
3. $D_\infty(\mathrm{Complete}(B, I, U, i)) \leq D_\infty(B_I) + 1$

In Algorithm 2 we formalize how to move down the protocol tree and when the decision tree should query.

In line 6 we use the Projection Claim we defined above and in line 7 we update A, B and I in such a way that the new value of A_I becomes $\{x_{I\setminus\{i\}} : x \in A, x_i \in U_i\}$ and the new value of B_I becomes Complete(B, I, U, i). This means that Projection Claim says that after line 7 A_I is thick and we have some bounds on the density loss for both A_I and B_I. Note that $A \times B$ will be subset of R_v since A and B are never increased and every time we change v we intersect A and B with A_v and B_v, respectively.

In order to prove correctness of Algorithm 2 we will maintain four invariants during the simulation.

1. A_I is thick
2. $y_i[x_i] = z_i$ for all $x \in A, y \in B, i \in [n] \setminus I$
3. $D_\infty(A_I) \leq 2cost_A - (1 - \lambda)(n - |I|) \log \ell$

Algorithm 2 Parallel decision tree simulates a deterministic communication protocol.

```
 1: $v = root, R = A \times B = [\ell]^n \times \{0,1\}^{n\ell}, I = [n]$
 2: while $v$ is not a leaf do
 3:     $I_{old} = I$
 4:     while Alice speaks at $v$ do
 5:         while $\exists i \in I$ such that $\mathrm{AvgDeg}_i(A_I) < \ell^\lambda$ do
 6:             Use Projection Claim to get $U$
 7:             $A = \{x \in A : x_i \in U\}, B = \{y \in B : y_{I\setminus\{i\}} \in \mathrm{Complete}(B, I, U, i)\}, I = I \setminus \{i\}$
 8:         end while
 9:         $v = v_l$ if $|(A \cap A_{v_l})_I| > |(A \cap A_{v_r})_I|$ else $v_r$
10:         $A = A \cap A_v$
11:         Make $A_I$ thick
12:     end while
13:     Query $z_i$ for $i \in I_{old} \setminus I$
14:     for $i \in I_{old} \setminus I$ do
15:         $B = \{y \in B : y_i[x_i] = z_i \ \forall x \in A\}$
16:     end for
17:     while Bob speaks at $v$ do
18:         $v = v_l$ if $|(B \cap B_{v_l})_I| > |(B \cap B_{v_r})_I|$ else $v_r$
19:         $B = B \cap B_v$
20:     end while
21: end while
22: return Same clause as the communication protocol.
```

4. $D_\infty(B_I) \le cost_B + n - |I|$

Let us go directly to the proof of the invariants. It is easy to check that all four invariants hold in the beginning of the simulation: $\mathrm{MinDeg}_i(A_I) = \ell$ for all i so 1 holds, 2 holds since $I = [n]$, left and right-hand side are all zero in both invariant 3 and 4.

For invariant 1 we note that A and I are only updated in line 7 and lines 10-11. For line 7 the Projection Claim directly gives us that A_I is thick. Note that we are allowed to use the Projection Claim in line 6 since A_I is thick by invariant 1 and invariant 4 gives $D_\infty(B_I) \le cost_B + n - |I| \le 2n \log \ell$. After line 10 A_I might not be thick anymore, but in line 11 we make A_I thick. We delay the exact description of how we make A_I thick a little.

Every time we remove a coordinate, i, from I we will later in line 15 restrict B in such a way that invariant 2 holds. Note that invariant 2 does not necessarily hold when we are between line 7 and line 15 in our execution, but we actually only need it to hold when we have finished the simulation.

Invariant 3 requires a bit more work. We need to deal with line 7 and lines 10-11. Let A_I^{new} be value of A_I after we have executed line 7 and A_I^{old} be A_I before we execute line 7 and similarly for I. Then by the Projection Claim and the fact that $\mathrm{AvgDeg}_i(A_I^{old}) \le \ell^\lambda$ we have

$$\begin{aligned} D_\infty(A_I^{new}) &\le D_\infty(A_I^{old}) - \log \ell + \log \mathrm{AvgDeg}_i(A_I^{old}) \\ &\le D_\infty(A_I^{old}) - (1-\lambda)\log \ell \end{aligned}$$

Now we can use that invariant 3 holds for A_I^{old} to get

$$\begin{aligned} D_\infty(A_I^{new}) &\le (2cost_A - (1-\lambda)(n - |I^{old}|)\log \ell) - (1-\lambda)\log \ell \\ &= 2cost_A - (1-\lambda)(n - |I^{new}|)\log \ell \end{aligned}$$

For line 10-11 we note that $cost_A$ is increased by one at line 9. Since we go to the largest child $D_\infty(A_I)$ is increased by at most 1 in line 10. In line 11 when we make A_I thick the $D_\infty(A_I)$ will, as we shall see, only increase by at most 1. So invariant 3 holds.

Before we go on to invariant 4 we will describe how we make A_I thick and argue that it only increases $D_\infty(A_I)$ by 1. If A_I is non-thick we have, for some i, $\mathrm{MinDeg}_i(A_I) < \ell^\mu$ so there is some right node of $\mathrm{Graph}_i(A_I)$ that has degree less than ℓ^μ. Let us repeatedly remove all x for which $x_{I\setminus\{i\}}$ corresponds to a right node of $\mathrm{Graph}_i(A_I)$ with small degree. Clearly this will make A_I thick. $D_\infty(A_I)$ increased by 1 is the same as saying that we remove less than half of A_I so we need to argue that we do not remove too many $x \in A$. For now let us just claim that this is true and prove it later.

Claim 3.2. *Let A_I be such that $\mathrm{AvgDeg}_i(A_I) \ge \ell^\lambda/2$ for all $i \in I$. If we remove x such that $x_{I\setminus\{i\}}$ corresponds to a right node of $\mathrm{Graph}_i(A_I)$ with degree less than ℓ^μ then we will remove less than half of all $x_I \in A_I$.*

Line 10 reduces the size of A_I by at most half. This means that the $\mathrm{AvgDeg}_i(A_I)$ is also reduced by at most half so we are allowed to use the claim.

For invariant 4 we note that B and I is updated at line 7, line 15 and line 19. In line 7 the Projection Claim directly gives us that $D_\infty(B_I)$ is increased by at most 1 and since $|I|$ is reduced by 1 the invariant will still hold. For line 15 we note that $|B_I|$ is not changed at all. Of course, it might be the case that B is changed in line 15 but the definition of Complete ensures that B_I is unchanged. For line 19 it is enough to see that $D_\infty(B_I)$ will be increased by at most 1, since we go to the largest child, and $cost_B$ is increased by 1.

Now that we have proved that all invariants hold we will argue that the simulation is correct given that the invariants are correct. Further down we will prove the Projection Claim and Claim 3.2 which will be the only things left, but first a small argument that the decision tree is round-efficient and has a small cost.

Clearly the decision tree will use the same number of rounds as the communication protocol since it only does queries in between Alice speaking and Bob speaking in the simulation. The cost of the decision tree is the number of queries the decision tree does which is $n-|I|$. By invariant 3 we know that $2cost_A-(1-\lambda)(n-|I|)\log\ell \geq D_\infty(A_I) \geq 0$ and since $cost_A \leq c\log\ell$ we conclude that $n-|I| \leq 2c/(1-\lambda) = O(c)$.

Claim 3.3. *The decision tree will answer with a falsified clause of F no matter what the value of z is.*

Proof. We will show that there exists some $x \in A, y \in B$ such that $y_i[x_i] = z_i$ holds for all i. If this is true then it must be the case that the clause answered by the communication protocol, C, is falsified since $(x,y) \in (A\times B) \subseteq R_v$ and the correctness of the communication protocol means that any $(x,y) \in R_v$ falsifies C.

For any queried coordinate $i \in [n]\setminus I$ invariant 2 directly gives us $y_i[x_i] = z_i$ for all $x \in A, y \in B$. For the non-queried coordinates I we will consider the following:

```
$I' = I, A' = A, B' = B$
**while** $i \in I'$ **do**
    Use Projection Claim to get $U$.
    $A' = \{x \in A' : x_i \in U\}, B' = \{y \in B' : y_{I'\setminus\{i\}} \in \mathrm{Complete}(B', I', U, i)\}, I' = I' \setminus \{i\}$
**end while**
```

First we check the conditions for the Projection Claim: $A'_{I'}$ will be thick in the beginning by invariant 1 and the projection lemma makes sure it stays thick after we have updated. $D_\infty(B'_{I'})$ will in the beginning satisfy $D_\infty(B'_{I'}) \leq cost'_B + n - |I'|$ and according to the Projection Claim $D_\infty(B'_{I'})$ is increased by at most 1 in each step so after we are done (which is at most n steps) we have $D_\infty(B'_{I'}) \leq cost'_B + n - |I'| + n \leq c\log\ell + 2n \leq 2n\log\ell$.

