Lower Bounds and Trade-offs in Proof Complexity

SUSANNA F. DE REZENDE

Doctoral Thesis
Stockholm, Sweden 2019

KTH School of Electrical Engineering and Computer Science
TRITA-EECS-AVL-2019:47 SE-100 44 Stockholm
ISBN 978-91-7873-191-6 SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i datalogi fredagen den 14 juni 2019 klockan 14.00 i Kollegiesalen, Kungl Tekniska högskolan, Brinellvägen 8, Stockholm.

Tryck: Universitetsservice US AB

Abstract

Propositional proof complexity is a field in theoretical computer science that analyses the resources needed to prove statements. In this thesis, we are concerned about the length of proofs and trade-offs between different resources, such as length and space.

A classical NP-hard problem in computational complexity is that of determining whether a graph has a clique of size k. We show that for all $k \ll n^{1/4}$ regular resolution requires length $n^{\Omega(k)}$ to establish that an Erdős–Rényi graph with n vertices and appropriately chosen edge density does not contain a k-clique. In particular, this implies an unconditional lower bound on the running time of state-of-the-art algorithms for finding a maximum clique.

In terms of trading resources, we prove a length-space trade-off for the cutting planes proof system by first establishing a communication-round trade-off for real communication via a round-aware simulation theorem. The technical contribution of this result allows us to obtain a separation between monotone-AC^{i-1} and monotone-NC^{i}.

We also obtain a trade-off separation between cutting planes (CP) with unbounded coefficients and cutting planes where coefficients are at most polynomial in the number of variables (CP*). We show that there are formulas that have CP proofs in constant space and quadratic length, but any CP* proof requires either polynomial space or exponential length. This is the first example in the literature showing any type of separation between CP and CP*.

For the Nullstellensatz proof system, we prove a size-degree trade-off via a tight reduction of Nullstellensatz refutations of pebbling formulas to the reversible pebbling game. We show that for any directed acyclic graph G it holds that G can be reversibly pebbled in time t and space s if and only if there is a Nullstellensatz refutation of the pebbling formula over G in size $t + 1$ and degree s.

Finally, we introduce the study of cumulative space in proof complexity, a measure that captures the space used throughout the whole proof and not only the peak space usage. We prove cumulative space lower bounds for the resolution proof system, which can be viewed as time-space trade-offs where, when time is bounded, space must be large a significant fraction of the time.

Sammanfattning

Satsbeviskomplexitet är ett område inom teoretisk datalogi som analyserar de resurser som behövs för att bevisa satser. I denna avhandling är vi intresserade av bevisens längd och avvägningar mellan olika resurser, såsom längd och minne.

Ett klassiskt NP-svårt problem i beräkningskomplexitet är att avgöra om en graf har en klick av storlek k. Vi visar att för alla $k \ll n^{1/4}$ krävs längd $n^{\Omega(k)}$ i reguljär resolution för att bevisa att en Erdős–Rényi graf med n hörn och lämpligt vald kantdensitet inte innehåller en k-klick. I synnerhet innebär detta en ovillkorlig undre gräns på körtiden för de för närvarande bästa algoritmerna för att hitta en maximal klick.

När det gäller resursfördelning bevisar vi en avvägning mellan längd och minne för bevissystemet skärande plan (cutting planes) genom att först upprätta en avvägning för kommunikations-rundor för reell kommunikation via ett simuleringssats. Det tekniska bidraget från detta resultat gör det möjligt för oss att få en separation mellan monoton-AC^{i-1} och monoton-NC^{i}.

Vi får också en avvägningsseparation mellan skärande plan (CP) med obegränsade koefficienter och skärande plan där koefficienterna högst är polynomiskt stora i antalet variabler (CP*). Vi visar att det finns formler som har CP-bevis i konstant minne och kvadratisk längd, men där alla CP* bevis kräver antingen polynomiskt minne eller exponentiell längd. Detta är det första exemplet som visar en separation mellan CP och CP*.

För Nullstellensatz-bevissystem visar vi en avvägning mellan storleks och gradtal via en optimal reduktion av Nullstellensatz-refutationer av pebblingformler till reversibla stenläggningsspel, eller pebblingspel. Vi visar att för alla riktade acykliska grafer G gäller att G har en reversibel pebbling-strategi i tid t och minne s om och endast om det finns ett Nullstellensatz-bevis för pebblingformeln över G i storlek $t+1$ and grad s.

Slutligen introducerar vi studien av kumulativt minne i beviskomplexitet, som bokför det totala minne som används genom hela beviset, istället för endast det maximala. Vi bevisar kumulativa undre gränser för resolution, som kan betraktas som avvägningar mellan längd och minne: när tiden är begränsad, behöver beviset använda stort minne under en betydande del av tiden.

Acknowledgements

I have very much to thank my advisor, Jakob Nordström, for, and could hardly fit it all in this page. To begin with, he introduced me to—and before that, convinced me to study—the fascinating field of proof complexity. Thank you for sharing your enthusiasm for research in general and proof complexity in particular. I am also grateful for your untiring motivation and guidance throughout these years, for your patience in teaching writing skills, for your good humour, and for your openness to share so many interesting problems. I will have fond memories of research discussions in the research lounge, in the corridors, and during long evenings at Simons.

I would also like to thank my co-advisor, Johan Håstad, for always being available to talk about research or anything else. Thank you for sharing and listening to research ideas, and for all the good advice you have given me, specially during this last year.

A special thanks to all the people I have worked with during these five years: Marc, Ilario, Massimo, Albert, Alexander, Kilian, Dmitry, Sagnik, Aaron, Or, Robert and Toni. Not all our attempts of solving problems, to put it mildly, were successful—some of the ones that were can be found in this thesis—but regardless it was a great pleasure working with you. I learned a lot from our discussions and email correspondence, and I hope to continue collaborating in the future. The innumerable hours, days and months—sometimes even years, although thankfully those are numerable—spent coming up with several buggy, and occasionally a few successful, ideas are what makes the research experience most enjoyable.

To other colleagues that have made the field of computational complexity so welcoming. In particular, I am grateful to Paul Beame, Igor Oliveira and Rafael Oliveira for providing good orientation and advice. A special thanks to Igor for helping me write my first real grant application. Apropos grants, I am thankful to the European and to the Swedish Research Council who provided funding for my first years of PhD studies and to the Knut and Alice Wallenberg foundation that have not only funded my last years but have also granted me a postdoc scholarship for the next two years. I would also like to thank the Simons Institute for the semester spent there as a research fellow with support from Google and the National Science Foundation.

I am very grateful to all the current and former members of our approximation, proof complexity and SAT-solving group for creating such a dynamic research environment: Johan, Jakob, Per (with a special thanks for the help with the Swedish abstract), Marc, Ilario, Massimo, Kilian, Dmitry, Sagnik, Aaron, Mladen, Christoph, Joseph, Guillaume, Jonas, Aleksa, Jan, Stephan, Jesús, Meysam, Janne, and Jo. A big thank you to all those who are or have been in the TCS department for the pleasant conversations over lunch or coffee and for making our workplace so agreeable. In addition to the people I have already mentioned, I would like to mention (with the risk of forgetting people): Danupon (special thanks for reading an early version of my thesis and for the helpful comments), Mads, Viggo, Karl, Martin, Sonja, Musard, Philipp, Dilian, Johan Boye, Douglas, Cyrille, Roberto, and Elena. Also Vahid, Stefan, Johan Karlander, and the late

Henrik Eriksson: tack för att ni uppmuntrade mig att prata svenska! Finally, thanks to my fellow PhD colleagues: Cenny, Sangxia, Benny, Oliver, Pedro, Hamed, Adam, Hojat, Freyr, Jana, Lukáš, Thatchaphol, Jan, Andreas, Xin, Mohit, He Ye, and Christian. A special thanks to Emma for always being willing to give a helping hand—even last minute—and, in particular, for having more than once corrected my texts in Swedish.

I also want to thank so many other friends that make Sweden feel like home. Michelle thank you for always being so encouraging and supportive: you made this PhD journey so much nicer and it has been wonderful to walk this home stretch with you. Isabel and José, thank you for the company and motivation in the endeavour of learning Swedish. Mina, Inês, Joana and Elin: I'm happy we met; it was a real pleasure working with you. Wanjiao thank you for delightful conversations and for always being so understanding. Marília, Bianca, Cris, Joyce, Juliana, Gabi, Poliana, and many others: thank you for bringing some Brazilian warmth all year round. To so many friends I have made in Sweden—which to me more than friends are family—thank you for your support and kindness.

A heartfelt thank you to my family back in Brazil. Mom, Dad: it is impossible to transmit to you how thankful I am for your unconditional love, for always being there and for supporting me, no matter what. And thank you, Dad, for proofreading this thesis. To my siblings: Daniel and Adriana, Diana, Joel and Paula, Beatriz, Djenane and Denise. How I appreciate every moment spent with you (even if it is over the phone). You are all a constant encouragement to strive to make this world a better place. A sincere thank you to my grandparents: you have taught me so much about the important things in life.

Finally, I would like to thank God, from whom I have received so much, for creating such a rich and beautiful world and for leaving its truth and beauty for us to discover, little by little.

Contents

Those who assert that the mathematical sciences say nothing of the beautiful or the good are in error. For these sciences say and prove a great deal about them; if they do not expressly mention them, but prove attributes which are their results or definitions, it is not true that they tell us nothing about them.

— Aristotles, *Metaphysica, Book 13 Part 3*

Part I

Thesis

Chapter 1

Introduction

Everyone knows a beautiful proof when they see one. It most often involves an ingenious insight or unexpected connections. However, there is something more essential than cleverness that earns a proof the title of beautiful: simplicity. A proof is only aesthetically pleasing if it is simple, short and easy to follow. (Of course the exact meaning of "simple", "short" and "easy" might differ from person to person, but this is beside the point here.) We are interested in studying these attributes of proofs. Do all theorems have beautiful proofs?

For the purpose of this discussion, we can think of a theorem as a statement that is always true—also known as *tautology*—and a proof as a sequence of lines, each of which can be derived from previous lines. The number of lines in the proof is the *length* of the proof; the amount of information contained in each line and what rules are used to derive a new line—the *proof system*—defines the complexity of the proof; and the number of lines we must keep in memory in order to verify the proof—often referred to as the *space* of the proof—determines how easy it is to follow.

With this terminology, the questions that motivate this thesis can be phrased as follows. What characterises tautologies that require long proofs in a given proof system? Are there tautologies where minimising proof space leads necessarily to a large increase in proof length?

These inquiries emanate from the central question in proof complexity: is there a proof system in which every tautology has a short (i.e., polynomial length) proof? If you were to ask a computer scientist or a mathematician, they would probably say they believe not—in fact, a substantial part of complexity theory is based on this assumption—yet this has never been demonstrated. This is (literally) the million-dollar question in proof complexity: a negative answer would solve the P vs. NP question, a problem that is recognised as one of the most fundamental in mathematics and is among the seven Clay Institute Millennium Prize Problems [Mil00].

While solving this problem seems currently out-of-reach, one could aim at the less ambitious goal of showing that a particular proof system does not have polynomial

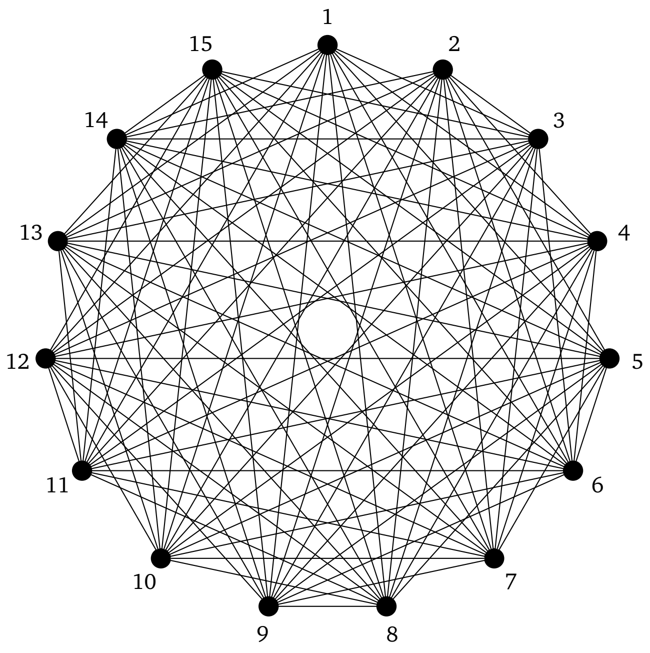


Figure 1.1: Invitee graph

length proofs of every tautology. This line of research was initiated by Cook and Reckhow [CR79] and led to the endeavour—often referred to as Cook's program—of proving lower bounds for increasingly stronger proof systems with the intention of shedding light on the P vs. NP problem. Regardless of how realistic this program is, understanding the power and limitation of proof systems is interesting in its own right and—spoiler alert—also has applications in the design and analysis of algorithms.

To illustrate the problem of determining if a tautology has a proof of polynomial length, let us consider the following hypothetical problem. Suppose you are hosting a party and you are deciding who to invite. You would like to have a nice environment in the party so you do not want to invite two people that do not get along with each other. That said, you would like to have as many people as possible in your celebration.

In graph theoretical terms, this problem is called the maximum clique problem. The possible invitees are the vertices in the graph, and two vertices are connected if these two people get along with each other. A clique in this graph is a subgraph in which every pair of vertices are connected. The problem is then to find a clique of largest size.

Returning to your invitee problem, suppose you have fifteen possible friends to invite and that their affinity-graph looks like the one in Figure 1.1. Can you find a largest clique in this graph?

It will probably take you only a minute or two to find a clique of size five (there are in fact 243 such cliques). But are these the largest cliques? How can you be certain that there is no clique of size six? How can you present a proof of this fact?

One way to convince yourself that there is no clique of size six is to try all possible subsets of size six and check that none of these form a clique. Since there are fifteen vertices in this graph, you would have to consider $\binom{15}{6} = 5005$ different subsets—a quite tedious task. If you were to write out this method of reasoning you would have a proof that there is no clique of size six: it would indeed be a simple proof (in that the method of reasoning behind it is quite simple), but would not qualify as short in terms of the size of the graph. Is there a shorter proof of this fact, or is this type of "brute-force" reasoning indeed necessary?

For this particular graph, an attentive observer might notice some symmetry and take advantage of this fact to not have to consider all 5005 subsets of size six. A perhaps even sharper observer might note that we can partition the vertices of this graph into five parts, each of which contains three vertices that are pairwise not connected. Clearly, any set of six vertices must contain at least two vertices from a same part. But since there are no edges between vertices of the same part, this set cannot form a clique. We can therefore say that such a 5-partition of the vertices is a proof that the graph contains no clique of size six. (See Figure 1.2 for an example of such a partition, where each part is indicated by a different colour.) Note that this proof required a slightly more advanced method of reasoning.

Although for our concrete example it was possible to find a short proof that there were no cliques of size six, it is not clear that this would be the case for every graph. In fact, the problem of determining whether a graph contains a clique of size k—referred to as the k-clique problem—is an NP-complete problem [Kar72]. What this means is that, on the one hand, if the graph does contain a k-clique then there is a short proof of this—identifying the clique, for example—but on the other, if the graph has no k-clique then we cannot guarantee that this fact has a short proof—in some cases there are, but in others we simply do not know. Additionally, if you could demonstrate that there are k-clique free graphs for which no short proofs of k-clique freeness exist, then you would be proving that the whole family of NP-complete problems do not always have short proofs (and you would win a million dollars!).

Now the reader might be thinking that this is all very interesting from a theoretical point of view, but are there any practical applications of proof complexity? Indeed there are: lengths of proofs are intimately related with running times of algorithms. For example, the execution trace of an algorithm that finds an optimal solution to a problem can be seen as a proof—formalisable in some system—that the solution is indeed optimal. Therefore, studying particular proof systems helps us understand the behaviour of the class of algorithms that are based on this system. The most noticeable example of such relation is that of SAT-solvers, algorithms that determine the satisfiability of a propositional formula. All state-of-the-art SAT-solvers—which successfully solve industrial instances with millions of variables—are, at their core, based on the so-called *resolu-*

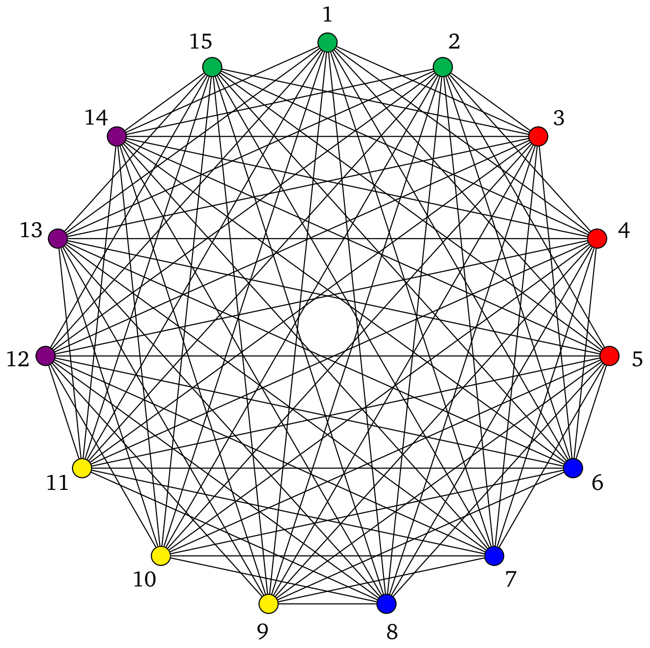


Figure 1.2: A 5-partition of the invitee graph

tion proof system and therefore proving lower bounds for resolution implies (ignoring pre-processing techniques) lower bounds on the running time of these algorithms.