We know that $D_\infty(B'_{I'}) \leq 2n\log\ell$ so B' must be non-empty. For A' we know from invariant 3 that $D_\infty(A'_{I'}) < \infty$ in the beginning and the Projection Claim states that the density loss decreases so in the end $D_\infty(A'_{I'}) < \infty$ and therefore A' is non-empty. When I' is empty we know that any $y_{[n]\setminus I} \in B'_{[n]\setminus I}$ can be completed to any value on I, in particular there is some $y \in B'$ such that $y_i[x_i] = z_i$ for all $i \in I, x \in A'$.

Since $A' \subseteq A$ and $B' \subseteq B$ we know that there is some $x \in A$ and some $y \in B$ such that $y_i[x_i] = z_i$ holds for all i. □

Now all we have left to prove Lemma 3.3 are Claim 3.1 and Claim 3.2. We begin by proving Claim 3.2.

Proof of Claim 3.2. We know that

$$\frac{|A_I|}{|A_{I\setminus\{i\}}|} = \mathrm{AvgDeg}_i(A_I) \geq \ell^\lambda/2$$

so the number of right nodes of $\mathrm{Graph}_i(A_I)$ with non-zero degree is at most $|A_{I\setminus\{i\}}| \leq 2\ell^{-\lambda}|A_I|$. Since we only remove x that are related to nodes with degree less than ℓ^μ we get an upper bound on the number of x we remove per coordinate: $2\ell^{-\lambda+\mu}|A_I|$. When we consider all coordinates and the facts $n = \ell^\gamma$ and $\mu + \gamma < \lambda$ we see that we remove at most $2\ell^{\mu+\gamma-\lambda}|A_I| < |A_I|/2$. $\square$

To prove Claim 3.1 we need to prove all three statements. We will consider a random $U \subset [\ell]$ of size ℓ^δ and look at each statement separately and show that the random U satisfy that statement with probability $1 - o(1)$. Since a random $U \subset [\ell]$ of size ℓ^δ satisfy each statement with probability $1 - o(1)$ there must exist a $U \subset [\ell]$ of size ℓ^δ that satisfy all three statements. Probability $1 - o(1)$ means that as $n \to \infty$ the probability approaches to 1. We will use the following two claims to simplify the proof of Claim 3.1.

Claim 3.4 ([RM97]). *If A_I is thick and $U \subset [\ell]$ is a random subset of size ℓ^δ then $A_{I\setminus\{i\}} = \{x_{I\setminus\{i\}} : x \in A, x_i \in U\}$ with probability at least $1 - o(1)$.*

Claim 3.5 ([DRNV16]). *If $D_\infty(B_I) \le 2n \log \ell$ and $U \subset [\ell]$ is a random subset of size ℓ^δ then $D_\infty(\text{Complete}(B, I, U, i)) \le D_\infty(B_I) + 1$ with probability at least $1 - o(1)$.*

Claim 3.5 gives us statement 3 in Claim 3.1 directly. Statement 1 says that $\{x_{I\setminus\{i\}} : x \in A, x_i \in U\}$ should be thick. Since A_I is thick $A_{I\setminus\{i\}}$ must also be thick so statement 1 is true with probability $1 - o(1)$ by Claim 3.4. Statement 2 says that $D_\infty(\{x_{I\setminus\{i\}} : x \in A, x_i \in U\}) \le D_\infty(A_I) - \log \ell + \log \text{AvgDeg}_i(A_I)$. We will again use Claim 3.4 and a fact from [GPW18] namely

$$D_\infty(A_{I\setminus\{i\}}) = D_\infty(A_I) - \log \ell + \log \text{AvgDeg}_i(A_I)$$

To see why the equation above is true just use $\text{AvgDeg}_i(A_I) = |A_I|/|A_{I\setminus\{i\}}|$ and the definition of D_∞.

The proof of Claim 3.1 is done, but we still need to prove the two claims that helped us. We begin with the proof of Claim 3.4 since it is shorter.

Proof of Claim 3.4. If we think about how U is related with our graph-perspective we see that U is a subset of the left nodes of $\text{Graph}_i(A_I)$. If $A_{I\setminus\{i\}} \neq \{x_{I\setminus\{i\}} : x \in A, x_i \in U\}$ then there is some right node of $\text{Graph}_i(A_I)$ that is not connected to any node in U. Since A_I is thick each right node has at least ℓ^μ neighbours. The probability that a random subset $U \subset [\ell]$ of size ℓ^δ would "miss" all ℓ^μ neighbours is at most

$$(1 - \ell^\mu/\ell)^{\ell^\delta}$$

Using the well known fact (e is Euler's constant)

$$e^{-1} = \lim_{h\to\infty} (1 - 1/h)^h \underset{h>0}{\implies} e^{-1} \ge (1 - 1/h)^h$$

with $h = \ell^{1-\mu}$ we get

$$(1 - \ell^\mu/\ell)^{\ell^\delta} \le \exp(-\ell^{\delta+\mu-1})$$

Now considering that there are at most $\ell^{n-1} = \exp((\ell^\gamma - 1)\ln \ell)$ right nodes the probability that a random set $U \subset [\ell]$ would "miss" all neighbours to any right vertex is at most

$$\exp(-\ell^{\delta+\mu-1}) \cdot \exp((\ell^\gamma - 1)\ln \ell) = \exp(-\ell^{\delta+\mu-1} + (\ell^\gamma - 1)\ln \ell) = o(1)$$

In the last step we used the fact that $\delta + \mu - 1 > \gamma$. $\square$

To prove Claim 3.5 we will again use a helper claim

Claim 3.6. *Let $b \in \{0,1\}$ and let $W \subseteq \{0,1\}^\ell$ be of size larger than $2^{\ell - 3n\log\ell}$. If $U \subset [\ell]$ is a random subset of size ℓ^δ then W contains a y_i such that $y_i[u] = b$ for all $u \in U$ with probability at least $1 - o(1)$.*

Proof of Claim 3.5. Here we will work with the graph $\mathsf{Graph}_i(B_I)$. Remember that the left nodes correspond to values of $y_i \in \{0,1\}^\ell$ and right nodes correspond to values of $y_{I\setminus\{i\}} \in \{0,1\}^{\ell(|I|-1)}$. Let $\hat{B}$ be the set of all $y_{I\setminus\{i\}} \in \{0,1\}^{\ell(|I|-1)}$ corresponding to right nodes that have more than $2^{\ell-2n\log\ell}/4$ neighbours. Consider a single $\hat{y} \in \hat{B}$ and use Claim 3.6 with W be the set of neighbours for both $b = 0$ and $b = 1$ it should be clear that $\hat{y} \in \mathsf{Complete}(B, I, U, i)$ with probability $1 - o(1)$. Since almost all elements of $\hat{B}$ is in $\mathsf{Complete}(B, I, U, i)$ the following holds with probability $1 - o(1)$:

$$|\mathsf{Complete}(B, I, U, i)| \geq \frac{2}{3}|\hat{B}|$$

Now it is enough to show that the size of $\hat{B}$ is large. Since all nodes in $\hat{B}$ has degree $\leq 2^\ell$ and the rest has degree $\leq 2^{\ell-2n\log\ell}/4$ the following inequality holds:

$$|B_I| \leq 2^\ell|\hat{B}| + \frac{2^{\ell-2n\log\ell}}{4}(2^{\ell(|I|-1)} - |\hat{B}|)$$

Using our assumption $D_\infty(B_I) \leq 2n\log\ell \iff |B_I| \geq 2^{\ell|I|-2n\log\ell}$ we get

$$|B_I| \leq 2^\ell|\hat{B}| + |B_I|/4$$

Putting things together we get that

$$|\mathsf{Complete}(B, I, U, i)| \geq \frac{1}{2}2^{-\ell}|B_I| \iff D_\infty(\mathsf{Complete}(B, I, U, i)) \leq D_\infty(B_I) + 1$$

holds with probability $1 - o(1)$. $\square$

Next we will use a theorem from Kruskal [Kru63], but before we state the theorem we need to introduce some definitions.

Let us call a set, $A \subseteq \{0,1\}^\ell$, of bit strings k-regular if all bit strings in A have exactly k bits set to 1. [1] If A is k-regular then define the shadow set, $\partial(A)$, to be set of all bit strings which can be obtained by taking a bit string in A and flipping a 1 to a 0. Note that $\partial(A)$ is $(k-1)$-regular.

We will use the usual binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ with the addition that if $k > n$ or $k < 0$ then $\binom{n}{k} = 0$.

Theorem 3.2 (Kruskal-Katona Theorem). *If $A \subseteq \{0,1\}^\ell$ is k-regular, and if*

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \ldots + \binom{a_s}{s}$$

then

$$|\partial(A)| \geq \binom{a_k}{k-1} + \binom{a_{k-1}}{k-2} + \ldots + \binom{a_s}{s-1}$$

[1]Note that this A has nothing to do with A corresponding to Alice's input

Define the iterated shadow set as $\partial^k(A) = \partial^{k-1}(A)$ with $\partial^0(A) = A$ and define the union as $\partial^{\leq k}(A) = \cup_{j=0}^{k} \partial^j(A)$. If we repeatedly use Kruskal-Katona we get this simple corollary.

Corollary 3.2.1. *If $A \subseteq \{0,1\}^\ell$ is k-regular, and if*

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \ldots + \binom{a_s}{s}$$

then

$$|\partial^{\leq k}(A)| \geq \sum_{j=0}^{k} \binom{a_k}{k-j} + \binom{a_{k-1}}{k-1-j} + \ldots + \binom{a_s}{s-j}$$

Before we get in to the proof of Claim 3.6, we state an elementary claim from [DRNV16].

Claim 3.7 ([DRNV16]). *Following holds*

$$\sum_{j=0}^{\ell-u} \sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i-j} \geq 2^\ell - 2^{\ell-t+1} + 2^{u \log \ell}$$

The proof of Claim 3.7 is a quite lengthy calculation using mostly elementary properties of binomial coefficients and as it does not really provide much intuition it will be skipped.

Proof of Claim 3.6. For simplicity we will prove it for $b = 0$ and by symmetry it will hold for $b = 1$ as well. Let $\mathcal{U}$ be the family of all $U \subset [\ell]$, $|U| = \ell^\delta$, such that Claim 3.6 does *not* hold. Let q be the probability that Claim 3.6 does not hold, $q = \Pr[\forall w \in W\colon \exists u \in U\colon w[u] = 1]$. The following holds

$$|\mathcal{U}| = q\binom{\ell}{\ell^\delta} = q\frac{\ell}{\ell^\delta}\binom{\ell-1}{\ell^\delta-1} \geq \sum_{i=0}^{q\frac{\ell}{\ell^\delta}} \binom{\ell-1-i}{\ell^\delta-1} = \sum_{i=0}^{q\frac{\ell}{\ell^\delta}} \binom{\ell-1-i}{\ell-\ell^\delta-i}$$

We can look at a $U \subset [\ell]$ as a bit string of length ℓ with a 0 in the ith position if and only if $i \in U$. Now define $\mathcal{U}' \subset \{0,1\}^\ell$ to be $\mathcal{U}$ but with each U as a bit string. Note that $\mathcal{U}'$ is $(\ell - \ell^\delta)$-regular. By Corollary 3.2.1 with $A = \mathcal{U}'$ we get that

$$|\partial^{\leq k}(\mathcal{U}')| \geq \sum_{j=0}^{\ell-\ell^\delta} \sum_{i=0}^{q\frac{\ell}{\ell^\delta}} \binom{\ell-1-i}{\ell-\ell^\delta-i-j}$$

Note that $\partial^{\leq k}(\mathcal{U}')$ contains no element of W. To see why this is true suppose opposite: then there is some element $w \in W \cap \partial^{\leq k}(\mathcal{U}')$. w must be in the shadow of at least one bit string corresponding to $U \in \mathcal{U}$ meaning that w has 0 in all positions $i \in U$ but that would mean that our Claim 3.6 would hold which it does not by definition of $\mathcal{U}$.

Since $\partial^{\leq k}(\mathcal{U}')$ contains no element of W the following clearly holds:

$$|W| \leq 2^\ell - |\partial^{\leq k}(\mathcal{U}')| \leq 2^\ell - \sum_{j=0}^{\ell-\ell^\delta} \sum_{i=0}^{q\frac{\ell}{\ell^\delta}} \binom{\ell-1-i}{\ell-\ell^\delta-i-j}$$

Now we use Claim 3.7 with $u = \ell^\delta$ and $t = q\frac{\ell}{\ell^\delta}$ to get

$$|W| \leq 2^\ell - (2^\ell - 2^{\ell - q\frac{\ell}{\ell^\delta}+1} + 2^{\ell^\delta \log \ell}) \leq 2^{\ell - q\frac{\ell}{\ell^\delta}+1}$$

Since we know that $|W| > 2^{\ell - 3n\log \ell}$ we have

$$\ell - 3n\log\ell < \ell - q\frac{\ell}{\ell^\delta} + 1 \iff q < 3n\ell^{\delta-1}\log\ell + \ell^{\delta-1}$$

The fact that $n = \ell^\gamma$ and $\gamma + \delta < 1$ gives $q = o(1)$.

□

3.4.4 Real communication

Due to lack of time this part is skipped. Lemma 3.3 is proven for real communication in the article "How limited interaction hinders real communication", [DRNV16].

3.5 Decision tree to Dymond-Tompa game

In this section we will be given a decision tree and build a strategy for the pebbler in a Dymond-Tompa game. More precisely, we will show the following lemma:

Lemma 3.4. *If there exists a parallel decision tree for the falsified clause search problem $Search(Peb_G)$ using r rounds and cost c then there exists a r-round strategy for the pebbler in the Dymond-Tompa game on the graph G with cost $c + 1$.*

The decision tree will be a parallel decision tree for the falsified clause search problem of a pebbling contradiction, $Search(Peb_G)$. Actually we will prove that there exists a strategy for the game on the graph $\widetilde{G}$. The graph $\widetilde{G}$ is created by taking the graph $G = (V, E)$ and adding an extra sink connected with the old sink. In other words $\widetilde{G} = (V \cup \{t'\}, E \cup \{(t, t')\})$ where t is the sink in G and t' is the new sink we add to $\widetilde{G}$. Clearly the cost for the pebbler-strategy in the Dymond-Tompa game on G is not more than that on $\widetilde{G}$ so it is sufficient to prove it for $\widetilde{G}$. If there is a path from node a to node b such that no nodes between are pebbled we say that b is *reachable* from node a.

Let's pretend that we are the pebbler and describe how the decision tree help us create a strategy. We will start in the root of the decision tree and then move down the decision tree depending on the challenger's move. If the decision tree has queried the value of x_i and we have moved down in the decision tree corresponding to the value $x_i = 0$ then we say that the node with x_i is false and similarly with $x_i = 1$ we call the node x_i true. We define the added sink t' to be a false node.

During the whole time the we keep the following conditions true:

1. The currently challenged node is false.
2. There is a path, which we denote Φ, from the currently challenged node to the sink t' that only consists of false and unqueried nodes.
3. If a node is false it lies on the path Φ.

4. A node that can reach the currently challenged node has a pebble if and only if it has been queried by the decision tree. (With exception for the added sink t')

When it is our turn we look at the decision tree and place pebbles on the nodes that the decision tree wants to query and that can reach the currently challenged node. The challenger then challenge one of the new nodes with pebbles or stays at the currently challenged node. Now we should go down in the decision tree, so we need to decide which of the queried nodes that are true and which are false. The node that the challenger chooses should be false. For all other nodes queried we say that they are false if they are on the path Φ between the node just chosen by the challenger and the sink t'.

All of our invariants are true as we go down the decision tree. The challenged node will clearly be false. For second condition if the challenged node is v and the challenger chooses node u we extend Φ to be between u and the sink t' since u can reach v the second condition will still hold. The third condition is true since we only make a node false if it is on the path Φ. For the fourth condition we note that the set of nodes that can reach the currently challenged node is only ever decreasing, and if such a node is queried then it will be pebbled as well.

We continue down the decision tree until we get to a leaf. Now we show that the Dymond-Tompa game also has ended. Since the decision tree is at a leaf it points to a clause that is falsified. Let's consider the three types of clauses in Peb_G, source axiom, pebbling axiom and sink axiom.

Case (source axiom) (pebbling axiom) In both of these cases there is a node that is false with all direct predecessors true, let's call the false node v. Condition 3 says that v must be on the path Φ and since all direct predecessors are true condition 2 gives that v must be the challenged node. All direct predecessors can reach v so condition 4 gives that all direct predecessors have pebbles. Therefore the Dymond-Tompa game has ended.

Case (sink axiom) In this case the sink of G which we called t will be true. Since t is the only node of $\widetilde{G}$ that is connected to the sink t' it must lie on all paths Φ of length greater than 1. By condition 2 and the fact that t is true we conclude that the only possible way this could happen is if t' is the challenged node. If t' is challenged the Dymond-Tompa game has ended since the only direct predecessor t is pebbled.

3.6 Lower bounds on Dymond-Tompa game

In this section we state a lemma that gives us graphs with lower bounds on the pebbler-strategy for the Dymond-Tompa game on the graphs.

Lemma 3.5. *For any $m, d \in \mathbb{N}^+$ such that m is a power of 2, there exists an explicitly constructible DAG $G(m, d)$ of depth d with $O(dm)$ nodes and indegree at most 2 such that for any $r \leq d$ the cost of an r-round Dymond-Tompa game is at least $\min\{\frac{r2^{d/r}}{8}, \frac{m}{8}\}$*

Due to lack of time the proof of this lemma is skipped. The proof is given in the article "How limited interaction hinders real communication", [DRNV16].

3.7 Putting it all together

Now we will wrap up the chapter by putting all pieces together and prove the main theorem, 3.1. For convenience we state and prove a lemma that takes us from cutting planes-refutations to Dymond-Tompa games.