Chapter 2

Background

In the beginning of the 20th century, the foundations of mathematics were strongly shaken by paradoxes and inconsistencies in the early attempts to clarify the basis on which mathematics was being built. Perhaps the most prominent inconsistency from that time is Russell's paradox "if R is the set of all sets that are not members of themselves, then $R \in R \iff R \notin R$", which merited the famous reply from Gottlob Frege: "Your discovery of the contradiction caused me the greatest surprise and, I would almost say, consternation, since it has shaken the basis on which I intended to build arithmetic" [VH67].

This concrete inconsistency in set theory was solved by adopting a certain axiomatic system, but several other fundamental problems remain open even today. In particular, David Hilbert's proposed solution to the crisis—to prove the consistency of complex systems in terms of simpler ones, so that the consistency of all mathematics would be reduced to basic arithmetic—was shown to be unattainable by Gödel's Incompleteness Theorem [Göd31], published in 1931.

Only a few years later, Alan Turing [Tur37] defined a mathematical model of computation, now called Turing machines, that allowed him to prove the unsolvability of Hilbert's *Entscheidungsproblem*—proved independently by Alonzo Church [Chu36]—by showing that the halting problem is undecidable. All these events led to an increased interest in understanding what can or cannot be proven in a certain language, and what can or cannot be computed.

In the late 1960s a new field emerged within the foundations of mathematics with the introduction of the notions of polynomial time algorithms, complexity classes and reductions between problems. The emphasis was now not only on what is computable, or what is provable, but on how efficient this computation can be, or how short these proofs can be. When analysing computations, this area is known as computational complexity, and when considering proofs, proof complexity. These are two closely related areas and, although our focus is the analysis of proofs, computational aspects will come up throughout this thesis.

As originally conceived by Stephen Cook and Robert Reckhow [CR79], propositional proof complexity is "the study of the size of the shortest proof of a propositional tautology in various proof systems as a function of the size of the tautology." A propositional tautology is a formula that evaluates to true for all possible assignments to the variables, for example, the law of excluded middle $x \lor \neg x$. A proof system is simply a sound system for proving tautologies. Perhaps the more natural ones are Frege systems [Fre93, CR79], which operate with Boolean expressions—built from variables and connectives such as $\{\neg, \land, \lor, \rightarrow\}$—and are defined in terms of a set of sound and implicationally complete inference rules and axioms. An example of such rule is *modus ponens* $\frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$.

When analysing a proof, the most important characteristic is its length, which is a lower bound on the time required to find, or even to verify, the proof. However, another very relevant measure is the space required to verify it—a lower bound on the memory needed for such a task. In this work, apart from studying proof length, we are also interested in understanding the relation between length and space; in particular, when optimising one measure leads inevitably to a blow-up in the other.

Before we formally define the formulas and proof systems that are most relevant for this thesis, we introduce *pebble games*. These games were first defined in [PH70] with the purpose of understanding space in computations and are also a very useful tool to study space and length-space trade-offs in proof complexity.

2.1 Pebble Games

Pebble games are played on directed acyclic graphs (DAG). A vertex in a DAG is a *source* if it has no incoming edges and is a *sink* if it has no outgoing edges. Given a DAG G with a unique sink, the *standard pebble game* [PH70] on G is a single-player game that is played with a set of pebbles. Initially, there are no pebbles on the graph, and at each step the player can either place a pebble on a vertex v whose immediate predecessors—denoted by $\mathsf{parents}(v)$—already have pebbles (in particular, the player can always place a pebble on a source) or remove a pebble from any vertex. The goal of the game is to place a pebble on the sink by using as few pebbles as possible. This simple game is a model of deterministic sequential computation, and has been used to study flowcharts and recursive schemata [PH70], register allocation [Set75] and time and space as Turing-machine resources [Coo74, HPV77].

By varying some of the rules of the game, it is possible to define pebble games that capture non-determinism (*black-white pebbling* [CS76]), parallelism (*parallel pebbling* [AS15]), and reversible computation (*reversible pebbling* [Ben89]). Applications of different variants of the game include—just to mention a few—algorithmic time and space trade-offs [Cha73], parallel time [DT85], communication complexity [RM99], monotone space complexity [CP14, FPRC13], cryptography [AS15, DNW05], energy dissipation during computation [Ben89], quantum computing [MSR$^+$18, BSD$^+$19] and proof complexity [BN08, BW01, BEGJ00].

In particular, in the last couple of decades, pebbling has played a key role in length-space trade-offs in proof complexity (see, e.g., [Nor13]). The results presented in this thesis build and extend on prior applications of pebbling, and all the flavours mentioned above—standard, black-white, parallel and reversible—will play an important role in some context.

To get a unified description of all types of the pebble game we will mention in this thesis, it is convenient to define pebbling as follows.

Definition 2.1.1 (Pebble games). Let $G = (V, E)$ be a DAG with a unique sink vertex z. The black-white pebble game on G is the following one-player game. At any time i, we have a *black-white pebbling configuration* $\mathbb{P}_i = (B_i, W_i)$ of black pebbles B_i and white pebbles W_i on the vertices of G, at most one pebble per vertex. The rules of how a pebble configuration $\mathbb{P}_{i-1} = (B_{i-1}, W_{i-1})$ can be changed to $\mathbb{P}_i = (B_i, W_i)$ are as follows:

1. A black pebble may be placed on a vertex v only if all immediate predecessors of v are covered by pebbles in both $\mathbb{P}_{i-1}$ and $\mathbb{P}_i$, i.e.,

   $$v \in (B_i \setminus B_{i-1}) \Rightarrow \mathsf{parents}(v) \subseteq \mathbb{P}_{i-1} \cap \mathbb{P}_i \ .$$

   Note that, in particular, a black pebble can always be placed on a source vertex.

2. A black pebble on any vertex in $\mathbb{P}_{i-1}$ can be removed in $\mathbb{P}_i$.

3. A white pebble can be placed on any vertex in $\mathbb{P}_i$.

4. A white pebble on a vertex v in $\mathbb{P}_{i-1}$ may be removed in $\mathbb{P}_i$ only if all immediate predecessors of v are covered by pebbles in both $\mathbb{P}_{i-1}$ and $\mathbb{P}_i$, i.e.,

   $$v \in (W_{i-1} \setminus W_i) \Rightarrow \mathsf{parents}(v) \subseteq \mathbb{P}_{i-1} \cap \mathbb{P}_i \ .$$

   In particular, a white pebble can always be removed from a source vertex.

A *(complete) pebbling* $\mathcal{P}$ of G is a sequence $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_\tau)$ where $\mathbb{P}_0 = \mathbb{P}_\tau = (\emptyset, \emptyset)$, every configuration $\mathbb{P}_i$ can be obtained from $\mathbb{P}_{i-1}$ using the rules 1–4 and $z \in \bigcup_{i=0}^{\tau}(B_i \cup W_i)$ (that is, at some point the sink is pebbled).

A pebbling is *sequential* if, for all $i \in [\tau]$, $\mathbb{P}_{i-1}$ and $\mathbb{P}_i$ differ by only one pebble, in other words, only one application of a single rule 1–4 is allowed at every step. In a *parallel* pebbling an arbitrary number of applications of the rules 1–4 can be made to get from $\mathbb{P}_{i-1}$ to $\mathbb{P}_i$ (but observe that all moves must be legal with respect to $\mathbb{P}_{i-1}$).

A *black pebbling* (or *standard pebbling*) is a pebbling where $W_i = \emptyset$ for all $i \in [\tau]$. A more restricted game is *reversible pebbling* that can be defined as a black pebbling in which removals have to obey rule 4, that is, a pebble on a vertex v in $\mathbb{P}_{i-1}$ may be removed in $\mathbb{P}_i$ only if all immediate predecessors of v are covered by pebbles in both $\mathbb{P}_{i-1}$ and $\mathbb{P}_i$.

The *time* of a pebbling $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_\tau)$ is $t(\mathcal{P}) = \tau$; the (maximum) *space* is $s(\mathcal{P}) = s = \max_{i \in [\tau]} |B_i| + |W_i|$; and the *cumulative space* is $c(\mathcal{P}) = c = \sum_{i \in [\tau]} |B_i| + |W_i|$ (where we note that $c \leq st$).

2.2 Formulas

Pebble games can be encoded in CNF by so-called *pebbling formulas* [BW01]. These formulas play an important role in four of the five papers in this thesis. The other family of formulas that will be relevant to us are clique formulas. In this section we establish some basic terminology and then define these two families.

A *literal* over a Boolean variable x is either the variable x itself (a *positive literal*) or its negation $\neg x$ (a *negative literal*), sometimes denoted $\overline{x}$. A *clause* $C = \ell_1 \vee \cdots \vee \ell_w$ is a disjunction of literals. We write $\perp$ to denote the empty clause without any literals. A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We think of clauses and CNF formulas as sets: order is irrelevant and there are no repetitions. Given a CNF formula F, we refer to the clauses in F as *axioms*.

One way of proving a tautology F is to determine that $\neg F$ leads to a contradiction. This view is often more convenient to consider and that is why we define the negation of tautologies—that is, unsatisfiable formulas—below, and later on will refer to proofs of unsatisfiablity of these formulas.

2.2.1 Pebbling Formulas

Let G be a DAG with a single sink z, let $S \subseteq V(G)$ be the sources of G and recall that $\mathsf{parents}(v)$ denote the immediate predecessors of the vertex v. The pebbling formula on G, denoted Peb_G, is defined over variables x_v for $v \in V(G)$ and encodes that sources are true

$$x_s \qquad\qquad s \in S \ , \tag{2.1a}$$

and that truth propagates from predecessors to successors

$$x_v \vee \bigvee_{u \in \mathsf{parents}(v)} \neg x_u \qquad\qquad v \in V(G) \ , \tag{2.1b}$$

but that the sink is false

$$\neg x_z \ . \tag{2.1c}$$

Note that if G has n vertices, the formula Peb_G is an unsatisfiable CNF formula over n variables with $n+1$ clauses.

2.2.2 Clique Formulas

Given a graph G we can encode a CNF formula $Clique(G,k)$ asserting that G contains a k-clique by claiming that for $i \in [k]$ there exists an ith clique member

$$\bigvee_{v \in V} x_{v,i} \qquad\qquad i \in [k] \ , \tag{2.2a}$$

and that two non-neighbouring vertices cannot both be in the clique

$$\neg x_{u,i} \vee \neg x_{v,j} \qquad\qquad i,j \in [k], i \neq j, u,v \in V, \{u,v\} \notin E \ , \tag{2.2b}$$

where the intended meaning of the variables is that $x_{v,i}$ is true if vertex v is the ith clique member. We could also add functionality axioms stating that at most one vertex is the ith clique member

$$\neg x_{u,i} \lor \neg x_{v,i} \qquad\qquad i \in [k], u, v \in V, u \neq v \ . \tag{2.2c}$$

We refer to (2.2b) as *edge axioms*, (2.2a) as *clique axioms* and (2.2c) as *functionality axioms*. Note that $Clique(G,k)$ is satisfiable if and only if G contains a k-clique, and that this is true even if clauses (2.2c) are omitted.

2.3 Proof Systems

In this section, we define the proof systems that are most relevant to the result we include in this thesis. The first two—resolution and cutting planes—are dynamic proof systems in the sense that the proof is presented step-by-step with intermediate derivations. The third and last proof system—Nullstellensatz—is a static proof system: the proof is presented in one shot. In the dynamic setting, it is natural to analyse length and space of proofs, while in the static setting other complexity measures will show up.

2.3.1 Resolution

Resolution is undoubtedly the most well-studied system in proof complexity. A *resolution refutation* $\pi : F \vdash \bot$ of an unsatisfiable CNF formula F—or a *resolution proof* for (the unsatisfiability of) F—is an ordered sequence of clauses $\pi = (D_1, \ldots, D_\tau)$ such that $D_\tau = \bot$ is the empty clause containing no literals, and for each $i \in [\tau]$ either D_i is a clause in F or there exist $j < i$ and $k < i$ such that D_i is derived from D_j and D_k by the *resolution rule*

$$\frac{B \lor x \qquad C \lor \neg x}{B \lor C} \ , \tag{2.3}$$

for $D_i = B \lor C,\ D_j = B \lor x,\ D_k = C \lor \neg x$. We refer to $B \lor C$ as the *resolvent* of $B \lor x$ and $C \lor \neg x$ over x, and to x as the *resolved variable*. The *length* of a resolution refutation $\pi = (D_1, \ldots, D_\tau)$ is τ.

It is often useful to consider every non-axiom clause in the proof as having a reference to the two clauses from which it was derived; in particular, since a same clause may appear more than once in a refutation. For this reason, it is convenient to view a resolution refutation $\pi = (D_1, \ldots, D_\tau)$ as a labelled DAG with the set of nodes $\{1, \ldots, L\}$ and edges (j,i), (k,i) for each application of the resolution rule deriving D_i from D_j and D_k. Each node i in this DAG is labelled by its associated clause D_i, and each non-source node is also labelled by the resolved variable in its associated derivation step in the refutation. Note the the number of nodes in the graph is equal to the length of the refutation. A resolution refutation is called *regular* if along any source-to-sink path in its associated DAG every variable is resolved at most once, and it is called *tree-like* if the underlying graph is a tree.

In order to study space of resolution derivations, we consider the proof from the *blackboard model* perspective. Following [ABRW02, ET01], a resolution refutation $\pi : F \vdash \bot$ can be defined as a sequence of *configurations* $\pi = (\mathbb{C}_0, \ldots, \mathbb{C}_{\tau'})$, where each configuration is a set of clauses such that $\mathbb{C}_0 = \emptyset$, $\bot \in \mathbb{C}_{\tau'}$, and for all $i \in [\tau']$ we obtain $\mathbb{C}_i$ from $\mathbb{C}_{i-1}$ by applying exactly one of the following type of rules:

Axiom download $\mathbb{C}_i = \mathbb{C}_{i-1} \cup \{A\}$ for $A \in F$;

Inference $\mathbb{C}_i = \mathbb{C}_{i-1} \cup \{D\}$ for D derived by the resolution rule from clauses in $\mathbb{C}_{i-1}$;

Erasure $\mathbb{C}_i \subsetneq \mathbb{C}_{i-1}$.

The (maximum) *space* of π is $\max\{|\mathbb{C}_i| : \mathbb{C}_i \in \pi\}$ and the *cumulative space* is $\sum_{\mathbb{C}_i \in \pi} |\mathbb{C}_i|$. As defined above, the length of a resolution refutation is the number of axiom downloads and inference steps. In some cases, however, we consider length to be the total number of steps τ'—including erasures—but this is a minor difference since it differs by at most a factor 2.

2.3.2 Cutting Planes

The *cutting planes (CP)* proof system, introduced in [CCT87] as a formalization of the integer linear programming algorithm in [Gom63, Chv73], operates with linear inequalities and inferences that are sound over integer solutions. The derivation rules in cutting planes are *linear combinations*

$$\frac{\sum_i a_i x_i \geq A \qquad \sum_i b_i x_i \geq B}{\sum_i (ca_i + db_i) x_i \geq cA + dB} \tag{2.4}$$

and *division*

$$\frac{\sum_i c a_i x_i \geq A}{\sum_i a_i x_i \geq \lceil A/c \rceil}\,, \tag{2.5}$$

where a_i, b_i, c, d, A, and B are all integers and $c, d \geq 0$.

In order to use cutting planes to refute unsatisfiable CNF formulas, we translate clauses C to linear inequalities $L(C)$ by identifying the clause $\bigvee_{j \in P} x_j \vee \bigvee_{j \in N} \neg x_j$ with the inequality $\sum_{j \in P} x_j + \sum_{j \in N} (1 - x_j) \geq 1$ and include *variable axioms* $x \geq 0$ and $-x \geq -1$ ensuring all variables take $\{0, 1\}$ values. The goal, then, is to derive the inequality $0 \geq 1$ which is a proof of unsatisfiablity. It is not hard to show that CP can polynomially simulates resolution, and, therefore, we can conclude that deriving $0 \geq 1$ is possible if and only if no $\{0, 1\}$-assignment satisfies all constraints.