Lemma 3.6 (Main lemma)**.** *Let G be DAG with n nodes and let $\ell = n^{3+\epsilon}$ for some $\epsilon > 0$. If there exists a cutting planes-refutation for the CNF-formula $Lift_\ell(Peb_G)$ with space S and length L then there exists a $O(\log L)$-round strategy for the pebbler in the Dymond-Tompa game on G with cost $O(S \log(L)/\log(\ell))$.*

Proof. By Lemma 3.2 there must exist a real communication protocol for the problem $Search(Lift_\ell(Peb_G))$ that uses $O(\log L)$ rounds and costs $O(S\log(L))$. This implies, as argued in Subsection 3.3.3, that there exists a communication protocol for $Lift_\ell(Search(Peb_G))$ as well. By Lemma 3.3 there must exist a parallel decision tree for $Search(Peb_G)$ using $O(\log L)$ rounds and with cost $O(S \log(L)/\log(\ell))$. By Lemma 3.4 there must exist a strategy for the pebbler in the Dymond-Tompa game on G with cost $O(S \log(L)/\log(\ell))$ using at most $O(\log L)$-rounds. □

Now we are ready for the main theorem which we restate here in a stronger form. The proof is not super long but a bit messy.

Theorem 3.3 (Stronger Main Theorem)**.** *For any constants $K > 0$ there exists a family of CNF-formulas $\{F_N\}_{N=1}^{\infty}$ with $O(N)$ number of clauses such that for any $\epsilon > 0$ any cutting planes-refutation that uses space less than $N^{1/10-\epsilon}$ must have length at least $2^{\log^K N}$.*

Proof. By setting $d = \log^{K+1} m$ and $r = O(\log^K m)$ in Theorem 3.5 we know that there exists a graph G with $n = O(m \log^{K+1} m)$ nodes such that any $O(\log^K m)$-round strategy costs at least $\min\{\frac{\log^K(m)2^{\log m}}{8}, \frac{m}{8}\} = \Omega(m)$. Since G has indegree 2 we know that Peb_G will have width 3 and therefore we know that $Lift_\ell(Peb_G)$ will have $O(\ell^3 n) = O(n^{10+3\epsilon})$ clauses.

For a given N we pick m to be a power of 2 such that N is at most a constant factor off from $(m\log^{K+1} m)^{10+3\epsilon}$. Since $N = \Theta(n^{10+3\epsilon})$ it is clear that $Lift_\ell(Peb_G)$ has $O(N)$ number of clauses.

Assume that there is a cutting planes-refutation for $Lift_\ell(Peb_G)$ with space $N^{1/10-\epsilon}$ and length $2^{\log^K N}$. By Lemma 3.6 there is a $O(\log^K N)$-rounds strategy for the pebbler in the Dymond-Tompa game on G with cost $O(N^{1/10-\epsilon}\log^K(N)/\log(\ell))$. Note that $\log^K N = O(\log^K n)$ and that

$$O(N^{1/10-\epsilon}\log^K(N)/\log(\ell)) < O(n^{1-\epsilon}) = O((m\log^{K+1} m)^{1-\epsilon}) = o(m)$$

which is a contradiction since the cost of the Dymond-Tompa game was $\Omega(m)$. The first inequality comes from the fact that $N^{1/10-\epsilon} \approx (n^{10+3\epsilon})^{1/10-\epsilon} < n^{1-9\epsilon}$.

□

Chapter 4

Potential improvements

In this chapter we will discuss how Lemma 3.3 could be improved and what implications this would have on the length-space trade-off.

It turns out that if we could make ℓ smaller we would actually get stronger parameters in the length-space trade-off, we also note that the only time we used that $\ell \geq n^{3+\epsilon}$ is in the proof of Lemma 3.3.

In the paper "Lifting with Sunflowers" [LMM+20] they prove a lifting lemma with $\ell = n^{1+\epsilon}$ so suppose we could improve our Lemma 3.3 to work with $\ell = n^{1+\epsilon}$ and go through the proof of the Stronger Main Theorem 3.3 to see how it would affect the results. We create a graph G with $n = O(m \log^{K+1} m)$ nodes such that any $O(\log^K m)$-round strategy costs $\Omega(m)$. $Lift_\ell(Peb_G)$ has only $O(n^{4+3\epsilon})$ clauses now. Let $N = \Theta(n^{4+3\epsilon})$ and assume that there is a cutting planes-refutation for $Lift_\ell(Peb_G)$ with space $N^{1/4-\epsilon}$ and length $2^{\log^K N}$. This would imply that there exists a $O(\log^K N)$-rounds strategy with cost

$$N^{1/4-\epsilon} \log^K(N)/\log \ell \leq N^{1/4-\epsilon} \log^K(N) < N^{1/4-\epsilon/2} < m$$

which is a contradiction, thus no such cutting planes-refutation can exist.

All this means that we could strengthen the main result to something along the line: For any $\epsilon > 0$ there exists a family of CNF-formulas such that any cutting planes-refutation of space less than $N^{1/4-\epsilon}$ requires length at least $2^{\log^{100} N}$.

How much could we improve the result by lowering ℓ? If we manage to prove Lemma 3.3 with ℓ being constant or poly-logarithmic in n we would get, by the same reasoning as above, that any cutting planes-refutation of space less than $N^{1-\epsilon}$ requires super-polynomial length.

For completeness we will also state known upper bounds for cutting planes. It turns out that *any* unsatisfiable CNF-formula can be refuted by cutting planes in constant (formula) space [GPT15]. So it would not be possible to prove an unconditional lower bound for space needed by cutting planes. In the "Limited Interaction"-article it is shown that cutting planes refutations of length $O(N)$ and space $O(N^{2/5})$ exist for the formulas, $Lift_\ell(Peb_G)$, used in this thesis. It might look like this last upper bound would limit how much we could improve

ℓ, since a constant ℓ implies a very strong lower bound. However, this is not the case because the upper bound depends on how large ℓ is.

Chapter 5
Alternative proof of lifting lemma

In this chapter we will prove a lemma that is very similar to Lemma 3.3. This lemma is the same except with the differences that we do not consider rounds nor real communication and that the parameter ℓ is set to $n^{1+\epsilon}$ instead of $n^{3+\epsilon}$.

Lemma 5.1. *Let F be an unsatisfiable CNF-formula over n variables and let $\ell = n^{1+\epsilon}$ for some $\epsilon > 0$. If there exists a deterministic communication protocol for $Lift_\ell(Search(F))$ with cost $c\log(\ell)$ then there exists a decision tree with cost $O(c)$.*

The proof will essentially be the same as the one given in "Lifting with Sunflowers" [LMM+20].

5.1 Introduction

A lot of the setup will be the same as in Section 3.4. In fact the whole overview part, 3.4.1, will hold true for this chapter as well. A lot of notations will also be the same: v is a node in the communication protocol tree, $A \subseteq [\ell]^n$ denote possible inputs for Alice, $B \subseteq \{0,1\}^{n\ell}$ are possible inputs for Bob, $I \subseteq [n]$ will be the non-queried coordinates, density loss D_∞ is the same and $R_v \subseteq [\ell]^n \times \{0,1\}^{n\ell}$ are all inputs that reach node v.

One difference is that in this chapter we will define $\lambda < 1$ to be some value such that $(1+\epsilon)\lambda > 1$. This means that $\ell^\lambda > n$.

If $y \in B$, $I \subseteq [n]$ and $\alpha \in [\ell]^{|I|}$ then $y[I,\alpha]$ means the bit string of length $|I|$ where the ith bit is the α_ith bit of y_k, where $k \in I$ is the ith smallest element of I. If $x \in [\ell]^k$ and $y \in (\{0,1\}^\ell)^k$ we will sometimes use the notation $y[x]$ to denote the bit string of length k where the ith bit is $y_i[x_i]$. The syntax $a[b]$ is quite overloaded and depends on the types of a and b, but hopefully it will not cause any confusion.

In the proof of Lemma 5.1 we will not use the graph-perspective, $\mathsf{Graph}_i(A_I)$, but instead use blockwise min-entropy as we defined in Section 2.6. We can see that in the beginning the blockwise min-entropy of A is $\log \ell$ and every time we move down in the protocol tree we decrease the blockwise min-entropy of A by at most one since we are moving to the largest child.

Before we go into the proof we will prove Full Range Lemma which is a powerful lemma that will be very useful for us later. In the proof of Full Range Lemma we will use a general statement about CNF-formulas:

Claim 5.1. *Let $C = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ be a CNF-formula over the variables $x_1, x_2, \ldots, x_n$. Let C^{mon} be the C but where we have changed every literal of the form x_i to the literal $\overline{x}_i$. Then C^{mon} will have at least as many satisfiable assignments as C.*

For the quite easy proof of Claim 5.1 see claim 2.4 in "Lifting with Sunflower" [LMM+20].