Similarly to the blackboard model perspective in resolution, a *cutting planes (CP)* proof of unsatisifiability of a CNF formula F, or refutation of F, can be defined as a sequence of configurations $(\mathbb{L}_0, \ldots, \mathbb{L}_\tau)$ where configurations are sets of linear inequalities $\sum_j a_j x_j \geq c$ with $a_j, c \in \mathbb{Z}$ such that $\mathbb{L}_0 = \emptyset$, the inequality $0 \geq 1$ occurs in $\mathbb{L}_\tau$, and for $t \in [\tau]$ we obtain $\mathbb{L}_i$ from $\mathbb{L}_{i-1}$ by one of the following rules:

Axiom download $\mathbb{L}_i = \mathbb{L}_{i-1} \cup \{L\}$ for L being either the encoding $L(C)$ of an axiom $C \in F$ or a variable axiom $x_j \geq 0$ or $-x_j \geq -1$ for any variable x_j.

Inference $\mathbb{L}_i = \mathbb{L}_{i-1} \cup \{L\}$ for L inferred from inequalities in $\mathbb{L}_{i-1}$ by one of the cutting planes derivation rules.

Erasure $\mathbb{L}_i \subsetneq \mathbb{L}_{i-1}$.

The *length* of a CP refutation is the number of derivation steps τ. The *formula space* (or *line space*) of a configuration $\mathbb{L} = \left\{\sum_j a_{i,j}x_{i,j} \geq c_i \,\middle|\, i \in [s]\right\}$ is the number of inequalities s in it, and the *total space* of $\mathbb{L}$ is $\sum_{i\in[s]}\bigl(\log|c_i|+\sum_j \log|a_{i,j}|\bigr)$. We obtain the formula space or total space of a refutation by taking the maximum over all configurations in it.

Note that the total space depends on the magnitude of coefficients of the inequalities in the proof. In this setting, it is natural to ask whether cutting planes refutations require large coefficients to realise the full power of the proof system. In order to formalise this question, we define CP* to be the subsystem of cutting planes with the restriction that all coefficients in the proof are *polynomially bounded* or, in other words, a cutting planes refutation π of a formula F with n variables is a CP* refutation if the largest coefficient in π has magnitude $\mathsf{poly}(n)$.

2.3.3 Hilbert's Nullstellensatz

As a proof system, Hilbert's Nullstellensatz—often referred to simply as Nullstellensatz—provides a certificate that a set of polynomials do not have a common root. More formally, let $\mathbb{F}$ be a field, and let $\mathcal{P} = \{p_1 = 0, p_2 = 0, \ldots, p_m = 0\}$ be an unsatisfiable system of polynomial equations in $\mathbb{F}[x_1, x_2, \ldots, x_n]$. A *Nullstellensatz refutation* of $\mathcal{P}$ is a sequence of polynomials $q_1, q_2, \ldots, q_m \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that

$$\sum_{i=1}^{m} p_i q_i = 1$$

where the equality is syntactic.

The *degree* of the refutation is $\max_i \deg(p_i q_i)$; the *Nullstellensatz degree* of $\mathcal{P}$ is the minimum degree of any Nullstellensatz refutation of $\mathcal{P}$. We define the *size* of the refutation to be the total number of monomials encountered when all products of polynomials are expanded out as linear combinations of monomials.

To analyse Nullstellensatz refutations of CNF formulas, we consider the standard encoding of each clause $C = \bigvee_{x\in P} x \vee \bigvee_{x\in N} \neg x$ as the polynomial equation

$$\mathcal{E}(C) \equiv \prod_{x\in P}(1-x)\prod_{x\in N} x \ .$$

Observe that $\mathcal{E}(C) = 0$ is satisfied (over $0/1$ assignments to z_i) if and only if the corresponding assignment satisfies C. For a CNF formula F, we abuse notation and let

$\mathcal{E}(F) = \{\mathcal{E}(C) : C \in F\} \cup \{x_i^2 - x_i\}_{i\in[n]}$, where the second set of polynomial equations restricts the x_i inputs to $\{0, 1\}$ values. We note that—as shown in [BIK+97]—if $\mathcal{P}$ is a system of polynomial equations over $\mathbb{F}[x_1, \ldots, x_n]$ with no $\{0, 1\}$ solutions, then there exists a Nullstellensatz refutation of $\mathcal{P} \cup \{x_i^2 - x_i = 0\}_{i\in[n]}$.

2.4 Communication Complexity

Communication complexity plays an important role in some of our length-space trade-off results in proof complexity. In this section, we define the communication model of interest for us and then explain in general lines how we can obtain lower bounds in proof complexity via lower bounds in communication complexity.

The classical communication model of [Yao79] consists of two parties, traditionally referred to as *Alice* and *Bob*, each holding a separate input $x \in X$ and $y \in Y$, respectively. They both have knowledge of a function $f : X \times Y \to \{0, 1\}$ and their goal is to compute $f(x, y)$ by exchanging the minimum number of bits. The communication is guided by a *protocol* that the parties agree on before receiving their inputs. The protocol can be viewed as a binary tree where Alice and Bob start at the root and together trace a path to a leaf depending on what is spoken: every node in the tree specifies who is going to speak, the value of the spoken bit—which is only a function of the node they are currently at and of the input x if Alice speaks or y if Bob does—determines which of the successors of the node they will consider next, and leaves are labelled by correct values $f(x, y)$. The cost of a protocol is the maximum number of bits communicated on any input, that is, the length of the longest root-to-leaf path in the protocol tree. Another measure that will be of interest to us is the number of rounds, which is defined as the maximum number of alternations between Alice and Bob speaking.

For our applications, we also need to study the more general *real communication* model in [Kra98], where Alice and Bob interact via a referee. In order to introduce the concept of rounds in this model, it is convenient to describe the protocol as a (non-binary) tree, where at node v in the protocol tree Alice and Bob send k_v real numbers $\phi_{v,1}(x), \ldots, \phi_{v,k_v}(x)$ and $\psi_{v,1}(y), \ldots, \psi_{v,k_v}(y)$, respectively, to the referee. The referee announces the results of the comparisons $\phi_{v,i}(x) \leq \psi_{v,i}(y)$ for $i \in [k_v]$ as a k_v-bit binary string, after which the players move to the ith successor node in the protocol tree. As in the classical model, leaves are labelled by correct values $f(x, y)$. The number of rounds r of a protocol is the depth of the tree and the cost c is the maximum number of comparisons made by the referee for any input. It is easy to see that this model can simulate standard deterministic communication (for instance, if Alice wants to send a message, she sends the complement of that message to the referee and Bob sends a list of the same length with all entries $1/2$), and is in fact strictly stronger (since the equality function can be solved with just two bits of communication).

Communication problems as defined above can be extended to relations in the obvious way: for any relation $S \subseteq X \times Y \times Q$, the communication problem for S is one in which Alice is given $x \in X$, Bob is given $y \in Y$, and they are required to communicate to find some q such that $(x, y, q) \in S$. For our applications in proof complexity, we are interested in a specific type of relation that arises from a *total search problem*, defined as a relation $\mathcal{S} \subseteq \mathcal{I} \times \mathcal{O}$ such that for all $z \in \mathcal{I}$ there is an $o \in \mathcal{O}$ such that $(z, o) \in \mathcal{S}$. Intuitively, S represents the computational task in which we are given an input $z \in \mathcal{I}$ and would like to find an output $o \in \mathcal{O}$ that satisfies $(z, o) \in \mathcal{S}$. In proof complexity, an important example of a total search problem is the *falsified clause search problem*. Given a CNF formula F over variables $z_1, \ldots, z_n$, the falsified clause search problem $Search(F) \subseteq \{0,1\}^n \times F$ contains the tuple (z, C) if and only if the clause C is falsified by the assignment z.

To turn this into a communication problem, we either partition the set of variables into two sets or we compose it with an *inner function*, also referred to as a *gadget*, $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{I}$. Given a search problem $\mathcal{S} \subseteq \mathcal{I}^n \times \mathcal{O}$ and a function $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{I}$, we define the *composition* $\mathcal{S} \circ g^n \subseteq \mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{O}$ in the natural way: $(x_1 \ldots x_n, y_1 \ldots y_n, o) \in \mathcal{S} \circ g^n$ if and only if $(g(x_1, y_1) \ldots g(x_n, y_n), o) \in \mathcal{S}$. We sometimes write $S \circ g$ instead of $S \circ g^n$ if n is clear from the context.

Before ending this section, we record an observation (which can be found, e.g., in [HN12]) that is important for some of our applications. To make the statement more comprehensible, we explain in general terms a proof strategy for obtaining length-space trade-offs in proof complexity. Given a CNF formula F and a gadget g, we consider the so-called *lifted CNF formula* $F \circ g$, which has a natural partition of variables between Alice and Bob. We can then show that a short, space-efficient refutation of the formula $F \circ g$ can be used to construct an efficient protocol for $Search(F \circ g)$. While it is not immediately clear how to prove lower bounds for $Search(F \circ g)$, we can prove lower bounds for the related composed search problem $Search(F) \circ g$, for some F and some g, via the so-called *lifting theorems*. Despite the fact that $Search(F) \circ g$ is not the same problem as $Search(F \circ g)$, we can reduce the former to the latter.

Observation 2.4.1. *For any unsatisfiable CNF F and any Boolean gadget g, a communication protocol for $Search(F \circ g)$ can be adapted to a communication protocol for $Search(F) \circ g$ in the same model and with the same parameters.*

2.5 Circuit Complexity

As a by-product of the techniques developed for proof complexity results, we also obtain some results in monotone circuit complexity. A Boolean circuit $\mathcal{C}$ is a single sink DAG where each non-source node—usually referred to as a *gate*—is labelled by AND, OR, or NOT, with the restriction that NOT gates have in-degree, or *fan-in*, 1. We say $\mathcal{C}$ computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if $\mathcal{C}$ has n sources, each labelled by an input bit, and for all $x \in \{0,1\}^n$, the circuit on input x evaluates to $f(x)$. The *size* of the circuit is the number of gates and the *depth* is the length of a longest path from a source to the sink.

A formula is a circuit in which all gates have out-degree, or *fan-out*, at most 1, and a monotone Boolean circuit is a circuit with no NOT gates. Monotone real circuits, which were introduced by Pudlák [Pud97], are a generalization of monotone Boolean circuits where each gate is allowed to compute any non-decreasing real function of its inputs, but the inputs and output of the circuit are Boolean.

Chapter 3

Contributions

3.1 Clique is Hard on Average for Regular Resolution

Summary of *Clique is Hard on Average for Regular Resolution* by Albert Atserias, Ilario Bonacina, Susanna F. de Rezende, Massimo Lauria, Jakob Nordström, and Alexander Razborov [ABdR+18]

In Paper A, we study regular resolution refutations of the k-clique formula on Erdős-Rényi random graphs $\boldsymbol{G} \sim \mathscr{G}(n, p)$—graphs on n vertices where every edge is present with probability p. We prove an $n^{\Omega(k)}$ average-case lower bound for such refutations, when the graph is sampled with appropriate edge density. This implies a lower bound on the running time of most state-of-the-art algorithms used in practice to solve the k-clique problem (see, e.g., [Pro12, McC17] for a survey on such algorithms) since the underlying method of reasoning can be captured by regular resolution. This lower bound is tight up to multiplicative constants in the exponent since regular resolution can solve the problem in time $n^{O(k)}$ simply by checking whether any of the $\binom{n}{k}$ many sets of vertices of size k forms a clique.

Theorem 3.1.1 (Informal). *For any integer $k \ll \sqrt[4]{n}$, given an n-vertex graph G sampled at random from the Erdős-Rényi model with the appropriate edge density, regular resolution asymptotically almost surely requires length $n^{\Omega(k)}$ to certify that G does not contain a k-clique.*

As mentioned in Chapter 1, determining whether a graph contains a clique of size k is an NP-complete problem. In fact, it is one of the—now classical—problems that appeared in Karp's list of 21 NP-complete problems [Kar72] and is considered one of the most basic computational problems on graphs. Although NP-completeness only indicates that the problem is hard in the worst case, the k-clique problem appears to be hard also on average—we know of no efficient algorithms that with high probability can decide if an Erdős-Rényi random graph with appropriate edge densities has a k-clique [Kar76, Ros10].

To prove our average case lower bound, it is crucial to identify what combinatorial structures of randomly sampled graphs account for the hardness of refuting the k-clique formula on these graphs. We define such a structural property, which we call *clique-denseness*, and then divide the proof into two parts. On the one hand, we prove that clique-dense graphs are hard to refute; on the other, we show that Erdős-Rényi random graphs with appropriate edge density are asymptotically almost surely clique-dense. The latter argument turns out to be much more involved than most arguments of this kind, but, in a nutshell, it involves Chernoff bounds, a somewhat elaborate construction of "bad sets", and a delicate balancing of parameters.

Given the correct definition of clique-denseness, the first part of the proof is the more challenging one. It is based on a bottleneck counting argument similar to [Hak85] but with a slight twist. From a bird's eye view, the classical argument requires definitions of a set of *bottleneck nodes* and of a distribution of random paths on the DAG underlying the proof. The next step is to show that every path from the distribution contains some bottleneck node, but at the same time that it is highly unlikely that a random path contains any particular bottleneck node. By a union bound argument, we can then conclude that there must be many bottleneck nodes and so the proof must be long. The twist in our argument, introduced already in [RWY02], is to define pairs of bottleneck nodes, instead of single nodes.

There are several steps in the bottleneck argument that rely on the resolution refutation being regular. In order to strengthen this result to hold also for (general) resolution, new ideas seem to be needed. However, the abstract combinatorial property of graphs we identify does not in itself have any connection with regularity. We believe an important contribution of this paper is to have identified this structural property of random graphs, which might very well be useful to extend this results to stronger proof systems.

3.2 How Limited Interaction Hinders Real Communication

Summary of *How Limited Interaction Hinders Real Communication (and What it Means for Proof and Circuit Complexity)* by Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals [dRNV16]

In Paper B, we prove length-space trade-offs for cutting planes (CP), where upper bounds hold for derivations with constant size coefficients, and the lower bounds apply even for derivations with unbounded coefficients. These results are the first *true trade-offs*—in the sense that there are refutations both of small size and small space, only not simultaneously—for which the small space refutations have polynomially bounded coefficients. These are also the first trade-offs to hold uniformly for resolution, polynomial calculus and cutting planes, thus capturing the main methods of reasoning used in current state-of-the-art SAT solvers.

Below, we state two examples of trade-offs we obtain. The first one is a "robust trade-off": there are proofs in linear length (which require large space), proof in polylogarithmic space (which are long) and even if you allow polynomial space $n^{1/10-\epsilon}$ there is no polynomial length proof.

Theorem 3.2.1. *There is a family of CNF formulas over* $\Theta(n)$ *clauses that have*

- *a CP proof with constant-size coefficients of length* $O(n)$ *and*
- *a CP proof with constant-size coefficients in space* $\operatorname{polylog} n$*, but*
- *any CP proof, even with coefficients of unbounded size, in space* $n^{1/10-\epsilon}$ *requires superpolynomial length.*

The second trade-off holds over a smaller space range, but restricting space causes the length to go from linear to exponential.

Theorem 3.2.2. *There is a family of CNF formulas over* $\Theta(n)$ *clauses that have*

- *a CP proof with constant-size coefficients of length* $O(n)$ *and*
- *a CP proof with constant-size coefficients in space* $O(n^{1/40})$*, but*
- *any CP proof, even with coefficients of unbounded size, in space* $n^{1/20-\epsilon}$ *requires length* $\exp(\Omega(n^{1/40}))$*.*

As a by-product of the techniques we developed to show the proof complexity result, we were able to separate $\mathsf{monotone\text{-}AC}^{i-1}$ from $\mathsf{monotone\text{-}NC}^{i}$, solving an open problem in monotone circuit complexity [GS92, Joh01].

Theorem 3.2.3. *For every* $i \in \mathbb{N}$ *there is a Boolean function over* n *variables that can be computed by a monotone circuit of depth* $\log^i n$*, fan-in* 2*, and size* $O(n)$*, but for which every monotone circuit of depth* $O(\log^{i-1} n)$ *requires superpolynomial size.*

In addition, we prove an exponential separation of the $\mathsf{monotone\text{-}AC}$ hierarchy.

Theorem 3.2.4. *For every* $i \in \mathbb{N}$ *there is a Boolean function over* n *variables that can be computed by a monotone circuit of depth* $\log^i n$*, fan-in* $n^{4/5}$*, and size* $O(n)$*, but for which for every* $q \in \mathbb{N}$ *every monotone circuit of depth* $q \log^{i-1} n$ *requires size* $\exp\left(\Omega\left(n^{\frac{1}{11q}}\right)\right)$*.*

Let us now make a tour d'horizon of the proof of these theorems, focusing on the proof complexity results since the monotone circuit ones follow from a similar and even slightly simpler argument.

The formulas we consider are pebbling formulas composed with the *indexing gadget* $g : [\ell] \times \{0,1\}^\ell$ of length ℓ, defined as $g(x,y) = y_x$. The proof of the lower bounds can be viewed as a proof by contradiction via a chain of reductions that goes through communication complexity, decision trees and the so-called Dymond–Tompa [DT85] game played on a DAG.