Lemma 5.2 (Full Range Lemma)**.** *Let $\ell \geq n^{1+\epsilon}$ and let $I \subseteq [n]$. Let $A_I \times B_I \subseteq [\ell]^{|I|} \times (\{0,1\}^{\ell})^{|I|}$ be such that A_I has blockwise min-entropy at least $\lambda \log \ell - O(1)$ and $|B_I| > 2^{\ell|I| - n\log \ell}$. Then there exists an $x^* \in A_I$ such that for every $\beta \in \{0,1\}^{|I|}$, there exists a $y \in B_I$ such that $y[x^*] = \beta$ holds.*

Proof. Assume for contradiction that for each $x \in A_I$ there is a $\beta_x \in \{0,1\}^{|I|}$ such that $y[x] \neq \beta_x$ holds for all $y \in B_I$. Now we will consider a CNF-formula over variables $y_1, y_2, \ldots, y_{\ell|I|}$ where we add one clause C_x for each $x \in A_I$ such that clause C_x is false exactly when $y[x] = \beta_x$. According to Claim 5.1 the number of solutions to the CNF is maximized when $\beta_x = 1^{|I|}$ holds for all x. Therefore, the following holds

$$|B_I| = |\{y \in B_I : \forall x \in A_I, y[x] \neq \beta_x\}| \leq |\{y \in \{0,1\}^{|I|\ell} : \forall x \in A, y[x] \neq 1^{|I|}\}|$$

For each $x \in A_I$ we define $S_x \subseteq [|I|\ell]$ to be all numbers (i, j) where $i \in [|I|]$ and $j \in [\ell]$ satisfy $x_i = j$. Define $S_{A_I} = \{S_x : x \in A_I\}$ and note that each S_x has size $|I|$ and that S_{A_I} is block-respecting. The blockwise min-entropy of S_{A_I} is $\lambda \log \ell - O(1)$ which, by the fact $(1+\epsilon)\lambda > 1$, is larger than $\log(K \log(|I|/(2^{-2n\log \ell})))$ for any constant K. Therefore we are allowed to use the Blockwise Robust Sunflower Lemma, 2.2, with $\mathcal{F} = S_{A_I}, s = |I|, \kappa = 2^{-2n\log \ell}$ to get that

$$\Pr[\forall S_x \in S_{A_I} : S_x \not\subseteq \mathbf{S_y}] < 2^{-2n\log \ell}$$

where $\mathbf{S_y}$ is uniformly random over $[|I|\ell]$. Another way to look at the inequality above is

$$\Pr[\forall x \in A_I : \mathbf{y}[x] \neq 1^{|I|}] < 2^{-2n\log \ell}$$

where $\mathbf{y} \in \{0,1\}^{|I|\ell}$ is the indicator vector for $\mathbf{S_y}$. By considering all $y \in \{0,1\}^{|I|\ell}$ and the above probability we get

$$|B_I| \leq |\{y \in \{0,1\}^{|I|\ell} : \forall x \in A, y[x] \neq 1^{|I|}\}| < 2^{\ell|I| - 2n\log \ell}$$

This is clearly a contradiction since $B_I > 2^{\ell|I| - n\log \ell}$. □

5.2 Rectangle partition

In this section we will define a way to partition the possible input sets A and B, and then we will state and prove a number of facts about the partition. In Algorithm 3 we define the partition formally. The statement A is fixed on $[n] \setminus I$ just means that the size of $A_{[n]\setminus I}$ is one.

Note that A^j does *not* mean a set of sequences of length j but instead j is just an index.

Here comes some claims about the rectangle partition that will be used later.

Algorithm 3 Rectangle Partition

```
1: Input: $A \subseteq [\ell]^n, B \subseteq \{0,1\}^{n\ell}, I \subseteq [n]$.
2: Assert: $A$ is fixed on $[n] \setminus I$.
3: $\mathcal{F} = \{\}, A^{\geq j} = A, j = 1$
4: while $|A^{\geq j}| \geq |A|/2$ do
5:     Let $I_j \subseteq I$ be maximal subset such that $A^{\geq j}$ has less than $\lambda \log \ell$-blockwise min-entropy on $I_j$. In case no such set exists let $I_j$ be the empty set.
6:     Let $\alpha_j \in [\ell]^{|I_j|}$ be such that $\Pr[x \in A^{\geq j}, x[I_j] = \alpha_j] \geq 2^{-\lambda |I_j| \log \ell}$.
7:     Define $A^j = \{x \in A^{\geq j}, x[I_j] = \alpha_j\}$
8:     $A^{\geq j} = A^{\geq j} \setminus A^j, \mathcal{F} = \mathcal{F} \cup \{(I_j, \alpha_j)\}, j = j + 1$
9: end while
10: for $(I_j, \alpha_j) \in \mathcal{F}$ do
11:     for $\beta \in \{0,1\}^{\ell |I_j|}$ do
12:         $B^{j,\beta} = \{y \in B, y[I_j, \alpha_j] = \beta\}$
13:     end for
14: end for
15: Return $\mathcal{F}, \{A^j\}_j, \{B^{j,\beta}\}_{j,\beta}$
```

Claim 5.2. *For all $x \in A^j, y \in B^{j,\beta}, i \in I_j$ we have $y_i[x_i] = \beta_i$*

Proof. Follows from the definitions of A^j and $B^{j,\beta}$. □

Claim 5.3. *For all j the blockwise min-entropy of $A^j_{I \setminus I_j}$ is at least $\lambda \log \ell$.*

Proof. Suppose, for contradiction, that there exist an $I^* \subseteq I \setminus I_j$ and $\alpha^* \in [\ell]^{|I^*|}$ satisfying $\Pr_{x \sim A^j}[x_{I^*} = \alpha^*] \geq 2^{-\lambda |I^*| \log \ell}$. Then the following

$$\begin{aligned}\Pr_{x \sim A^{\geq j}}[x_{I^*} = \alpha^* \wedge x_{I_j} = \alpha_j] &\geq 2^{-\lambda |I_j| \log \ell} \cdot \Pr_{x \sim A^j}[x_{I^*} = \alpha^*] \\ &\geq 2^{-\lambda |I_j \cup I^*| \log \ell}\end{aligned}$$

shows a contradiction since I_j was maximal. □

Claim 5.4. *For all $(I_j, \alpha_j) \in \mathcal{F}$ it holds that $D_\infty(A^j_{I \setminus I_j}) \leq D_\infty(A_I) - (1 - \lambda)|I_j| \log \ell + 1$.*

Proof. From the definition of α_j and A^j we get that the size of A^j is at least $|A^{\geq j}| \cdot 2^{-\lambda |I_j| \log \ell}$ and the size of $A^{\geq j}$ is, by the stop condition, at least half of A. Note that $|A^j_{I \setminus I_j}| = |A^j_I| = |A^j|$ since A^j is fixed on $[n] \setminus (I \setminus I_j)$, and similarly we have $|A^{\geq j}_I| = |A^{\geq j}|$. Putting all these facts together gives:

$$\begin{aligned}D_\infty(A^j_{I \setminus I_J}) &= |I \setminus I_j| \log \ell - \log |A^j| \\ &\leq |I \setminus I_j| \log \ell - \log(|A^{\geq j}| \cdot 2^{-\lambda |I_j| \log \ell}) \\ &\leq |I| \log \ell - |I_j| \log \ell - \log(|A|/2) + \lambda |I_j| \log \ell \\ &= D_\infty(A_I) - (1 - \lambda)|I_j| \log \ell + 1\end{aligned}$$

□

Our last claim will be that the size of Bob's possible input sets are large. We will show that there is some j such that $B^{j,\beta}$ are large for all $\beta \in \{0,1\}^{|I_j|}$. In the simulation it will be this j (or rather I_j) specifying which variables the decision tree should query.

In the proof of Claim 5.5 we will use the notation $\cup_j A^j$ to mean the union of all A^j. The size of $\cup_j A_I^j$ is at least half the size of A_I since we stop when the size of $A^{\geq j}$ is less than half of A. In other words the blockwise min-entropy of $\cup_j A_I^j$ has dropped by at most 1 compared to A_I.

Claim 5.5. *Let $I \subseteq [n]$, $A \times B \subseteq [\ell]^n \times (\{0,1\}^\ell)^n$ be such that the following holds*

1. *A_I has blockwise min-entropy at least $\lambda \log \ell - O(1)$*
2. *$|B_I| \geq 2^{\ell|I| - n\log\ell + 1}$*

Let $\mathcal{F}$, $\{A^j\}_j$, $\{B^{j,\beta}\}_{j,\beta}$ be the result of our rectangle partition on $A \times B$. There exists a j such that for all $\beta \in \{0,1\}^{|I|}$ the following holds

$$|B^{j,\beta}_{I \setminus I_j}| \geq |B_I| / 2^{\ell|I_j| + 2|I_j|\log\ell}$$

Proof. We will show that there exists a j such that for all $\beta \in \{0,1\}^{|I_j|}$ we have

$$|B_I^{j,\beta}| = |\{y \in B_I : y[I_j, \alpha_j] = \beta\}| \geq |B_I| / 2^{2|I_j|\log\ell}$$

If this is true the size of $B^{j,\beta}_{I\setminus I_j}$ will be large enough since we can at most lose a factor $2^{|I_j|\ell}$ compared to $B_I^{j,\beta}$.