We suppose there is a short space-efficient cutting planes proof for the lifted CNF formula $F \circ g$. As was made explicit in [HN12], we can obtain from an efficient proof an efficient communication protocol for the falsified clause search problem *Search*$(F \circ g)$. The exact model of communication needed depends on the proof system. For cutting planes, standard deterministic communication is not enough, so we use real communication [Kra98]. We make the additional simple but crucial observation that the protocol obtained from a short proof is also round-efficient. By Observation 2.4.1, we get a protocol for *Search*$(F) \circ g$ that is both communication- and round-efficient.

We now arrive at the main technical contribution of this paper. We generalize the lifting theorem in [RM99, GPW15] to preserve rounds and use ideas from [BEGJ00] to adapt the protocol to hold for real communication. From a real communication protocol with few rounds and small cost, we obtain a *parallel decision tree* with small depth and few queries. Parallel decision trees were introduced by Valiant [Val75b] and differ from decision trees in that they are not necessarily binary: at every node the tree is allowed to query any number of bits.

Theorem 3.2.5. *Let S be a search problem and let g be the indexing gadget. If there is a real communication protocol for $S \circ g^n$ with communication c and r rounds, then there is a decision tree for S with $\mathrm{O}(c/\log n)$ queries and depth r.*

To obtain contradiction, the last step is to prove lower bounds for the falsified search problem on pebbling formulas on a DAG G for parallel decision trees. It is quite straightforward to see that this problem is equivalent to the (parallel version of the) Dymond–Tompa game played on G. We then modify graph constructions of [LT82] to obtain the desired Dymond–Tompa lower bound.

3.3 Lifting with Simple Gadgets and Applications

Summary of *Lifting with Simple Gadgets and Applications to Circuit and Proof Complexity* by Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals [dRMN$^+$19]

In Paper C, we address the question of whether cutting planes (CP)—which allows the inequalities to use coefficients of arbitrary size—is polynomially equivalent to the variant in which the coefficients are polynomially bounded (CP*). This question was raised in [BC96] and has evaded all solution attempts so far. In this work, we finally make progress by exhibiting a family of formulas that have short constant-space CP proofs but that in small space require exponential length CP* proofs.

Theorem 3.3.1. *There is a family of CNF formulas of size N that have cutting planes refutations of length $\tilde{O}(N^2)$ and space $O(1)$, but for which any refutation of length L and space s with polynomially bounded coefficients must satisfy $s \log L = \tilde{\Omega}(N)$.*

To attain such a separation, we exploit the fact that only with high-weight coefficients it is possible to encode several equalities with a single equality (or with two inequalities). To take advantage of this observation, we concoct a formula that cannot be refuted in small length unless reasoning about several equalities at the same time. These are based on pebbling formulas that have the very useful property that, independently of the DAG they are defined on, they can always be refuted in small length in resolution by reasoning about several literals at once. Moreover, pebbling formulas that are defined on DAGs that require large space to be pebbled, when composed with the XOR function, in resolution require reasoning about large conjunctions of clauses at the same time [BN08].

Instead of XOR, we compose pebbling formulas with the equality gadget (EQ). By reasoning about a conjunction of several equalities, CP can simulate the short resolution refutation of pebbling formulas in constant space and quadratic length. To obtain the separation, we need to prove that CP* cannot refute these formulas in small space and small length simultaneously. The first step is to reduce the problem to a communication complexity problem: first using the connection between proofs and protocols as per [HN12]—where we note that since CP* has bounded coefficients, this can be done efficiently in the deterministic communication model—and then applying Observation 2.4.1 to obtain a composed search problem.

The reason we cannot apply previously known lifting theorems is that the composed search problem we obtain is of the form $\mathit{Search}(F) \circ \mathrm{EQ}$ and the known lifting theorems only apply for certain gadgets, such as indexing and inner product. Moreover, as pointed out in [LM19], it is provably not possible to lift decision tree complexity to communication complexity with the equality gadget. Loff and Mukhopadhyay [LM19] are able to get around this problem by proving a lifting theorem from a stronger complexity measure, namely the 0-query complexity. Unfortunately, this is not the right measure for us, since pebbling formulas have very small 0-query complexity.

We address this issue by considering a lifting theorem of Pitassi and Robere [PR18] that lifts Nullstellensatz degree and generalizing it to use any gadget of high-enough rank; in particular, the equality gadget.

Theorem 3.3.2. *Let F be a CNF over n variables, let $\mathbb{F}$ be any field, and let g be any gadget of rank at least r. Then the deterministic communication complexity of $\mathit{Search}(F \circ g^n)$ is at least $\mathsf{NS}_{\mathbb{F}}(F)$, the Nullstellensatz degree of F, as long as $r \geq cn/\mathsf{NS}_{\mathbb{F}}(F)$ for some large enough constant c.*

Finally we must prove a Nullstellensatz degree lower bound from pebbling formulas by showing that this measure is exactly equal to the reversible pebbling cost of the underlying graph.

Lemma 3.3.3. *For any field $\mathbb{F}$ and any directed acyclic graph G the Nullstellensatz degree of Peb_G is equal to the reversible pebbling cost of G.*

By considering graphs that have high reversible pebbling cost, we get the space-time lower bound for CP* and Theorem 3.3.1 follows.

Our new lifting also enabled us to prove the first explicit separation between monotone real and monotone Boolean formulas. Prior to this, only a non-explicit separation was known [Ros97].

Theorem 3.3.4. *There is an explicit family of functions f_n over $\mathrm{O}(n \operatorname{polylog} n)$ variables that can be computed by monotone real formulas of size $\mathrm{O}(n \operatorname{polylog} n)$ but for which every monotone Boolean formula requires size $2^{\Omega(n/\log n)}$.*

3.4 Nullstellensatz Size-Degree Trade-offs

Summary of *Nullstellensatz Size-Degree Trade-offs from Reversible Pebbling* by Susanna F. de Rezende, Or Meir, Jakob Nordström, and Robert Robere [dRMNR19]

In Paper D, we obtain strong size-degree trade-offs for Nullstellensatz. To understand how tight the trade-offs can be, we note that if a CNF formula over n variables has a Nullstellensatz refutation of degree d, then it is not hard to show that the formula has a Nullstellensatz refutation of simultaneous degree d and size $n^{\mathrm{O}(d)}$. It is not true, however, that if there is a refutation of size $n^{\mathrm{O}(d)}$ then there necessarily is a refutation of degree d. A natural question regarding the relation between size and degree is whether for any given function $d_1(n)$ there is a family of CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

1. F_n has a Nullstellensatz refutation of degree $d_1(n)$;
2. F_n has a Nullstellensatz refutation of (close to) linear size and degree $d_2(n) \gg d_1(n)$;
3. Any Nullstellensatz refutation of F_n of degree only slightly below $d_2(n)$ must have size nearly $n^{d_1(n)}$.

We present explicit constructions of such formulas in different parameter regimes. As an example of such a result, we prove the following trade-off.

Theorem 3.4.1 (Informal). *There is a family of 3-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

1. *There is a Nullstellensatz refutation of F_n of degree $d_1 = \mathrm{O}\big(\sqrt[6]{n}\log n\big)$.*
2. *There is a Nullstellensatz refutation of F_n of nearly linear size and degree $d_2 = \mathrm{O}\big(\sqrt[3]{n}\log n\big)$.*
3. *Any Nullstellensatz refutation of F_n of degree at most $\sqrt[3]{n}$ requires size at least $n^{\Omega(\sqrt[6]{n})}$.*

To obtain these trade-offs, we prove a surprisingly simple and tight correspondence between reversible pebbling and Nullstellensatz refutations of pebbling formulas. Moreover, this equivalence holds regardless of the field used in the refutation.

Theorem 3.4.2 (Informal). *Let G be a DAG with a single sink. There is a reversible pebbling strategy for G in time at most t and space at most s if and only if there is a Nullstellensatz refutation (in any field) for Peb_G of size at most $t+1$ and degree at most s.*

Given this relation, in order to obtain size-degree trade-offs for Nullstellensatz it is enough to exhibit graphs that have time-space trade-off for reversible pebbling. There are several known graphs that present trade-offs for standard pebbling, but these do not automatically translate to reversible pebbling since the upper bounds do not carry over—at least not verbatim. We focus on some of these graph constructions [CS80, CS82, LT82] and, although we are not able to achieve as tight results as in the standard pebbling, we nevertheless obtain strong reversible time-space trade-offs.

3.5 Cumulative Space in Black-White Pebbling and Resolution

Summary of *Cumulative Space in Black-White Pebbling and Resolution* by Joël Alwen, Susanna F. de Rezende, Jakob Nordström, and Marc Vinyal [AdRNV17]

In Paper E, we initiate the study of cumulative space in proof complexity. Cumulative space was introduced by [AS15] to study *memory-hard functions*—functions that require a large amount of memory to be computed—in cryptography. In some settings, however, only considering (maximum) memory required is not enough to guarantee security. For example, suppose an adversary has access to limited working memory and would like to perform a brute-force attack where she must realise several independent computations. Then, having to execute computations that only require a few specific instants of high memory usage is very different from having computations that require high memory usage for a significant fraction of the time. In the former case, it might be easy to parallelise computations by intercalating the peak memory usage; while in the latter, much of the work has to be done sequentially, and therefore the time needed becomes too large for the attack to be feasible.

In proof complexity, cumulative space could be very relevant in proof search. Algorithms, such as SAT-solvers, that find proofs of unsatisfiablity of formulas are constantly learning new clauses and, since memory is limited, must occasionally decide what clauses to forget. In order to run fast, the algorithm must be smart enough—and lucky enough—to not forget clauses which will be needed further on in the proof. To find a proof that has only one instance of large (maximum) space would mean to guess the right clauses to keep in memory that one time, while to find a proof that has large space during a significant fraction of the time would require repeatedly good luck.

While the concept of cumulative space seems to be as natural as maximum space and as time-space trade-offs, we are not aware of it having been studied in the context of proof complexity before. In this first paper on cumulative space in proof complexity we focus on the resolution proof system, as was also done in the first paper on (maximum) space complexity [ET01].

We define several versions of parallel resolution, with different degrees of parallelism and study the cumulative space complexity of these variants. We believe parallel models of resolution could be useful to analyse algorithms that attempt to parallelise state-of-the-art SAT solvers using so-called *conflict-driven clause learning (CDCL)* [BS97, MS99].

One of the resolution variants we consider is the semantic inference-parallel model, where axiom downloads must occur sequentially, but any set of clauses that follows semantically from the current configuration can be derived in one step. In this model, any unsatisfiable formula can be refuted in simultaneous linear length and quadratic cumulative space by downloading each axiom sequentially and then deriving contradiction in one step. We prove that there are formulas for which this cumulative space upper bound is tight.

Theorem 3.5.1 (Informal). *There is a family of CNF formulas $\{F_n\}_{n\in\mathbb{N}^+}$ of size $\Theta(n)$ that have syntactic sequential resolution refutations of length $\mathrm{O}(n)$, and hence also in maximum space $\mathrm{O}(n/\log n)$, but for which any semantic inference-parallel refutation requires cumulative space $\Omega(n^2)$.*

We also show that there are formulas that require small maximum space and nevertheless require almost maximum cumulative space.

Theorem 3.5.2 (Informal). *There is a family of CNF formulas $\{F_n\}_{n\in\mathbb{N}^+}$ of size $\Theta(n)$ that have syntactic sequential resolution refutations of length $\mathrm{O}(n)$ and also refutations in maximum space $\mathrm{O}(\log n)$, but for which any semantic inference-parallel refutation requires cumulative clause space $\Omega(n^2/\log n)$.*

Cumulative space can also be viewed as a stronger version of time-space trade-offs. Indeed, cumulative space is equal to time multiplied by average space, and so proving cumulative space lower bounds in particular implies time-space lower bounds. We exhibit formulas that present smooth trade-offs where bounded maximum space implies cumulative space lower bounds.

Theorem 3.5.3 (Informal). *There is a family of CNF formulas $\{F_n\}_{n\in\mathbb{N}^+}$ of size $\Theta(n)$ such that for any $s = \mathrm{O}\big(\sqrt{n}\big)$ the formula F_n has a resolution refutation of length $\mathrm{O}\big(n^2/s^2\big)$ and maximum space $\mathrm{O}(s)$, but any refutation of F_n in maximum space s requires cumulative clause space $\Omega(n^2/s)$.*

To obtain these results, the main tool used are pebble games—as is the case in the study of (maximum) space in resolution and of cumulative space in cryptography. The model of parallel black pebbling was introduced by [AS15] precisely for the applications in cryptography. For our proof complexity applications, we must consider instead appropriately defined versions of the black-white pebble game and then translate results to the different models of resolution by observing that the reductions in [BN08, BN11] also apply to our new setting. The cumulative space lower bounds and the trade-offs we obtain for black-white pebbling may be of independent interest.

Part II

Included Papers

A

Paper A

Clique is Hard for Regular Resolution

Albert Atserias, Ilario Bonacina, Susanna F. de Rezende,
Massiom Lauria, Jakob Nordström, and Alexander Razborov

Full length version of the article published in Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18), June 2018, pp. 866–877.

Abstract

We prove that for $k \ll \sqrt[4]{n}$ regular resolution requires length $n^{\Omega(k)}$ to establish that an Erdős–Rényi graph with appropriately chosen edge density does not contain a k-clique. This lower bound is optimal up to the multiplicative constant in the exponent, and also implies unconditional $n^{\Omega(k)}$ lower bounds on running time for several state-of-the-art algorithms for finding maximum cliques in graphs.

A.1 Introduction

Deciding whether a graph has a k-clique is one of the most basic computational problems on graphs, and has been extensively studied in computational complexity theory ever since it appeared in Karp's list of 21 NP-complete problems [Kar72]. Not only is this problem widely believed to be infeasible to solve exactly (unless $\mathsf{P} = \mathsf{NP}$) there does not even exist any polynomial-time algorithm for approximating the maximal size of a clique to within a factor $n^{1-\epsilon}$ for any constant $\epsilon > 0$, where n is the number of vertices in the graph [Hås99, Zuc07]. Furthermore, the problem appears to be hard not only in the worst case but also on average in the Erdős-Rényi random graph model—we know of no efficient algorithms for finding cliques of maximum size asymptotically almost surely on random graphs with appropriate edge densities [Kar76, Ros10].

In terms of upper bounds, the k-clique problem can be solved in time roughly n^k simply by checking if any of the $\binom{n}{k}$ many sets of vertices of size k forms a clique, which is polynomial if k is constant. This can be improved slightly to $O(n^{\omega k/3})$ using algebraic techniques [NP85], where $\omega \leq 2.373$ is the matrix multiplication exponent, although in practice such algebraic algorithms are outperformed by combinatorial ones [Vas09].

The motivating problem behind this work is to determine the exact time complexity of the clique problem when k is given as a parameter. As noted above, all known algorithms require time $n^{\Omega(k)}$. It appears quite likely that some dependence on k is needed in the exponent, since otherwise we have the parameterized complexity collapse $\mathsf{FPT} = \mathsf{W}[1]$ [DF95]. Even more can be said if we are willing to believe the Exponential Time Hypothesis (ETH) [IP01]—then the exponent has to depend linearly on k [CHKX04], so that the trivial upper bound is essentially tight.

Obtaining such a lower bound unconditionally would, in particular, imply $\mathsf{P} \neq \mathsf{NP}$, and so currently seems completely out of reach. But is it possible to prove $n^{\Omega(k)}$ lower bounds in restricted but nontrivial models of computation? For circuit complexity, this challenge has been met for circuits that are of bounded depth [Ros08] or are monotone [Ros14]. In this paper we focus on computational models that are powerful enough to capture several algorithms that are used in practice.

When analysing such algorithms, it is convenient to view the execution trace as a proof establishing the maximal clique size for the input graph. In particular, if this graph does not have a k-clique, then the trace provides an efficiently verifiable proof of the statement that the graph is k-clique-free. If one can establish a lower bound on the length of such proofs, then this implies a lower bound on the running time of the algorithm, and this lower bound holds even if the algorithm is a non-deterministic heuristic that somehow magically gets to make all the right choices. This brings us to the topic of *proof complexity* [CR79], which can be viewed as the study of upper and lower bounds in restricted nondeterministic computational models.

Using a standard reduction from k-clique to SAT, we can translate the problem of k-cliques in graphs to that of satisfiability of formulas in conjunctive normal form (CNF). If an algorithm for finding k-cliques is run on a graph G that is k-clique-free, then we can extract a proof of the unsatisfiability of the corresponding CNF formula—the k-clique formula on G—from the execution trace of the algorithm. Is it possible to show any nontrivial lower bound on the length of such proofs? Specifically, does the *resolution* proof system—the method of reasoning underlying state-of-the-art SAT solvers [BS97, MS99, MMZ$^+$01]—require length $n^{\Omega(k)}$, or at least $n^{\omega_k(1)}$, to prove the absence of k-cliques in a graph? This question was asked in, e.g., [BGLR12] and remains open.