Assume for contradiction that no such j exists. Then there is a $\beta_j \in \{0,1\}^{|I_j|}$ for each j such that $|\{y \in B_I : y[I_j,\alpha_j] = \beta_j\}| < |B_I|/2^{2|I_j|\log\ell}$. Now we will divide B_I into two parts, depending on if there exists a j such that $y[I_j, \alpha_j] = \beta_j$. Let $B_I^=$ be the set for which there is such a j and let the rest be $B_I^{\neq}$. The trick is now that we can show $|B_I^=| < |B_I|/2$ and $|B_I^{\neq}| < |B_I|/2$ which clearly is a contradiction.

$|B_I^{\neq}| < |B_I|/2$ since otherwise we can use the Full Range Lemma, 5.2, on $B_I^{\neq}$ and $\cup_j A_I^j$ to get some $x^* \in A_I^j$ such that every β there is some $y \in B_I^{\neq}$ with $y[x^*] = \beta$. Now pick any β such that $\beta[I_j] = \beta_j$ and we get $y[I_j, \alpha_j] = \beta_j$ which is a contradiction by definition of $B_I^{\neq}$.

To show $|B_I^=| < |B_I|/2$ we will define $\mathcal{F}(k)$ to be the set $\{(I_j,\alpha_j) \in \mathcal{F} : |I_j| = k\}$. First note that if $|I_j| = 0$ then $|B_I^{j,\beta}| = |B_I|$ and we would be done, therefore we assume that $|\mathcal{F}(0)| = 0$. Clearly there are at most $\binom{|I|}{k}$ different sets I_j of size k and each I_j can have at most ℓ^k different assignments α_j. Since $|I| \leq n < \ell^\lambda$ we have

$$\mathcal{F}(k) \leq \ell^k \binom{\ell^\lambda}{k} < \ell^k \frac{\ell^{\lambda k}}{2}$$

which holds for all $k \geq 2$. For $k = 1$ we simply have $\mathcal{F}(1) \leq \ell^{1+\lambda}$. Since each $y \in B_I^=$ satisfy $y[I_j, \alpha_j] = \beta_j$ for some j and for each j at most $|B_I|/2^{2|I_j|\log\ell}$ number of $y \in B_I$ satisfy $y[I_j,\alpha_j] = \beta_j$ we get

$$|B_I^=| \leq \sum_j |B_I| / 2^{2|I_j|\log\ell}$$

If we instead of summing over j sum over the size of I_j we get

$$\begin{aligned}
|B_I^=| &\le \sum_{k=1}^{|I|} \mathcal{F}(k) \frac{|B_I|}{2^{2k\log\ell}} \\
&< \ell^{\lambda-1}|B_I| + \sum_{k=2}^{\infty} \ell^k \frac{\ell^{\lambda k}}{2} \frac{|B_I|}{2^{2k\log\ell}} \\
&= \ell^{\lambda-1}|B_I| + \frac{|B_I|}{2} \sum_{k=2}^{\infty} \ell^{(\lambda-1)k} \\
&< |B_I|/2
\end{aligned}$$

In the last step we used the formula for geometric series.

□

5.3 Simulation

Algorithm 4 formalizes how we move down in the communication protocol tree as we create our decision tree. Afterwards we will prove that the decision tree created in this way is both correct and efficient. We also need to prove that the preconditions for Claim 5.5 holds true and that the assertion in the rectangle partition holds true (A fixed on $[n] \setminus I$).

Algorithm 4 Decision tree simulates deterministic communication protocol

```
1: $v = root, R = A \times B = [\ell]^n \times \{0,1\}^{n\ell}, I = [n], t = 0$
2: while $v$ is not a leaf do
3:     Set $v$ to $v_l$ if $|(R \cap R_{v_l})_I| > |R_I|/2$ otherwise set $v$ to $v_r$.
4:     Do rectangle partition on $(A \cap A_v) \times (B \cap B_v)$ and denote the result by $\{A^j\}_j, \{B^{j,\beta}\}_{j,\beta}, \{(I_j, \alpha_j)\}_j$.
5:     Use Claim 5.5 to get index $j$.
6:     Query $z_i$ for each $i \in I_j$ and denote the result $\beta$.
7:     $A = A^j, B = B^{j,\beta}, I = I \setminus I_j, t = t + 1$
8: end while
9: return Same clause as the communication protocol.
```

We will start by noting that A is fixed on $[n] \setminus I$ holds during the whole simulation since A^j is, by definition, fixed on I_j. Next we prove the efficiency of the decision tree.

Claim 5.6. *The decision tree will be efficient. In other words $n - |I| = O(c)$.*

Proof. We will show that

$$0 \le D_\infty(A_I) \le 2t - (1-\lambda)(n - |I|)\log\ell$$

holds for all t. This is enough since it, together with the fact $t \le c\log\ell$, gives $n - |I| \le 2c/(1-\lambda)$. Note that it holds in the beginning when $t = 0$. Assuming it holds for t we can show that it holds for $t + 1$

$$\begin{aligned} D_\infty(A^j_{I\setminus I_j}) &\le D_\infty((A \cap A_v)_I) - (1-\lambda)|I_j|\log\ell + 1 && \text{(by Claim 5.4)} \\ &\le 1 + D_\infty(A_I) - (1-\lambda)|I_j|\log\ell + 1 \\ &\le 1 + 2t - (1-\lambda)(n - |I|)\log\ell - (1-\lambda)|I_j|\log\ell + 1 \\ &= 2(t+1) - (1-\lambda)(n - |I \setminus I_j|)\log\ell \end{aligned}$$

□

To help us prove correctness and the preconditions for Claim 5.5 we will prove that the following invariants are true in the beginning of each iteration of the while loop (and at the end of the simulation).

1. $y_i[x_i] = z_i$ is true for all $i \notin I$.
2. A_I has blockwise min-entropy at least $\lambda \log \ell$.
3. $|B_I| \ge 2^{\ell|I| - t - 2(n-|I|)\log\ell}$.

Proof of invariants. We will prove that the invariants are correct by induction over t. Note that all conditions hold when $t = 0$. Now we want to show, assuming the conditions hold for t, that all conditions hold for $t + 1$.

Invariant 1 follows from Claim 5.2 and invariant 2 follows from Claim 5.3. To prove invariant 3 we need to show

$$|B^{j,\beta}_{I\setminus I_j}| \ge 2^{\ell|I\setminus I_j| - (t+1) - 2(n - |I\setminus I_j|)\log\ell}$$

Using Claim 5.5 and then the induction hypothesis we get

$$\begin{aligned} |B^{j,\beta}_{I\setminus I_j}| &\ge |(B \cap B_v)_I| / 2^{\ell|I_j| + 2|I_j|\log\ell} \\ &\ge 2^{\ell|I| - t - 2(n-|I|)\log\ell - 1} / 2^{\ell|I_j| + 2|I_j|\log\ell} \\ &= 2^{\ell|I\setminus I_j| - (t+1) - 2(n - |I\setminus I_j|)\log\ell} \end{aligned}$$

□

Now when we have proved that the invariants are correct we can go on to prove that we are in fact allowed to use Claim 5.5 and then finally the correctness of our decision tree.

Claim 5.7. *The preconditions for Claim 5.5 on line 5 in Algorithm 4 is true*

Proof. We should show that $I \subseteq [n]$, $(A \cap A_v) \times (B \cap B_v)$ satisfy the conditions for Claim 5.5. The blockwise min-entropy of $(A \cap A_v)_I$ is at least one less than the blockwise min-entropy of A_I. So condition 1 must be true since blockwise min-entropy of A_I is at least $\lambda \log \ell$. Condition 2 holds since:

$$|(B \cap B_v)_I| \ge 2^{\ell|I| - t - 2(n-|I|)\log\ell}/2 \ge 2^{\ell|I| - n\log\ell + 1}$$

The first inequality holds by invariant 3 and the second inequality holds by the facts: $n - |I| = O(c)$, $t < c\log\ell$ and $c = o(n)$. □

Claim 5.8. *The decision tree will answer with a falsified clause of F no matter what the value of z is.*

Proof. As we did in the proof of 3.3 we will show that there exists some $x \in A, y \in B$ such that $y_i[x_i] = z_i$ holds for all $i \in [n]$. Now assume that we have reached the end of the simulation and that the invariants hold.

By invariant 1 we know that $y_i[x_i] = z_i$ holds for all $x \in A, y \in B, i \in [n] \setminus I$.

By invariant 2 we know that A_I has blockwise min-entropy $\lambda \log \ell$ and by invariant 3 we know that $|B_I| \geq 2^{\ell|I| - n \log \ell}$ and we can thus use the Full Range Lemma, 5.2, to get that there exists an $x \in A, y \in B$ such that $y_i[x_i] = z_i$ holds for all $i \in I$. □

5.4 Generalisations

Lemma 5.1 only works for deterministic communication and it does not consider rounds. As we ideally would like to use 5.1 to improve our main theorem, 3.1, we need to extend it to handle both real communication and be round-efficient.