The hardness of k-clique formulas for resolution is also a problem of intrinsic interest in proof complexity, since these formulas escape known methods of proving resolution lower bounds for a range of interesting values of k including $k = O(1)$. In particular, the interpolation technique [Kra97, Pud97], the random restriction method [BP96], and the size-width lower bound [BW01] all seem to fail.

To make this more precise, we should mention that some previous works do use the size-width method, but only for very large k. It was shown in [BIS07] that for $n^{5/6} \ll k \leq n/3$ resolution requires length $\exp\bigl(n^{\Omega(1)}\bigr)$ to certify that a dense enough Erdős-Rényi random graph is k-clique-free. The constant hidden in the $\Omega(1)$ increases with the density of the graph and, in particular, for very dense graphs and $k = n/3$ the length required is $2^{\Omega(n)}$. Also, for a specially tailored CNF encoding, where the ith member of the claimed k-clique is encoded in binary by $\log n$ variables, a lower bound of $n^{\Omega(k)}$ for $k \leq \log n$ can be extracted from a careful reading of [LPRT17]. However, in the more natural unary encodings, where indicator variables specify whether a vertex is in the clique, the size-width method cannot yield more than a $2^{\Omega(k^2/n)}$ lower bound since there are resolution proofs of width $O(k)$. This bound becomes trivial when $k \leq \sqrt{n}$.

In the restricted subsystem of *tree-like resolution*, optimal $n^{\Omega(k)}$ length lower bounds were established in [BGL13] for k-clique formulas on complete $(k-1)$-partite as well as on average for Erdős-Rényi random graphs of appropriate edge density. There is no hope to get hard instances for general resolution from complete $(k-1)$-partite graphs, however—in the same paper it was shown that all instances from the more general class of $(k-1)$-colourable graphs are easy for resolution. A closer study of these resolution proofs reveals that they are *regular*, meaning that if the proof is viewed as a directed acyclic graph (DAG), then no variable is eliminated more than once on any source-to-sink path.

More generally, regular resolution is an interesting and non-trivial model to analyse for the k-clique problem since it captures the reasoning used in many state-of-the-art algorithms used in practice (for a survey, see, e.g., [Pro12, McC17]). Nonetheless, it has remained consistent with state-of-the-art knowledge that for $k \leq n^{5/6}$ regular resolution might be able to certify k-clique-freeness in polynomial length independent of the value of k.

Our contributions We prove optimal $n^{\Omega(k)}$ average-case lower bounds for regular resolution proofs of unsatisfiability for k-clique formulas on Erdős-Rényi random graphs.

Theorem A.1.1 (Informal). *For any integer $k \ll \sqrt[4]{n}$, given an n-vertex graph G sampled at random from the Erdős-Rényi model with the appropriate edge density, regular resolution asymptotically almost surely requires length $n^{\Omega(k)}$ to certify that G does not contain a k-clique.*

At a high level, the proof is based on a bottleneck counting argument in the style of [Hak85] with a slight twist that was introduced in [RWY02]. In its classical form, such a proof takes four steps. First, one defines a distribution of random source-to-sink paths on the DAG representation of the proof. Second, a subset of the vertices of the DAG is identified—the set of *bottleneck nodes*—such that any random path must necessarily pass through at least one such node. Third, for any fixed bottleneck node, one shows that it is very unlikely that a random path passes through this particular node. Given this, a final union bound argument yields the conclusion that the DAG must have many bottleneck nodes, and so the resolution proof must be long.

The twist in our argument is that, instead of single bottleneck nodes, we need to define *bottleneck pairs* of nodes. We then argue that any random path passes through at least one such pair but that few random paths pass through any fixed pair; the latter part is based on Markov chain-type reasoning similar to [RWY02, Theorems 3.2, 3.5]. Furthermore, it crucially relies on that the graph satisfies a certain combinatorial property, which captures the idea that the common neighbourhood of a small set of vertices is well distributed across the graph. Identifying this combinatorial property is a key contribution of our work. In a separate argument (that, surprisingly, turned out to be much more elaborate than most arguments of this kind) we then establish that Erdős-Rényi random graphs of the appropriate edge density satisfy this property asymptotically almost surely. Combining these two facts yields our average-case lower bound.

Another contribution of this paper is a relatively simple observation that not only is regular resolution powerful enough to distinguish graphs that contain k-cliques from $(k-1)$-colourable graphs [BGL13], but it can also distinguish them from graphs that have a homomorphism to any fixed graph H with no k-cliques.

Paper outline The rest of this paper is organized as follows. Section A.2 presents some preliminaries. We show that some nontrivial k-clique instances are easy for regular resolution in Section A.3. Section A.4 contains the formal statement of the lower bounds we prove for Erdős-Rényi random graphs. In Section A.5 we define a combinatorial property of graphs and show that clique formulas on such graphs are hard for regular resolution, and the proof that Erdős-Rényi random graphs satisfy this property asymptotically almost surely is in Section A.6. Section A.7 explains why our results imply lower bounds on the running time of state-of-the-art algorithms for k-clique. We conclude in Section A.8 with a discussion of open problems.

A.2 Preliminaries

We write $G = (V, E)$ to denote a graph with vertices V and edges E, where G is always undirected, without loops and multiple edges. Given a vertex $v \in V$, we write $N(v)$ to denote the set of *neighbours of* v. For a set of vertices $R \subseteq V$ we write $\widehat{N}(R) = \bigcap_{v \in R} N(v)$ to denote the set of *common neighbours of* R. For two sets of vertices $R \subseteq V$ and $W \subseteq V$ we write $\widehat{N}_W(R) = \widehat{N}(R) \cap W$ to denote the set of *common neighbours of* R *inside* W. For a set $U \subseteq V$ we denote by $G[U]$ the subgraph of G induced by the set U. For $n \in \mathbb{N}^+$ we write $[n] = \{1, \ldots, n\}$. We say that $V_1 \mathbin{\dot{\cup}} V_2 \mathbin{\dot{\cup}} \cdots \mathbin{\dot{\cup}} V_k = V$ is a *balanced* k-partition of V if for all $i, j \in [k]$ it holds that $|V_i| \leq |V_j| + 1$. All logarithms are natural (base e) if not specified otherwise.

Probability and Erdős-Rényi random graphs We often denote random variables in boldface and write $\boldsymbol{X} \sim \mathscr{D}$ to denote that $\boldsymbol{X}$ is sampled from the distribution $\mathscr{D}$. A *p-biased coin*, or a *Bernoulli variable*, is the outcome of a coin flip that yields 1 with probability p and 0 with probability $1-p$. We use the special case of Markov's inequality saying that if $\boldsymbol{X}$ is non-negative, then $\Pr[\boldsymbol{X} \geq 1] \leq \mathbb{E}[\boldsymbol{X}]$. We also need the following special case of the multiplicative Chernoff bound: if $\boldsymbol{X}$ is a binomial random variable (i.e., the sum of i.i.d. Bernoulli variables) with expectation $\mu = \mathbb{E}[\boldsymbol{X}]$, then $\Pr[\boldsymbol{X} \leq \mu/2] \leq \mathrm{e}^{-\mu/8}$.

We consider the Erdős-Rényi distribution $\mathscr{G}(n,p)$ of random graphs on a fixed set V of n vertices. A random graph sampled from $\mathscr{G}(n,p)$ is produced by placing each potential edge $\{u,v\}$ independently with probability p, $0 \leq p \leq 1$ (the edge probability p may be a function of n). A property of graphs is said to hold *asymptotically almost surely* on $\mathscr{G}(n,p(n))$ if it holds with probability that approaches 1 as n approaches infinity.

For a positive integer k, let $\boldsymbol{X}_k$ be the random variable that counts the number of k-cliques in a random graph from $\mathscr{G}(n,p)$. It follows from Markov's inequality that asymptotically almost surely there are no k-cliques in $\mathscr{G}(n,p)$ whenever p and k are such that $\mathbb{E}[\boldsymbol{X}_k] = p^{\binom{k}{2}}\binom{n}{k}$ approaches 0 as n approaches infinity. This is the case, for example, if $p = n^{-2\eta/(k-1)}$ for $k \geq 2$ and $\eta > 1$.

CNF formulas and resolution A *literal* over a Boolean variable x is either the variable x itself (a *positive literal*) or its negation $\neg x$ (a *negative literal*). A *clause* $C = \ell_1 \vee \cdots \vee \ell_w$ is a disjunction of literals; we say that the *width* of C is w. The empty clause will be denoted by $\bot$. A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We think of clauses as sets of literals and of CNF formulas as sets of clauses, so that order is irrelevant and there are no repetitions. For a formula F we denote by $\mathit{Vars}(F)$ the set of variables of F.

A *resolution derivation* from a CNF formula F is as an ordered sequence of clauses $\pi = (D_1, \ldots, D_L)$ such that for each $i \in [L]$ either D_i is a clause in C or there exist $j < i$ and $k < i$ such that D_i is derived from D_j and D_k by the *resolution rule*

$$\frac{B \vee x \qquad C \vee \neg x}{B \vee C}\,, \tag{A.1}$$

$D_i = B \vee C$, $D_j = B \vee x$, $D_k = C \vee \neg x$. We refer to $B \vee C$ as the *resolvent* of $B \vee x$ and $C \vee \neg x$ over x, and to x as the *resolved variable*. The *length* (or *size*) of a resolution derivation $\pi = (D_1, \ldots, D_L)$ is L and it is denoted by $|\pi|$. A *resolution refutation* of F, or *resolution proof* for (the unsatisfiability of) F, is a resolution derivation from F that ends in the empty clause $\bot$.

A resolution derivation $\pi = (D_1, \ldots, D_L)$ can also be viewed as a labelled DAG with the set of nodes $\{1, \ldots, L\}$ and edges (j,i), (k,i) for each application of the resolution rule deriving D_i from D_j and D_k. Each node i in this DAG is labelled by its associated clause D_i, and each non-source node is also labelled by the resolved variable in its associated derivation step in the refutation. A resolution refutation is called *regular* if along any source-to-sink path in its associated DAG every variable is resolved at most once.

For a partial assignment ρ we say that a clause C *restricted by* ρ, denoted $C\restriction_\rho$, is the trivial 1-clause if any of the literals in C is satisfied by ρ or otherwise is C with all falsified literals removed. We extend this definition to CNFs in the obvious way: $(C_1 \wedge \ldots \wedge C_m)\restriction_\rho = C_1\restriction_\rho \wedge \ldots \wedge C_m\restriction_\rho$. Applying a restriction preserves (regular) resolution derivations. To see this, observe that in every application of the resolution rule the restricted consequence is either killed (becomes identically 1) or obtained, as before, by resolving the two restricted premises or it is a copy of one of them. Thus, we have:

Fact A.2.1. *Let π be a (regular) resolution refutation of a CNF formula F. For any partial assignment ρ to the variables of F there is an efficiently constructible (regular) resolution refutation $\pi\restriction_\rho$ of the CNF formula $F\restriction_\rho$, so that the length of $\pi\restriction_\rho$ is at most the length of π.*

Branching programs A branching program on variables $x_1, \ldots, x_n$ is a DAG that has one source node and where every non-sink node is labelled by one of the variables $x_1, \ldots, x_n$ and has exactly two outgoing edges labelled 0 and 1. The size of a branching program is the total number of nodes in the graph. In a *read-once branching program* it holds in addition that along every path every variable appears as a node label at most once.

For each node a in a branching program, let $X(a)$ denote the variable that labels a, and let a^0 and a^1 be the nodes that are reached from a through the edges labelled 0 and 1, respectively. A truth-value assignment $\sigma : \{x_1, \ldots, x_n\} \to \{0,1\}$ determines a path in a branching program in the following way. The path starts at the source node. At an internal node a, the path is extended along the edge labelled $\sigma(X(a))$ so that the next node in the path is $a^{\sigma(X(a))}$. The path ends when it reaches a sink. We write path(σ) for the path determined by σ. When extending the path from a node a to the node $a^{\sigma(X(a))}$, we say that the *answer to the query* $X(a)$ at a is $\sigma(X(a))$ and that the path *sets* the variable $X(a)$ to the value $\sigma(X(a))$. For each node a of the branching program, let $\beta(a)$ be the maximal partial assignment that is contained in any assignment σ such that path(σ) passes through a. Equivalently, this is the set of all those $\sigma(x_i) = \gamma$ for which the query x_i is made, and answered by γ, along every consistent path from the source to a. If the program is read-once, the consistency condition becomes redundant.

The *falsified clause search problem* for an unsatisfiable CNF formula F is the task of finding a clause $C \in F$ that is falsified by a given truth value assignment σ. A branching program P on the variables *Vars*(F) *solves* the falsified clause search problem for F if each sink is labelled by a clause of F such that for every assignment σ, the clause that labels the sink reached by path(σ) is falsified by σ. The minimal size of any regular resolution refutation of an unsatisfiable CNF formula F is exactly the same as the minimal size of any read-once branching program solving the falsified clause search problem for F. This can be seen by taking the refutation DAG and reversing the edges to get a branching program or vice versa. For a formal proof see, e.g., [Kra95a, Theorem 4.3].

The k-clique formula In order to analyse the complexity of resolution proofs that establish that a given graph does not contain a k-clique we must formulate the problem as a propositional formula in conjunctive normal form (CNF). We consider two distinct encodings for the clique problem originally defined in [BIS07].

The first propositional encoding we present, $Clique(G,k)$, is based on mapping of vertices to clique members. This formula is defined over variables $x_{v,i}$ ($v \in V, i \in [k]$) and consists of the following set of clauses:

$$\neg x_{u,i} \vee \neg x_{v,j} \qquad i,j \in [k], i \neq j, u,v \in V, \{u,v\} \notin E \ , \tag{A.2a}$$

$$\bigvee_{v \in V} x_{v,i} \qquad i \in [k] \ , \tag{A.2b}$$

$$\neg x_{u,i} \vee \neg x_{v,i} \qquad i \in [k], u,v \in V, u \neq v \ , \tag{A.2c}$$

We refer to (A.2a) as *edge axioms*, (A.2b) as *clique axioms* and (A.2c) as *functionality axioms*. Note that $Clique(G,k)$ is satisfiable if and only if G contains a k-clique, and that this is true even if clauses (A.2c) are omitted—we write $Clique^*(G,k)$ to denote this formula with only clauses (A.2a) and (A.2b).

The second version of clique formulas that we consider is the block encoding, which we denote by $Clique_{\text{block}}(G,k)$. This formula differs from the previous one in that it requires a k-clique that has a certain “block-respecting” structure. Let $V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k = V$ be a balanced k-partition of V. This formula, defined over variables x_v, encodes the fact that the graph contains a *transversal k-clique*, that is, a k-clique in which each clique member belongs to a different block. Formally, for any positive k and n, the formula $Clique_{\text{block}}(G,k)$ consists of the following set of clauses:

$$\neg x_u \vee \neg x_v \qquad u,v \in V, u \neq v, \{u,v\} \notin E \ , \tag{A.3a}$$

$$\bigvee_{v \in V_i} x_v \qquad i \in [k] \ , \tag{A.3b}$$

$$\neg x_u \vee \neg x_v \qquad i \in [k], u,v \in V_i, u \neq v \ . \tag{A.3c}$$

We refer to (A.3a) as *edge axioms*, (A.3b) as *clique axioms*, and (A.3c) as *functionality axioms*.

Note that a graph can contain a k-clique but contain no transversal k-clique for a given partition. Intuitively it is clear that proving that a graph does not contain a transversal k-clique should be easier than proving it does not contain any k-clique, since any proof of the latter fact must in particular establish the former. We make this intuition formal below.

Lemma A.2.2 ([BIS07]). *For any graph G and any $k \in \mathbb{N}^+$, the size of a minimum regular resolution refutation of $Clique(G,k)$ is bounded from below by the size of a minimum regular resolution refutation of $Clique_{\text{block}}(G,k)$.*

This lemma was proven in [BIS07] for tree-like and for general resolution via a restriction argument, and it is straightforward to see that the same proof holds for regular resolution.

A.3 Graphs That Are Easy for Regular Resolution

Before proving our main $n^{\Omega(k)}$ lower bound, in this section we exhibit classes of graphs whose clique formulas have regular resolution refutations of fixed-parameter tractable length, i.e., length $f(k) \cdot n^{O(1)}$ for some function f. This illustrates the strength of regular resolution for the k-clique problem. We note that the upper bounds claimed in this section hold not only for $Clique(G,k)$ but even for the subformula $Clique^{*}(G,k)$ that omits the functionality axioms (A.2c).

The first example is the class of $(k-1)$-colourable graphs. Such graphs are hard for tree-like resolution [BGL13], and the known algorithms that distinguish them from graphs that contain k-cliques are highly non-trivial [Lov79, Knu94]. The second example is the class of graphs that have a homomorphism into a fixed k-clique free graph.

Recall that a homomorphism from a graph $G=(V,E)$ into a graph $G'=(V',E')$ is a mapping $h: V \to V'$ that maps edges $\{u,v\} \in E$ into edges $\{h(u),h(v)\} \in E'$. A graph is $(k-1)$-colourable if and only if it has a homomorphism into the $(k-1)$-clique, which is of course k-clique free. Therefore our second example is a generalization of the first one (but the function $f(k)$ becomes larger).

Both upper bounds follows from a generic procedure, based on Algorithm 1, that builds read-once branching programs for the falsified clause search problem for the formula $Clique^{*}(G,k)$.

Given a k-clique free graph G define

$$I(G) = \left\{ G\left[\widehat{N}(R)\right] \;:\; R \text{ is a clique in } G \right\} \; . \tag{A.4}$$

Proposition A.3.1. *There is an efficiently constructible read-once branching program for the falsified clause search problem on formula $Clique^{*}(G,k)$ of size at most $|I(G)| \cdot k^2 \cdot |V(G)|^2$.*

Proof. We build the branching program recursively, following the strategy laid out by Algorithm 1. For the base case $k=1$, G must be the graph with no vertices. The branching program is a single sink node that outputs the clique axiom of index 1, i.e., the empty clause.

For $k>1$, fix $n=|V(G)|$ and an ordering $v_1,\ldots,v_n$ of the vertices in $V(G)$. We first build a decision tree T by querying the variables $x_{v_1,k}, x_{v_2,k}, \ldots$ in order, until we get an answer 1, or until all variables with second index k have been queried. If $x_{v_j,k}=0$ for all $j \in [n]$ then the kth clique axiom (A.2b) is falsified by the assignment (see line 9). Otherwise, let v be the first vertex in the order where $x_{v,k}=1$. The decision tree now queries $x_{w,i}$ for all $w \notin N(v)$ and all $i<k$ to check whether an edge axiom involving v is falsified (lines 4–5). If any of these variables is set to 1 the branching stops and the leaf node is labelled with the corresponding edge axiom $\neg x_{v,k} \vee \neg x_{w,i}$.

The decision tree T built so far has at most kn^2 nodes, and we can identify n "open" leaf nodes $a_{v_1}, a_{v_2}, \ldots, a_{v_n}$, where a_{v_i} is the leaf node reached by the path that sets $x_{v_i,k}=1$ and that does yet determine the answer to the search problem. Let us focus on a

Algorithm 1 Read-once branching program for the falsified clause search problem on $Clique^*(G,k)$.

```
Input  : $k \in \mathbb{N}^+$, a $k$-clique free graph $G$, an assignment
         $\alpha : \{x_{v,i}$ for $v \in V(G), i \in [k]\} \to \{0,1\}$
Output : A clause of $Clique^*(G,k)$ falsified by $\alpha$
Search($G, k, \alpha$): begin
    for $v \in V(G)$ do
        if $\alpha(x_{v,k}) = 1$ then
            for $w \notin N(v)$ and $i < k$ do
                if $\alpha(x_{w,i}) = 1$ then return edge axiom $\neg x_{v,k} \vee \neg x_{w,i}$ (A.2a)
            $G' \leftarrow G[N(v)]$
            $\alpha' \leftarrow \alpha$ restricted to variables $x_{w,j}$ for $w \in V(G')$ and $1 \le j \le k-1$
            return Search($G', k-1, \alpha'$)
    return the kth clique axiom (A.2b)
```

specific node a_v for some $v \in V(G)$. The partial assignment path(a_v) sets v to be the kth member of the clique and no vertex in $V(G) \setminus N(v)$ to be in the clique. Let G_v be the subgraph induced on G by $N(v)$, let S_v be the set of variables $x_{w,i}$ for $w \in N(v)$ and $i < k$, and let ρ_v be the partial assignment setting $x_{w,i} = 0$ for $w \notin N(v)$ and $i < k$. Clearly $\rho_v \subseteq$ path(a_v).

By the inductive hypothesis there exists a branching program B_v that solves the search problem on $Clique^*(G_v, k-1)$ querying only variables in S_v. This corresponds to the recursive call for the subgraph G_v and $k-1$ (lines 6–8). If we attach each B_v to a_v we get a complete branching program for $Clique^*(G,k)$. This is read-once because B_v only queries variables in S_v and these variables are not in path(a_v).

To prove that the composed program is correct we consider an assignment σ to the variables in S_v and show that the clause output by B_v on σ is also a valid output for the search problem on $Clique^*(G,k)$, i.e., it is falsified by the assignment path(a_v) $\cup\, \sigma$. Actually we show the stronger claim that it is falsified by $\rho_v \cup \sigma$, which is a subset of path(a_v) $\cup\, \sigma$. To this end, note that if the output of B_v on σ is an edge axiom of $Clique^*(G_v, k-1)$, this must be some $\neg x_{u,i} \vee \neg x_{w,j}$ for $i, j < k$, which is also an edge axiom of $Clique^*(G,k)$ and is falsified by $\sigma \subseteq \rho_v \cup \sigma$. Now if the output of B_v on σ is the ith clique axiom of $Clique^*(G_v, k-1)$, then σ falsifies $\bigvee_{w \in N(v)} x_{v,i}$, and therefore $\rho_v \cup \sigma$ falsifies the ith clique axiom in formula $Clique^*(G,k)$.

The construction so far is correct but produces a very large branching program (in particular, a tree-like one). In order to create a smaller branching program, we observe that if $u, v \in V(G)$ are such that $N(u) = N(w)$ then $G_u = G_w$, $B_u = B_w$ and $\rho_u = \rho_w$. In this case, we can identify nodes a_u and a_w, resulting in a node we denote a^*, and identify the branching programs B_u and B_w. The correctness of this new program is due to the fact that even after the identification of vertices $\rho_u \subseteq \text{path}(a^*)$ and $\rho_w \subseteq \text{path}(a^*)$. This process leads to having only one subprogram for each distinct induced subgraph at each level of the recursion.

In order to bound the size of this program, we decompose it into k levels. The source is at level zero and corresponds to the graph G. At level i there are nodes corresponding to all subgraphs induced by the common neighbourhood of cliques of size i. Each node in the ith level connects to the nodes of the $(i+1)$th level by a branching program of size at most kn^2. Notice that an induced subgraph in $I(G)$ cannot occur twice in the same layers, so the total size of the final branching program is at most $|I(G)| \cdot k^2 n^2$ nodes. □

We now proceed to prove the upper bounds mentioned previously. A graph G that has a homomorphism into a small k-clique free graph H may still have a large set $I(G)$, making Proposition A.3.1 inefficient. The first key observation is that if G has a homomorphism into a graph H then it is a subgraph of a blown up version of H, namely, of a graph obtained by transforming each vertex of H into a "cloud" of vertices where a cloud does not contain any edge, two clouds corresponding to two adjacent vertices in H have all possible edges between them, and two clouds corresponding to two non-adjacent vertices in H have no edges between them. A second crucial point is that if G' is a blown up version of H then it turns out that $|I(G')| = |I(H)|$, making Proposition A.3.1 effective for G'. The upper bound then follows from observing that the task of proving that G is k-clique free should not be harder than the same task for a supergraph of G. Indeed Fact A.3.2 formalises this intuition. It is interesting to observe that the constructions in Proposition A.3.1 and in Fact A.3.2 are efficient. The non-constructive part is guessing the homomorphism to H.

Fact A.3.2. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs with no k-clique such that $V \subseteq V'$ and $E \subseteq E' \cap \binom{V}{2}$. If $Clique^*(G', k)$ has a (regular) refutation of length L, then $Clique^*(G, k)$ also has a (regular) refutation of length L.*

Proof. Consider the partial assignment ρ that sets $x_{v,i} = 0$ for every $v \notin V$ and $i \in [k]$. The restricted formula $Clique^*(G', k)\restriction_\rho$ is isomorphic to $Clique^*(\widetilde{G}, k)$, where $V(\widetilde{G}) = V$ and $E(\widetilde{G}) = E' \cap \binom{V}{2}$, and thus, by Fact A.2.1, has a (regular) refutation π of length at most L. Removing edges from a graph only introduces additional edge axioms (A.2a) in the corresponding formula, therefore $Clique^*(\widetilde{G}, k) \subseteq Clique^*(G, k)$ and π is a valid refutation of $Clique^*(G, k)$ as well. □

It was shown in [BGL13] that the k-clique formula of a complete $(k-1)$-partite graph on n vertices has a regular resolution refutation of length $2^k n^{O(1)}$, although the regularity is not stressed in that paper. Since it is instructive to see how this refutation is constructed in this framework, we give a self-contained proof.

Proposition A.3.3 ([BGL13, Proposition 5.3]). *If G is a $(k-1)$-colourable graph on n vertices, then $Clique^*(G, k)$ has a regular resolution refutation of length at most $2^k k^2 n^2$.*

Proof. Let $V = V(G)$ and let $V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_{(k-1)}$ be a partition of V into colour classes. Define the graph $G' = (V, E')$ where the edge set E' has an edge between any pair of vertices belonging to two different colour classes. Clearly G is a subgraph of G'. Observe that any clique R in G' has at most one vertex in each colour class, and that the common neighbours of R are all the vertices in the colour classes not touched by R.

Therefore, there is a one-to-one correspondence between the members of $I(G')$ and the subsets of $[k-1]$. By Proposition A.3.1 there is a read-once branching program for the falsified clause search problem on formula $Clique^*(G', k)$ of size at most $2^k k^2 n^2$. This read-once branching program corresponds to a regular resolution refutation of $Clique^*(G', k)$ of the same size. By Fact A.3.2 there must be a regular resolution refutation of size at most $2^k k^2 n^2$ for $Clique^*(G, k)$ as well. □

Next we generalize Proposition A.3.3 to graphs G that have a homomorphism to a k-clique free graph H.

Proposition A.3.4. *If G is a graph on n vertices that has a homomorphism into a k-clique free graph H on m vertices, then $Clique^*(G, k)$ has a regular resolution refutation of length at most $m^k k^2 n^2$.*

Proof. Fix a homomorphism $h : V(G) \to V(H)$ and an ordering $u_1, \ldots, u_m$ of the vertices of H. Let $V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_m$ be the partition of $V(G)$ such that V_i is the set of vertices of G mapped to u_i by h. We define the graph $G' = (V, E')$ where

$$E' = \bigcup_{\{u_i,u_j\} \in E(H)} V_i \times V_j \ , \tag{A.5}$$

that is, G' is a blown up version of H that contains G as a subgraph. To prove our result we note that, by Proposition A.3.1, there is a read-once branching program for the falsified clause search problem on $Clique^*(G', k)$—and hence also a regular resolution refutations of the same formula—of size at most $|I(G')| \cdot k^2 n^2$. This implies that, by Fact A.3.2, there is a regular resolution refutation of $Clique^*(G, k)$ of at most the same size.

To conclude the proof it remains only to show that $|I(G')| \leq m^k$. By construction, h maps injectively a clique $R \subseteq V(G')$ into a clique $R_H \subseteq V(H)$ of the same size. Moreover, note that if $U = \widehat{N}(R_H)$, then $\widehat{N}(R) = \cup_{u_i \in U} V_i$. Therfore, for any clique $R' \subseteq V(G')$ that is mapped by h to R_H it holds that $\widehat{N}(R) = \widehat{N}(R')$, i.e., $\widehat{N}(R')$ is completely characterized by the clique in H it is mapped to. Thus $I(G)$ has at most one element for each clique in H and we have that $|I(G')| = |I(H)|$. Finally, note that $|I(H)| \leq m^k$ since, being k-clique free, H cannot have more than m^k cliques. □

A.4 Random Graphs Are Hard for Regular Resolution

The main result of this paper is an average case lower bound of $n^{\Omega(k)}$ for regular resolution for the k-clique problem. As we saw in Section A.2, the k-clique problem can be encoded in different ways and depending on the preferred formula the range of k for which we can obtain a lower bound differs. In this section we present a summary of our results for the different encodings.

Theorem A.4.1. *For any real constant $\epsilon > 0$, any sufficiently large integer n, any positive integer $k \le n^{1/4-\epsilon}$, and any real $\xi > 1$, if $\boldsymbol{G} \sim \mathscr{G}(n, n^{-2\xi/(k-1)})$ is an Erdős-Rényi random graph, then, with probability at least $1 - \exp(-\sqrt{n})$, any regular resolution refutation of $Clique_{\text{block}}(\boldsymbol{G}, k)$ has length at least $n^{\Omega(k/\xi^2)}$.*

The parameter ξ determines the density of the graph: the larger ξ the sparser the graph and the problem of determining whether $\boldsymbol{G}$ contains a k-clique becomes easier. For constant ξ, where the edge probability is somewhat close to the threshold for containing a k-clique, the theorem yields a $n^{\Omega(k)}$ lower bound which is tight up to the multiplicative constant in the exponent. The lower bound decreases smoothly with the edge density and is non-trivial for $\xi = o(\sqrt{k})$.

A problem which is closely related to the problem we consider is that of distinguishing a random graph sampled from $\mathscr{G}(n, p)$ from a random graph from the same distribution with a planted k-clique. The most studied setting is when $p = 1/2$. In this scenario the problem can be solved in polynomial time with high probability for $k \approx \sqrt{n}$ [Kuč95, AKS98]. It is still an open problem whether there exists a polynomial time algorithm solving this problem for $\log n \ll k \ll \sqrt{n}$. For $\boldsymbol{G} \sim \mathscr{G}(n, 1/2)$, Theorem A.4.1 implies that to refute $Clique_{\text{block}}(\boldsymbol{G}, k)$ asymptotically almost surely regular resolution requires $n^{\Omega(\log n)}$ size for $k = O(\log n)$ and super-polynomial size for $k = \mathrm{o}(\log^2 n)$.

An interesting question is whether Theorem A.4.1 holds for larger values of k. We show that for the formula $Clique(G, k)$ (recall that by Lemma A.2.2 this encoding is easier for the purpose of lower bounds) we can prove the lower bound for $k \le n^{1/2-\epsilon}$ as long as the edge density of the graph is close to the threshold for containing a k-clique.

Theorem A.4.2. *For any real constant $\epsilon > 0$, any sufficiently large integer n, any positive integer k, and any real $\xi > 1$ such that $k\sqrt{\xi} \le n^{1/2-\epsilon}$, if $\boldsymbol{G} \sim \mathscr{G}(n, n^{-2\xi/(k-1)})$ is an Erdős-Rényi random graph, then, with probability at least $1 - \exp(-\sqrt{n})$, any regular resolution refutation of $Clique(\boldsymbol{G}, k)$ has length at least $n^{\Omega(k/\xi^2)}$.*

In this paper we prove Theorem A.4.1 and we refer to the conference version of this paper [ABdR+18] for the proof of Theorem A.4.2. We note, however, that both proofs are very similar and having seen one it is an easy exercise to obtain the other. The proof of Theorem A.4.1 is deferred to Section A.6 and is based on a general lower bound technique we develop in Section A.5.

A.5 Clique-Denseness Implies Hardness for Regular Resolution

In this section we define a combinatorial property of graphs, which we call *clique-denseness*, and prove that if a k-clique-free graph G is clique-dense with the appropriate parameters, then this implies a lower bound $n^{\Omega(k)}$ on the length of any regular resolution refutation of the k-clique formula on G.

In order to argue that regular resolution has a hard time certifying the k-clique-freeness of a graph G, one property that seems useful to have is that for every small enough clique in the graph there are many ways of extending it to a larger clique. In other words, if $R \subseteq V$ forms a clique and R is small, we would like the common neighbourhood $\widehat{N}_V(R)$ to be large. This motivates the following definitions.

Definition A.5.1 (Neighbour-dense set). Given $G = (V, E)$ and $q, r \in \mathbb{R}^+$, a set $W \subseteq V$ is *q-neighbour-dense for $R \subseteq V$* if $\left|\widehat{N}_W(R)\right| \geq q$. We say that W is *(r, q)-neighbour-dense* if it is q-neighbour-dense for every $R \subseteq V$ of size $|R| \leq r$.

If W is an (r, q)-neighbour-dense set, then we know that any clique of size r can be extended to a clique of size $r + 1$ in at least q different ways by adding some vertex of W. Note, however, that the definition of (r, q)-neighbour-dense is more general than this since R is not required to be a clique.

We next define a more robust notion of neighbour-denseness. For some settings of r and q of interest to us it is too much to hope for a set W which is q-neighbour-dense for every $R \subseteq V$ of size at most r. In this case we would still like to be able to find a "mostly neighbour-dense" set W in the sense that we can "localize" bad (i.e., those for which W fails to be q-neighbour-dense) sets $R \subseteq V$ of size $|R| \leq r$.

Definition A.5.2 (Mostly neighbour-dense set). Given $G = (V, E)$ and $r', r, q, s \in \mathbb{R}^+$ with $r' \geq r$, a set $W \subseteq V$ is *(r', r, q, s)-mostly neighbour-dense* if there exists a set $S \subseteq V$ of size $|S| \leq s$ such that for every $R \subseteq V$ with $|R| \leq r'$ for which W is not q-neighbour-dense, it holds that $|R \cap S| \geq r$.

In what follows, it might be helpful for the reader to think of r' and r as linear in k and q and s as polynomial in n, where we also have $s \ll q$.

Now we are ready to define a property of graphs that makes it hard for regular resolution to certify that graphs with this property are indeed k-clique-free.