In [LMM+20] they do actually extend it to real communication, but in an earlier draft they did not.

5.4.1 Real communication

It turns out it is actually quite easy to generalise Lemma 5.1 to real communication. The proof will in fact work the exact same way with some additional reasoning about the size of $(A \cap A_v)_I \times (B \cap B_v)_I$.

Let us first consider deterministic communication and think about how our rectangle, $R = A \times B$, of compatible inputs changes when we go down in the protocol tree. If Alice speaks then $A \cap A_v$ will be at least half of A and $B \cap B_v = B$. Similarly if Bob speaks then $B \cap B_v$ will be at least half of B and $A \cap A_v = A$.

Now consider what happens for real communication where Alice picks a real number ϕ_x and Bob picks a real number ψ_y. Suppose we sort Alice's inputs according to the value of ϕ_x and Bob's inputs according to the value of ψ_y. We then associate each value of ϕ_x with the rows of a matrix and each value of ψ_y with the colons and fill the cell at row ϕ_x and colon ψ_y with the value of $\phi_x \leq \psi_y$, see Figure 5.1 for an example. As noted by [Joh98] there will always be at least one quadrant that has exactly one value of $\phi_x \leq \psi_y$, so even though $R \cap R_v$ is not a rectangle we can always restrict it to a rectangle such that $A \cap A_v$ is at least half of A and $B \cap B_v$ is at least half of B. If we read through the proof of Lemma 5.1 carefully we can see that this is in fact enough to make the proof work for the real communication.

5.4.2 Round-efficiency

It is much more difficult to extend Lemma 5.1 to handle rounds efficiently. If we look at Algorithm 4 we can see that every time we move down in the protocol tree there is the potential of a query. One natural approach to make Algorithm 4 handle rounds would be Algorithm 5. The only things that are unclear about Algorithm 5 are how we should choose the j at line 6 and what B^j is (as opposed to $B^{j,\beta}$). There are a number of things that are good about this approach. The decision tree will clearly be round-efficient and the proof

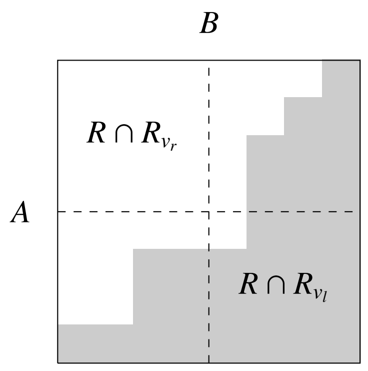


Figure 5.1: Monotone matrix. The white cells correspond to $\phi_x \leq \psi_y$, which is the same as $R \cap R_{v_r}$, and the grey cells corresponds to $\phi_x > \psi_y$, which is the same as $R \cap R_{v_l}$.

Algorithm 5 Attempt of round-efficient simulation of communication protocol.

```
1:  $v = root, R = A \times B = [\ell]^n \times \{0,1\}^{n\ell}, I = [n], t = 0$
2:  while $v$ is not a leaf do
3:      while Alice speaks at $v$ do
4:          Set $v$ to $v_l$ if $|(R \cap R_{v_l})_I| > |R_I|/2$ otherwise set $v$ to $v_r$.
5:          Do rectangle partition on $(A \cap A_v) \times (B \cap B_v)$.
6:          Take some good $j$
7:          $A = A^j, B = B^j, I = I \setminus I_j$
8:      end while
9:      Query $z_i$ for each $i \in I_j$ for all $j$ chosen.
10:     Restrict $B$ to be compatible.
11:     while Bob speaks at $v$ do
12:         Set $v$ to $v_l$ if $|(R \cap R_{v_l})_I| > |R_I|/2$ otherwise set $v$ to $v_r$.
13:         $B = B \cap B_v$
14:     end while
15: end while
16: return Same clause as the communication protocol.
```

of efficiency and correctness could be the same if the invariants would hold. Let us remind ourselves about the three invariants:

1. $y_i[x_i] = z_i$ is true for all $i \in [n] \setminus I$.
2. A_I has blockwise min-entropy at least $\lambda \log \ell$.
3. $|B_I| \geq 2^{\ell|I| - t - 2(n-|I|)\log \ell}$.

Both invariant 1 and invariant 2 will hold for any j, so we only really need to care about invariant 3.

When we did not care about rounds we used Claim 5.5 to get a j such that all $B^{j,\beta}$ are large. The problem now is that we can not do the query, so we do not know the value of β, but we still need to handle B somehow. One way to go forward here would be to do something similar to what was done in Section 3.4. If we let B^j be the "intersection of the projections" of all $B^{j,\beta}$ then we only need to prove that B^j are large. Ideally we would like to strengthen Claim 5.5 to, instead of showing that $|B^{j,\beta}_{I\setminus I_j}| \geq |B_I|/2^{\ell|I_j|+2|I_j|\log \ell}$, show that

$$|B^j_{I\setminus I_j}| = |\bigcap_{\beta\in\{0,1\}^{|I_j|}} B^{j,\beta}_{I\setminus I_j}| \geq |B_I|/2^{\ell|I_j|+2|I_j|\log \ell}$$

Unfortunately, there does not seem to be any way to adapt the current proof of Claim 5.5 to handle this change.

If we go back to Section 3.4 and look at the Claim 3.5 we see that it is actually quite close to what we want. The claim roughly states that $|\mathsf{Complete}(B, I, U, i)| \geq |B_I|/(2^{1+\ell})$ is very likely to hold. There are a couple of differences between Claim 3.5 and what we want. One is that Claim 3.5 only considers a single coordinate i and another is that we do not have a random U. We could actually extend the definition of Complete to handle multiple coordinates in a fairly natural way, $\mathsf{Complete}(B, I, \{U_i : i \in I_j\}, I_j)$ are all those $y_{I\setminus I_j}$ such that there is an $y \in B$ with $y_i[u] = b$ for all $u \in U_i, i \in I_j$. We would then have (at least for constant size I_j) that $\mathsf{Complete}(B, I, \{U_i : i \in I_j\}, I_j) \geq |B_I|/(2^{\ell|I_j|+|I_j|})$ is very likely to hold. If we think about $\{U_i : i \in I_j\}$ a little bit we see that it corresponds to $\alpha \in [\ell]^{|I_j|}$ with the differences that α is not random and that each U_i is a subset of $[\ell]$ instead of just an element of $[\ell]$. The fact that U is a subset instead of just an element is not a problem since smaller U would only make $\mathsf{Complete}(B, I, U, i)$ larger. So for random assignments (I_j, α_j) it is very likely that $|B^j_{I\setminus I_j}| \geq |B_I|/2^{\ell|I_j|+|I_j|}$ which is even stronger than what we need. The problem is of course that I_j and α_j are not random.

5.4.3 Min-Entropy vs Graphs

One interesting thing would be to compare between how the min-entropy used in this alternative lifting lemma compares to the graph perspective used in Chapter 3. This could perhaps help us understand similarities and differences between the proofs of the different lifting lemmas. In this section we will compare MinDeg and AvgDeg to MinEntropy and BlockwiseMinEntropy.

First we recap the definition of $\mathsf{Graph}_i(A_I)$. The graph $\mathsf{Graph}_i(A_I)$ is bipartite graph with left nodes corresponding to values of x_i and right nodes corresponding to values of $x_{I\setminus\{i\}}$ and for each $x \in A$ we add an edge between left node x_i and right node $x_{I\setminus\{i\}}$. $\mathsf{MinDeg}_i(A_I)$ and $\mathsf{AvgDeg}_i(A_I)$ are then the minimum (resp. average) degree of the right nodes with non-zero degree.