Definition A.5.3 (Clique-dense graph). Given $k \in \mathbb{N}^+$ and $t, s, \varepsilon \in \mathbb{R}^+$, $1 \leq t \leq k$ we say that a graph $G = (V, E)$ with a k-partition $V_1 \cup \cdots \cup V_k = V$ is *(k, t, s, ε)-clique-dense* if there exist $r, q \in \mathbb{R}^+$, $r \geq 4k/t^2$, such that

1. V_i is (tr, tq)-neighbour-dense for all $i \in [k]$, and
2. every (r, q)-neighbour-dense set $W \subseteq V$ is (tr, r, q', s)-mostly neighbour-dense for $q' = \varepsilon r s^{1+\varepsilon} \log s$.

Remark A.5.4 (The complete $(k-1)$-partite graph is not clique-dense). Since the property of clique-denseness in Definition A.5.3 is a sufficient condition for the lower bound, it is worth to pause and observe that this property does not hold for examples such as the $(k-1)$-colourable graphs which have non-trivially short proofs.

Consider indeed $G = (V, E)$ to be the $(k-1)$-colourable graph with balanced color classes and maximum set of edges. Namely $V = \bigcup_c U_c$ with $c \in [k-1]$ and $|U_c| = n/(k-1)$. The edges of the G are all pairs $\{u, v\}$ for $u \in U_c$ and $v \in U_{c'}$ with $c \neq c'$. Graph G satisfies Property 1 for essentially any reasonable k-partition of V, but it fails property (2) it a pretty extreme way.

Given $r < k-1$ we pick W to be the union of $r+1$ arbitrarily chosen colour classes. The chosen set W is (r, q)-neighbour-dense for any q up to $n/(k-1)$, because the common neighbourhood of any of r vertices in W must containt one of the colour classes U_c.

Can it W be (tr, r, q', s)-mostly neighbour-dense for some choice of the parameters? By definition $t > 1$, and we see that already if we choose R of size $r+1$ by picking one vertex for each colour class in W, the common neighbourhood is empty. To make W to be (tr, r, q', s)-mostly neighbour-dense for some choice of $s \ll q' < n$ we should find a set S of size s that has large intersection with any such R. But since S is much smaller than n (and therefore smaller than of $n/(k-1)$), it cannot cover completely any of the colour classes in W. Hence there are some R for which any chosen S has even empty intersection and yet R has no common neighbours in W.

Theorem A.5.5. *Given $k \in \mathbb{N}^+$ and $t, s, \varepsilon \in \mathbb{R}^+$ if the graph $G = (V, E)$ with balanced k-partition $V_1 \cup \cdots \cup V_k = V$ is (k, t, s, ε)-clique-dense, then every regular resolution refutation of the CNF formula $Clique_{\mathrm{block}}(G, k)$ has length at least $\Omega\big(s^{\varepsilon k/t^2}\big)$.*

The value of q' in Definition A.5.3 can be tailored in order to prove Theorem A.4.1 for slightly larger values of k. For example, setting $q' = 3\varepsilon s^{1+\varepsilon} \log s$ and making the necessary modifications in the proof would yield Theorem A.4.1 for $k \ll n^{1/3}$ but for a smaller range of edge densities. A similar adjustment was done in the conference version of this paper [ABdR$^+$18] to obtain Theorem A.4.2 for $k \ll n^{1/2}$.

We will spend the rest of this section establishing Theorem A.5.5. Fix $r, q \in \mathbb{R}^+$ witnessing that G is (k, t, s, ε)-clique-dense as per Definition A.5.3. We first note that we can assume that $tr \leq k$ since otherwise, by property 1 of Definition A.5.3, G contains a block-respecting k-clique and the theorem follows immediately.

By the discussion in Section A.2 it is sufficient to consider read-once branching programs, since they are equivalent to regular resolution refutations, and so in what follows this is the language in which we will phrase our lower bound. Thus, for the rest of this section let P be an arbitrary, fixed read-once branching program that solves the falsified clause search problem for $Clique_{\mathrm{block}}(G, k)$. We will use the convention of referring to "vertices" of the graph G and "nodes" of the branching program P to distinguish between the two. We sometime abuse notation and say that a vertex $v \in V$ is set to 0 or to 1 when we mean that the corresponding variable x_v is set to 0 or to 1.

Recall that for a node a of P, $\beta(a)$ denotes the maximal partial assignment that is contained in any assignment σ such that the path path(σ) passes through a. For any partial assignment β we write β^1 to denote the partial assignment that contains exactly the variables that are set to 1 in β. Clearly, if β falsifies an edge axiom or a functionality axiom, then so does β^1. Furthermore, for any $\gamma \supseteq \beta$, if β falsifies an axiom so does γ. We will use this monotonicity property of partial assignments throughout the proof.

For each node a of P and each index $i \in [k]$ we define two sets of vertices

$$V_i^0(a) = \{u \in V_i \mid \beta(a) \text{ sets } x_u \text{ to } 0\} \tag{A.6a}$$

$$V_i^1(a) = \{u \in V_i \mid \beta(a) \text{ sets } x_u \text{ to } 1\} \tag{A.6b}$$

of G. Observe that for $\beta = \beta(a)$ the set of vertices referenced by variables in β^1 is $\bigcup_i V_i^1(a)$.

Intuitively, one can think of $V_i^0(a)$ and $V_i^1(a)$ as the only sets of vertices in V_i assigned 0 and 1, respectively, that are "remembered" at the node a (in the language of resolution, they correspond to negative and positive occurrences of variables in the clause D_a associated with the node a). Other assignments to vertices in V_i encountered along some path to a have been "forgotten" and may not be queried any more on any path starting at a. Formally, we say that a vertex v is *forgotten at a* if there is a path from the source of P to a passing through a node b where v is queried, but v is not in $V_i^0(a)$ nor in $V_i^1(a)$. Furthermore, we say index i *is forgotten at a* if some vertex $v \in V_i$ is forgotten at a. Of utter importance is the fact that these notions are persistent: if a variable or an index is forgotten at a node a, then it will also be the case for any node reachable from a by a path. We say that a path in P *ends in the ith clique axiom* if the clause that labels its last node is the clique axiom (A.3b) of $Clique_{\text{block}}(G,k)$ with index i. The above observation implies that the index i cannot be forgotten at any node along such a path.

We establish our lower bound via a bottleneck counting argument for paths in P. To this end, let us define a distribution $\mathscr{D}$ over paths in P by the following random process. The path starts at the source and ends whenever it reaches a sink of P. At an internal node a with successor nodes a^0 and a^1, reached by edges labelled 0 and 1 respectively, the process proceeds as follows.

1. If $X(a) = x_u$ for $u \in V_i$ and i is forgotten at a then the path proceeds via the edge labelled 0 to a^0.
2. If $X(a) = x_u$ and $\beta(a) \cup \{x_u = 1\}$ falsifies an edge axiom (A.3a) or a functionality axiom (A.3c), then the path proceeds to a^0.
3. Otherwise, an independent $s^{-(1+\varepsilon)}$-biased coin is tossed with outcome $\gamma \in \{0,1\}$ and the random path proceeds to a^γ.

We say that in cases 1 and 2 the answer to the query $X(a)$ is *forced*. Note that any path α in the support of $\mathscr{D}$ must end in a clique axiom since α does not falsify any edge or functionality axiom by item 2 in the construction. Moreover, a property that will be absolutely crucial is that only answers 0 can be forced—answers 1 are always the result of a coin flip.

Claim A.5.6. *Every path in the support of $\mathscr{D}$ sets at most k variables to* 1.

Proof. Let α be a path in the support of $\mathcal{D}$. We argue that for each $i \in [k]$ at most one vertex $u \in V_i$ is such that the variable x_u is set to 1 on α. Let a and b be two nodes that appear in this order in α. If for some $i \in [k]$, and for some $u, v \in V_i$, x_u is set to 1 by α at node a and x_v is queried at b, then $v \neq u$ by regularity and, by definition of $\mathcal{D}$, the answer to query x_v will be forced to 0, either to avoid violating a functionality or an edge axiom, or because i is forgotten at b. $\square$

Let us call a pair (a, b) of nodes of P *useful* if there exists an index i such that $V_i^1(b) = \emptyset$, i is not forgotten at b (which in particular implies $V_i^1(a) = \emptyset$ and $V_i^0(a) \subseteq V_i^0(b)$), and the set $V_i^0(b) \setminus V_i^0(a)$ is (r, q)-neighbour-dense. For each useful pair (a, b), let $i(a, b)$ be an arbitrary but fixed index witnessing that (a, b) is useful. A path is said to *usefully* traverse a useful pair (a, b) if it goes through a and b in that order and sets at most $\lceil k/t \rceil$ variables to 1 between a and b (with a included and b excluded).

As already mentioned, the proof of Theorem A.5.5 is based on a bottleneck counting argument in the spirit of [Hak85], with the twist that we consider pairs of bottleneck nodes. To establish the theorem we make use of the following two lemmas which will be proven subsequently.

Lemma A.5.7. *Every path in the support of $\mathcal{D}$ usefully traverses a useful pair.*

Lemma A.5.8. *For every useful pair (a, b), the probability that a random $\boldsymbol{\alpha}$ chosen from $\mathcal{D}$ usefully traverses (a, b) is at most $2s^{-\varepsilon r/2}$.*

Combining the above lemmas, it is immediate to prove Theorem A.5.5. By Lemma A.5.7 the probability that a random path $\boldsymbol{\alpha}$ sampled from $\mathcal{D}$ usefully traverses some useful pair is 1. By Lemma A.5.8, for any fixed useful pair (a, b), the probability that a random $\boldsymbol{\alpha}$ usefully traverses (a, b) is at most $2s^{-\varepsilon r/2}$. By a standard union bound argument, it follows that the number of useful pairs is at least $\frac{1}{2}s^{\varepsilon r/2}$, so the number of nodes in P cannot be smaller than $\Omega\big(s^{\varepsilon r/4}\big) \geq \Omega\big(s^{\varepsilon k/t^2}\big)$ (recall that $r \geq 4k/t^2$ according to Definition A.5.3).

To conclude the proof it remains only to establish Lemmas A.5.7 and A.5.8.

Proof of Lemma A.5.7. Consider any path in the support of $\mathcal{D}$. As we already remarked, this path ends in the i^*th clique axiom for some $i^* \in [k]$ which in particular implies that $V_{i^*}^1(b) = \emptyset$ and that i^* is not forgotten at any b along this path. By Claim A.5.6, the path sets at most k variables to 1 and hence we can split it into t pieces by nodes $a_0, a_1, \ldots, a_t$ (a_0 is the source, a_t the sink) so that between a_j and a_{j+1} at most $\lceil k/t \rceil$ variables are set to 1. It remains to prove that for at least one $j \in [t]$ the set

$$W_j = V_{i^*}^0(a_j) \setminus V_{i^*}^0(a_{j-1}) \tag{A.7}$$

is (r, q)-neighbour-dense. Note that this will prove Lemma A.5.7 since by construction (a_{j-1}, a_j) is then a pair that is usefully traversed by the path.

Towards contradiction, assume instead that no W_j is (r,q)-neighbour-dense, i.e., that for all $j \in [t]$ there exists a set of vertices $R_j \subseteq V$ with $|R_j| \le r$ such that $\left|\widehat{N}_{W_j}(R_j)\right| \le q$. Let $R = \bigcup_{j\in[t]} R_j$. Since the path ends in the i^*th clique axiom we have $V^0_{i^*}(a_t) = V_{i^*}$. It follows that the sets $W_1,\dots,W_t$ in (A.7) form a partition of V_{i^*}, and therefore

$$\left|\widehat{N}_{V_{i^*}}(R)\right| = \sum_{j\in[t]} \left|\widehat{N}_{W_j}(R)\right| \le \sum_{j\in[t]} \left|\widehat{N}_{W_j}(R_j)\right| \le tq \ . \tag{A.8}$$

Since $|R| \le \sum_{j\in[t]} |R_j| \le tr$ this contradicts the assumption that V_{i^*} is (tr,tq)-neighbour-dense. Lemma A.5.7 follows. □

Proof of Lemma A.5.8. Fix a useful pair (a,b). Let E denote the event that a random path sampled from $\mathscr{D}$ usefully traverses (a,b). Let $i^* = i(a,b)$, $V^1(a) = \bigcup_{j\in[k]} V^1_j(a)$, and $W = V^0_{i^*}(b) \setminus V^0_{i^*}(a)$. Notice that W is guaranteed to be (r,q)-neighbour-dense by our definition of $i(a,b)$. Since G is (k,t,s,ε)-clique-dense by assumption, this implies that W is (tr,r,q',s)-mostly neighbour-dense, and we let S be the set that witnesses this as per Definition A.5.2. We bound the probability of the event E by a case analysis based on the size of the set $V^1(a)$. We remark that all probabilities in the calculations that follow are over the choice of $\boldsymbol{\alpha} \sim \mathscr{D}$.

Case 1 ($|V^1(a)| > r/2$): In this case, we simply prove that already the probability of reaching a is small. By definition of $V^1(a)$, we have that $|\beta^1(a)| = |V^1(a)|$. Recall that every answer 1 is necessarily the result of a $s^{-(1+\varepsilon)}$-biased coin flip, and that all these decisions are irreversible. That is, if a path ever decides to set a variable in $V^1(a)$ to 0, then its case is lost and it is guaranteed to miss a. Thus we can upper bound the probability of the event E by the probability that a random $\boldsymbol{\alpha}$ passes through a, and, in particular, by the probability of setting all variables in $\beta^1(a)$ to 1 as follows:

$$\Pr[\mathsf{E}] \le \Pr[\boldsymbol{\alpha} \text{ passes through } a] \le \left(s^{-(1+\varepsilon)}\right)^{|\beta^1(a)|} \le s^{-\varepsilon|V^1(a)|} \le 2s^{-\varepsilon r/2} \ . \tag{A.9}$$

Case 2 ($|V^1(a)| \le r/2$): For every path α, let $R(\alpha)$ denote the set of vertices u set to 1 by the path α at some node between a and b (with a included and b excluded); note that $R(\alpha) = \emptyset$ if α does not go through a and b, and that $|R(\alpha)| \le \lceil k/t \rceil$ for all paths α that satisfy the event E. For the sets

$$\mathcal{R}_0 = \{R : |R| \le \lceil k/t \rceil \text{ and } \left|\widehat{N}_W(R \cup V^1(a))\right| < q'\} \tag{A.10a}$$

$$\mathcal{R}_1 = \{R : |R| \le \lceil k/t \rceil \text{ and } \left|\widehat{N}_W(R \cup V^1(a))\right| \ge q'\} \tag{A.10b}$$

we have that

$$\Pr[\mathsf{E}] = \Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_0] + \Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_1] \ . \tag{A.11}$$

The first term in (A.11) is bounded from above by the probability of $R(\boldsymbol{\alpha}) \in \mathcal{R}_0$. Note that $|R| \le \lceil k/t \rceil \le 2k/t \le tr/2$ (since $r \ge 4k/t^2$) for $R \in \mathcal{R}_0$. Hence we have

$|R \cup V^1(a)| \leq tr/2 + r/2 \leq tr$ and therefore $|(R \cup V^1(a)) \cap S| \geq r$ by the choice of S. Thus, the probability of $R(\boldsymbol{\alpha}) \in \mathcal{R}_0$ is bounded by the probability that $|R(\boldsymbol{\alpha}) \cap S| \geq r/2$ since $|V^1(a)| \leq r/2$. But since S is small, we can now apply the union bound and conclude that

$$\begin{aligned}
\Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_0] &\leq \Pr[R(\boldsymbol{\alpha}) \in \mathcal{R}_0] &\text{(A.12)}\\
&\leq \Pr[|R(\boldsymbol{\alpha}) \cap S| \geq r/2] &\text{(A.13)}\\
&\leq \binom{|S|}{r/2} (s^{-(1+\varepsilon)})^{r/2} &\text{(A.14)}\\
&\leq |S|^{r/2} s^{-(1+\varepsilon)r/2} &\text{(A.15)}\\
&\leq s^{-\varepsilon r/2} \quad , &\text{(A.16)}
\end{aligned}$$

where for (A.14) we used the same "irreversibility" argument as in Case 1.

We now bound the second term in (A.11). First note that, by definition of W, if α is a path that passes through a and b in this order, then all $u \in W$ must be set to 0 in α at some node between a and b. For each path in the support of $\mathscr{D}$ that passes through a and b, some of the vertices in W will be set to zero as a result of a coin flip and others will be forced choices.

Fix a path α contributing to the second term in (A.11). We claim that along this path all the $\geq q'$ variables in $\widehat{N}_W(R(\alpha) \cup V^1(a))$ are set to 0 as a result of a coin flip. Indeed, since $V^1_{i^*}(b) = \emptyset$ and i^* is not forgotten at b, by the monotonicity property the same holds for every node along α before b. This implies that the answer to a query of the form x_u ($u \in W$) made along α cannot be forced by neither item 1 (forgetfulness) in the definition of $\mathscr{D}$ nor by a functionality axiom. Moreover, since $V^1(c) \subseteq R(\alpha) \cup V^1(a)$ for any node c on the path α between a and b, it holds that all variables x_u with $u \in \widehat{N}_W(R(\alpha) \cup V^1(a))$ can not be forced to 0 by an edge axiom either.