In Chapter 3 the condition to let the decision tree query was that $\mathsf{AvgDeg}_i(A_I) \geq \lambda \log \ell$ holds for all $i \in I$. Instead of having $\mathsf{AvgDeg}_i(A_I)$ one possibility could be to have $\mathsf{MinEntropy}_i(A_I) \geq \lambda \log \ell$ for all $i \in I$. Here $\mathsf{MinEntropy}_i(A_I)$ would be the min-entropy of the random variable $\mathbf{x_i} \in A_I$ where $\mathbf{x} \in A_I$ is uniformly random. Remembering the definition of min-entropy we have that $\mathsf{MinEntropy}_i(A_I) = \min_{x'\in A} -\log(\Pr[\mathbf{x_i} = x'_i])$ So $\mathsf{MinEntropy}_i(A_I)$ depends on how probable the most likely value of $\mathbf{x_i}$ is and if we relate this to $\mathsf{Graph}_i(A_I)$ we can see that $\Pr[\mathbf{x_i} = x'_i]$ is the same as the maximum degree of a left node divided by the sum of degrees of the left nodes. Since the maximum degree of a left node is at most the number of non-zero degree right nodes, $|A_{I\setminus i}|$, we have that $\mathsf{MinEntropy}_i(A_I) \geq -\log(|A_{I\setminus\{i\}}|/|A_I|)$. We also know that $\mathsf{AvgDeg}_i(A_I) = |A_I|/|A_{I\setminus\{i\}}|$ so actually the following holds

$$\mathsf{MinEntropy}_i(A_I) \geq \log(\mathsf{AvgDeg}_i(A_I))$$

This means that $\mathrm{AvgDeg}_i(A_I) \geq \ell^\lambda$ implies $\mathrm{MinEntropy}_i(A_I) \geq \lambda \log \ell$. The other way is not true since the following counterexample

$$A_I = \{(1,1,\ldots,1),(2,2,\ldots,2),\ldots,(i,i,\ldots,i),\ldots,(n,n,\ldots n)\}$$

has $\mathrm{MinEntropy}_i(A_I) = \log \ell$ whereas $\mathrm{AvgDeg}_i(A_I) = 1$. This shows that $\mathrm{AvgDeg}_i(A_I) \geq \ell^\lambda$ is a strictly stronger condition than $\mathrm{MinEntropy}_i(A_I) \geq \lambda \log \ell$.

The above counterexample also highlights a somewhat unintuitive property of blockwise min-entropy: even if $\mathrm{MinEntropy}_i(A_I) \geq \lambda \log \ell$ holds for all i it could be the case that the blockwise min-entropy of A_I is close to 0.

We will now show with an example that a large blockwise min-entropy do not imply large MinDeg. Let $A \subseteq [\ell]^n$ be the set $[\ell-1]^n \cup \{\ell\}^n$, in other words A has $(\ell-1)^n$ sequences consisting of only elements from $[\ell-1]$ and then a single sequence with many ℓ. The right-node corresponding to the ℓ sequence will have degree 1, so $\mathrm{MinDeg}_i(A) = 1$ for all $i \in [n]$. If we now look at the blockwise min-entropy for A we have by definition

$$\min_{I \subseteq [n], x'_I \in A_I} \frac{1}{|I|} \log(1/\Pr[\mathbf{x_I} = x'_I])$$

Any sequence in $[\ell-1]^{|I|}$ will have probability $\frac{(\ell-1)^{n-|I|}}{(\ell-1)^n+1}$ to occur, which is the maximum probability, so we get that the blockwise min-entropy of A is

$$-\frac{1}{|I|} \log\left(\frac{(\ell-1)^{n-|I|}}{(\ell-1)^n+1}\right) \approx \log \ell$$

The above example shows that the condition $\mathrm{MinDeg}_i(A_I) \geq \ell^\mu$ required in the proof of Lemma 3.3 is qualitatively different, and possible stronger, than corresponding condition $\mathrm{BlockwiseMinEntropy}(A_I) \geq \lambda \log \ell$ required in the proof of 5.1.

Chapter 6

Summary and Conclusion

The aim of the thesis was to combine the techniques used in "Limited interaction" [DRNV16] with techniques from "Lifting with Sunflowers" [LMM+20]. In order to manage this the proof in "Lifting with Sunflowers" needed to be extended to work with real communication and being round-efficient. It turns out that the former was relatively easy to achieve, while handling rounds is much more challenging.

In Section 5.4 we outlined an attempt of making Lemma 5.1 round-efficient. It was the most promising approach that was tested, but still there were some obstacles. The main problem was proving the size of the intersection of all $B^{j,\beta}_{I\setminus I_j}$. Adapting the proof of Claim 5.5 did not seem doable. Even though we did not find any way to adapt Claim 5.5 it of course still might be possible to use sunflowers or the full-range lemma in some creative way to it. We also looked at the "random assignment" approach that was used in the proof of Lemma 3.3. The main problem here is that the assignment α is not random, which might be possible to fix by changing the rectangle partition, but the issue will then be to prove the uniformity of $\boldsymbol{A}$.

In our attempt to generalise the lemma we only talked about extending the lemma to handle rounds when using deterministic communication. Even if we could extend the lemma to handle rounds we would still need to do real communication at the same time, which may or may not be easy.

We also looked at some of the differences between the entropy based approach used in the proof of Lemma 5.1 and the graph based approach used in the proof of Lemma 3.3. We saw that even if the BlockwiseMinEntropy was large the MinDeg could be really terrible. It could potentially be that the weaker condition of BlockwiseMinEntropy is not sufficient to guarantee round-efficiency, so this might be why it is hard to prove the round-efficiency with the approach we used.

References

[ALWZ20] Ryan Alweiss, Shachar Lovett, Kewen Wu, and Jiapeng Zhang. Improved bounds for the sunflower lemma. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 624–630, 2020.

[CCT87] William Cook, Collette R Coullard, and Gy Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.

[DRNV16] Susanna F De Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 295–304. IEEE, 2016.

[ER60] Paul Erdös and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960.

[Gal06] Jean Gallier. The completeness of propositional resolution: A simple and constructive proof. *arXiv preprint cs/0606084*, 2006.

[GPT15] Nicola Galesi, Pavel Pudlák, and Neil Thapen. The space complexity of cutting planes refutations. In *30th Conference on Computational Complexity (CCC 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[GPW18] Mika Goos, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM Journal on Computing*, 47(6):2435–2450, 2018.

[Joh98] Jan Johannsen. Lower bounds for monotone real circuit depth and formula size and tree-like cutting planes. *Information Processing Letters*, 67(1):37–41, 1998.

[Kru63] Joseph B Kruskal. The number of simplices in a complex. *Mathematical optimization techniques*, 10:251–278, 1963.

[LMM⁺20] Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. Lifting with sunflowers. In *Electron. Colloquium Comput. Complex*, page 111, 2020.

[RM97] Ran Raz and Pierre McKenzie. Separation of the monotone nc hierarchy. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 234–243. IEEE, 1997.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 23 februari, 2022

EXAMENSARBETE Proof complexity with the help of sunflowers
STUDENT Jonatan Nilsson
HANDLEDARE Jakob Nordström (LTH)
EXAMINATOR Jonas Skeppstedt (LTH)

Bevis-komplexitet med hjälp av solrosor

POPULÄRVETENSKAPLIG SAMMANFATTNING **Jonatan Nilsson**

Att hitta lösningar till ekvationer är ett väldigt fundamentalt problem. Ett relaterat problem är att bevisa att en olösbar ekvation saknar lösning. I bevis-komplexitet studerar man hur stora sådana olösbarhets-bevis måste vara. Det här arbetet undersöker om förbättringar inom solrosor (en del av det matematiska området kombinatorik) kan användas för att förbättra vår förståelse av bevis-komplexitet.

I det här arbetet så visar vi att det finns olösbara ekvationer som saknar små bevis. Dessa ekvationerna är då i någon mening extra svåra. En del av metoden för att visa att dessa olösbara ekvationer saknar små bevis är att från ekvationen skapa ett kommunikationsproblem.

Ett kommunikationsproblem har två personer, Alice och Bob, som tillsammans vill lösa ett problem. Alice och Bob har inte samma information, tillsammans har de tillräckligt för att lösa problemet, men inte var för sig. Alice och Bob måste därför prata med varandra för att kunna lösa problemet, men de vill helst kommunicera så lite som möjligt. Ett protokoll specificerar hur Alice och Bob ska prata för att lösa sitt problem.

Om en ekvation har ett litet olösbarhets-bevis så kommer det finnas ett litet protokoll (Alice och Bob behöver bara prata lite grann) till vårt kommunikationsproblem. Vi visar sedan att för vissa ekvationer så finns det inga små protokoll och därmed inga små bevis. Det är det här sista steget, att visa att det inte finns några små protokoll, som skulle kunna förbättras av solrosor.

Vad är då solrosor för någonting? En solros är ett antal överlappande mängder där snittet av varje par av mängder är snittet av alla mängder. Till exempel i Figur 1 så innehåller alla mängder de 3 kulorna i mitten medan alla andra kulor bara finns i en enda mängd. Inom området solrosor så är man intresserad av hur många överlappande mängder som behövs för att det garanterat ska finnas en solros av nån viss storlek.

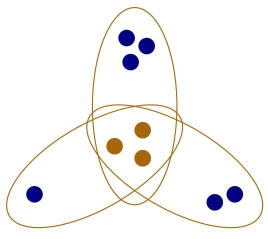


Figure 1: Exempel på en solros.

Det visar sig att det finns en koppling mellan kommunikationsproblem och solrosor som ungefär kan formuleras så här: små protokoll finns endast om det inte finns någon stor solros. Problemet är att kopplingen mellan solrosor och kommunikationsproblem har hittills inte varit tillräckligt stark för att det ska funka i vårt fall. Detta arbete har undersökt möjligheter att förbättra kopplingen mellan solrosor och kommunikationsproblem på ett sådant sätt att resultat inom bevis-komplexitet hade kunnat stärkas. Vi lyckades dock inte visa några resultatet inom området bevis-komplexitet som inte redan var kända.