Now the analysis of the second term in (A.11) is completed by the same Markov chain argument as in Case 1 above (noting that irreversibility of decisions still takes place):

$$\begin{aligned}
\Pr[\mathsf{E} \text{ and } R(\boldsymbol{\alpha}) \in \mathcal{R}_1] &\leq \Pr[\boldsymbol{\alpha} \text{ flips } \geq q' \text{ coins and gets 0-answers}] &\text{(A.17)}\\
&\leq (1 - s^{-(1+\varepsilon)})^{q'} &\text{(A.18)}\\
&\leq s^{-\varepsilon r/2} \quad . &\text{(A.19)}
\end{aligned}$$

Adding (A.16) and (A.19) we obtain the lemma. □

A.6 Random Graphs Are Almost Surely Clique-Dense

In this section we show that asymptotically almost surely an Erdős-Rényi random graph $\boldsymbol{G} \sim \mathscr{G}(n,p)$ is (k,t,s,ε)-clique-dense for the right choice of parameters.

Theorem A.6.1. *For any real constant $\varepsilon \in (0, 1/4)$, any sufficiently large integer n, any positive integer $k \le n^{1/4-\varepsilon}$, and any real $\xi > 1$, if $\boldsymbol{G} \sim \mathscr{G}(n, n^{-2\xi/(k-1)})$ is an Erdős-Rényi random graph then with probability at least $1 - \exp(-\sqrt{n})$ it holds that $\boldsymbol{G}$ is (k, t, s, ε)-clique-dense with $t = 32\xi/\varepsilon$ and $s = \sqrt{n}$.*

As a corollary of Theorem A.5.5 and Theorem A.6.1 we obtain Theorem A.4.1, the main result of this paper.

Proof of Theorem A.4.1. Clearly $t \ge 1$ as required by Definition A.5.3. We can also assume w.l.o.g. that $t \le k$ since otherwise $k/\xi^2 \le 32/(\xi\epsilon) \le O(1)$ and the bound becomes trivial. By plugging in the parameters given by Theorem A.6.1 to Theorem A.5.5 we immediately get the stated lower bound on the length of any regular refutation π of $Clique_{\text{block}}(\boldsymbol{G}, k)$

$$|\pi| \ge \Omega\big(s^{\varepsilon k/t^2}\big) \ge n^{\Omega(k/\xi^2)} \ . \qquad \square$$

We will spend the rest of this section proving Theorem A.6.1.

Let $\delta = 2\xi/(k-1)$. We show that, with probability at least $1 - \mathrm{e}^{-\sqrt{n}}$, the random graph $\boldsymbol{G}$ is (k, t, s, ε)-clique-dense for parameters as in the statement of the theorem, $r = 4k/t^2$ and $q = \frac{n^{1-t\delta r}}{4kt}$.

Recall that $q' = \varepsilon r s^{1+\varepsilon} \log s$. Let us argue that the parameters we use satisfy constraints

$$t\delta r \le \frac{\varepsilon}{2} \ , \tag{A.20}$$

$$\log k + tr \log n \le \frac{n^{1-t\delta r}}{32k} \cdot \frac{2\log n}{n^{1/2}} \ , \tag{A.21}$$

$$\frac{qn^{-t\delta r}s}{16tr} \ge \frac{n^{1+\varepsilon}}{256} \ , \tag{A.22}$$

$$q' \le \frac{qn^{-t\delta r}}{4} \cdot \frac{\log n}{n^{\varepsilon/2}} \ , \tag{A.23}$$

$$tr \le \frac{q}{2} \ , \tag{A.24}$$

which will be used further on in the proof.

As a first step note that

$$t\delta r = \frac{8\xi k}{t(k-1)} \le \frac{\varepsilon}{2} \ , \tag{A.25}$$

and hence (A.20) holds. Equation (A.21) follows from the chain of inequalities

$$\log k + tr \log n \le 2tr \log n = \frac{8k \log n}{t} \le \frac{k \log n}{16} \le \frac{n^{1-t\delta r}}{32k} \cdot \frac{2\log n}{n^{1/2}} \ . \tag{A.26}$$

To obtain (A.22) observe that

$$\frac{qn^{-t\delta r}s}{16tr} = \frac{n^{1-2t\delta r+1/2}}{256k^2} \ge \frac{n^{1-2t\delta r+2\varepsilon}}{256} \ge \frac{n^{1+\varepsilon}}{256} \ . \tag{A.27}$$

To see that (A.23) holds, note that

$$q' = \frac{2\varepsilon k n^{(1+\varepsilon)/2}\log n}{t^2} \le \frac{k^2 n^{(1+\varepsilon)/2}\log n}{16kt} \le \frac{n^{1-3\varepsilon/2}\log n}{16kt} \le \frac{qn^{-t\delta r}}{4}\cdot\frac{\log n}{n^{\varepsilon/2}} \ . \tag{A.28}$$

Finally, for (A.24), we just observe that

$$tr = \frac{4k}{t} \le \frac{k^3}{8k^2} \le \frac{n^{1-t\delta r}}{8kt} = \frac{q}{2} \ , \tag{A.29}$$

using the fact that $k \ge t$ and $k^3 \le n^{1-t\delta r}$.

We must now prove that asymptotically almost surely $\boldsymbol{G}$ is (k,t,s,ε)-clique-dense for the chosen parameterss. All probabilities in this section are over the choice of $\boldsymbol{G}$, and all previously introduced concepts like $\widehat{N}_W(R)$, neighbour-denseness etc. should be understood with respect to $\boldsymbol{G}$ as well (so that they are actually random variables and events in this sample space). Let $V = V(\boldsymbol{G})$ and $V_1 \cup \cdots \cup V_k = V$ be a balanced k-partition of V.

The fact that asymptotically almost surely V_i is (tr,tq)-neighbour-dense for all $i \in [k]$ is quite immediate. First, for any $i \in [k]$ and any $R \subseteq V$ with $|R| \le tr$,

$$\mathbb{E}\left[\left|\widehat{N}_{V_i}(R)\right|\right] = |V_i \setminus R| n^{-\delta|R|} \ge \left(\frac{n}{k} - tr\right)n^{-\delta tr} \ge \left(\frac{n}{k} - \frac{q}{2}\right)n^{-\delta tr} \ge \frac{n^{1-\delta tr}}{2k} \ , \tag{A.30}$$

where (A.30) follows from (A.24) and the trivial fact that $q \le \frac{n}{k}$. Hence, we can bound the probability that there exists an $i \in [k]$ such that V_i is not (tr,tq)-neighbour-dense by

$$\begin{aligned}
\Pr\left[\exists i \in [k]\ \exists R \subseteq V,\ |R| = \lfloor tr \rfloor \wedge \left|\widehat{N}_{V_i}(R)\right| \le tq\right]& \\
&\le k\binom{n}{tr}\max_{i,R}\Pr\left[\left|\widehat{N}_{V_i}(R)\right| \le tq\right] && \text{(A.31)}\\
&\le kn^{tr}\max_{i,R}\Pr\left[\left|\widehat{N}_{V_i}(R)\right| \le \frac{n^{1-t\delta r}}{4k}\right] && \text{(A.32)}\\
&\le kn^{tr}\exp\left(-\frac{n^{1-t\delta r}}{16k}\right) && \text{(A.33)}\\
&\le \exp\left(-\frac{n^{1-t\delta r}}{32k}\cdot\left(2-2\frac{\log n}{n^{1/2}}\right)\right) && \text{(A.34)}\\
&\le \mathrm{e}^{-\sqrt{n}} \ . && \text{(A.35)}
\end{aligned}$$

We note that (A.31) is a union bound, (A.32) follows from the definition of q, (A.33) is the multiplicative form of Chernoff bound (note that the events $v \in \widehat{N}_{V_i}(R) (v \in V \setminus R)$ are mutually independent), (A.34) follows from (A.21), and (A.35) holds for large enough n by (A.20) and the fact that $\varepsilon < 1/4$ and $k < n^{1/4}$.

All that is left to prove is that asymptotically almost surely $\boldsymbol{G}$ satisfies property 2 in Definition A.5.3, that is that every (r,q)-neighbour-dense set $W \subseteq V$ is (tr,r,q',s)-mostly neighbour-dense. For shortness let P be the event that $\boldsymbol{G}$ satisfies this property. We wish to show that $\Pr[\neg\mathsf{P}] \leq \mathrm{e}^{-\Omega(n)}$, and it turns out that due to our choice of parameters we can afford to use the crude union bound over all 2^n choices of W.

To be more specific, let $Q(W)$ denote the event that W is (r,q)-neighbour-dense. Given an (r,q)-neighbour-dense set $W \subseteq V$ we will define a set S_W which will be a "candidate witness" of the fact that W is (tr,r,q',s)-mostly neighbour-dense. First observe that, since W is (r,q)-neighbour-dense and $q' \leq q$ by (A.23), any set $R \subseteq V$ with $|R| \leq tr$ and $\left|\widehat{N}_W(R)\right| \leq q'$ must be such that $|R| > r$. We will use a sequence of such sets R and construct S_W in a greedy fashion. To this end, the following definition will be useful. A tuple of sets $(R_1,\ldots,R_m)$ is said to be *r-disjoint* if $\left|R_i \cap \left(\bigcup_{j<i} R_j\right)\right| \leq r$ for every $i \in [m]$.

Fix an arbitrary ordering of the subsets of V. Define $\vec{R}_W = (R_1,\ldots,R_m)$ to be a maximally long tuple such that, for every $i = 1,\ldots,m$, the set R_i is the first in the ordering such that $|R_i| \leq tr$, $\left|\widehat{N}_W(R_i)\right| \leq q'$ and $\left|R_i \cap \left(\bigcup_{j<i} R_j\right)\right| \leq r$. Note that $\vec{R}_W$ is r-disjoint. Now let $S_W = \bigcup_{i\leq m} R_i$.

Observe that, by maximality of $\vec{R}_W$, any set $R \subseteq V$ with $|R| \leq tr$ and $\left|\widehat{N}_W(R)\right| \leq q'$ must be such that $|R \cap S| > r$. This implies that if $|S_W| \leq s$ then S_W witnesses the fact that W is (tr,r,q',s)-mostly neighbour-dense. Therefore we have that

$$\Pr[\neg\mathsf{P}] \leq \Pr[\exists W \subseteq V,\ Q(W) \wedge |S_W| > s]\ . \tag{A.36}$$

Moreover, let $\mathcal{W}$ be the collection of all pairs $(W,\vec{R})$ such that $W \subseteq V$, $\vec{R} = (R_1,\ldots,R_\ell)$ for $\ell = \lceil s/tr\rceil$, $R_j \subseteq V$ and $0 < |R_j| \leq tr$ for each $j \in [\ell]$, and $\vec{R}$ is r-disjoint. Notice that if there exists an (r,q)-neighbour-dense W such that $\vec{R}_W = (R_1,\ldots,R_m)$ and $|S_W| > s$, then $m \geq \ell$ and $(W,(R_1,\ldots,R_\ell)) \in \mathcal{W}$. Furthermore, by definition of $\vec{R}_W$, for every $j \in [\ell]$ it holds that $\left|\widehat{N}_W(R_j)\right| \leq q'$. Hence we can conclude that

$$\Pr[\neg\mathsf{P}] \leq \Pr\left[\exists(W,\vec{R}) \in \mathcal{W},\ Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq q'\right] \tag{A.37}$$

$$\leq 2^n n^{tr\ell} \max_{(W,\vec{R})\in\mathcal{W}} \Pr\left[Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq q'\right] \tag{A.38}$$

$$\leq 2^n n^{s} \max_{(W,\vec{R})\in\mathcal{W}} \Pr\left[Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq \frac{q}{4} n^{-t\delta r}\right]\ , \tag{A.39}$$

where (A.39) follows for n large enough from the bound in (A.23).

Now fix $(W,\vec{R}) \in \mathcal{W}$ and let R_j^d (resp. R_j^c) be the subset of R_j disjoint from (resp. contained in) $\bigcup_{j'<j} R_{j'}$. Since $|R_j^c| \leq r$ by definition, it holds that if W is (r,q)-neighbour-dense then $\left|\widehat{N}_W(R_j^c)\right| > q$. Let $\mathsf{F}(j)$ be the event that $\left|\widehat{N}_W(R_j^c)\right| > q$ and $\left|\widehat{N}_W(R_j)\right| \leq \frac{q}{4}n^{-t\delta r}$. Note that $\Pr\left[Q(W) \wedge \forall j \in [\ell],\ \left|\widehat{N}_W(R_j)\right| \leq \frac{q}{4}n^{-t\delta r}\right]$ is at most $\Pr\left[\forall j \in [\ell],\ \mathsf{F}(j)\right]$. Let $\mathsf{F}'(j)$ be the event that $\mathsf{F}(j')$ holds for all $j' \in [j-1]$. We have that

$$\Pr\left[\forall j \in [\ell],\ \mathsf{F}(j)\right] = \prod_{j\in[\ell]} \Pr\left[\mathsf{F}(j) \mid \mathsf{F}'(j)\right]\ . \tag{A.40}$$

We can consider the factors of the previous product separately and bound each one by

$$\Pr\Big[\mathsf{F}(j) \;\Big|\; \mathsf{F}'(j)\Big] \le \sum_{\substack{U\subseteq W\\|U|\ge q}} \Pr\Big[\big|\widehat{N}_U(R_j^d)\big| \le \frac{q}{4}n^{-t\delta r} \;\Big|\; \widehat{N}_W(R_j^c) = U \wedge \mathsf{F}'(j)\Big] \cdot \Pr\Big[\widehat{N}_W(R_j^c) = U \;\Big|\; \mathsf{F}'(j)\Big] \tag{A.41}$$

$$\le \sum_{\substack{U\subseteq W\\|U|\ge q}} \Pr\Big[\big|\widehat{N}_U(R_j^d)\big| \le \frac{q}{4}n^{-t\delta r}\Big] \cdot \Pr\Big[\widehat{N}_W(R_j^c) = U \;\Big|\; \mathsf{F}'(j)\Big] \tag{A.42}$$

$$\le \sum_{\substack{U\subseteq W\\|U|\ge q}} \exp\left(-\frac{qn^{-t\delta r}}{16}\right) \cdot \Pr\Big[\widehat{N}_W(R_j^c) = U \;\Big|\; \mathsf{F}'(j)\Big] \tag{A.43}$$

$$= \exp\left(-\frac{qn^{-t\delta r}}{16}\right) \cdot \sum_{\substack{U\subseteq W\\|U|\ge q}} \Pr\Big[\widehat{N}_W(R_j^c) = U \;\Big|\; \mathsf{F}'(j)\Big] \tag{A.44}$$

$$\le \exp\left(-\frac{qn^{-t\delta r}}{16}\right) . \tag{A.45}$$

Equation (A.42) follows from the independence of any two events that involve disjoint sets of potential edges and (A.43) follows from the multiplicative Chernoff bound and the fact that

$$\mathbb{E}\Big[\big|\widehat{N}_U(R_j^d)\big|\Big] = |U \setminus R_j^d| n^{-\delta |R_j^d|} \ge (|U| - tr)n^{-\delta tr} \ge \frac{q}{2}n^{-\delta tr} . \tag{A.46}$$

So, putting everything together, we have that

$$\Pr[\neg \mathsf{P}] \le 2^n n^s \exp\left(-\frac{qn^{-t\delta r}\ell}{16}\right) \le \mathrm{e}^{(\log 2)n + \sqrt{n}\log n - (n^{1+\varepsilon})/256} \le \mathrm{e}^{-\Omega(n)} , \tag{A.47}$$

where the last inequality holds for n large enough, and the second to last inequality follows immediately from the bound in (A.22). This concludes the proof of Theorem A.6.1.

A.7 State-of-the-Art Algorithms for Clique

In this section we describe state-of-the-art algorithms for maximum clique and explain how regular resolution proofs bound from below the running time of these algorithms.

At the heart of most (if not all) of the state-of-the-art algorithms for maximum clique is a backtracking search, which in its simplest form examines all maximal cliques by enlarging a set of vertices that form a clique and backtracking when it certifies that the current set forms a maximal clique. A classical example of such a backtracking search is the BronKerbosch [BK73] algorithm which enumerates all maximal cliques in a graph.

This algorithm can be adapted to find a maximum clique as done in [CP90] improving the running time considerably by using a branch and bound strategy. At some point in the search tree it becomes clear that the current search-branch will not lead to a clique larger than the largest one found so far—in such cases the algorithm cuts off the search and backtracks immediately.

The most successful algorithms in practice are search trees with clever branch and bound strategies. In this section we will discuss the algorithm by Östergård [Öst02] using Russian doll search and a collection of algorithms that use colour-based branch and bound strategies [Woo97, Fah02, TS03, TK07, KJ07, TSH$^+$10, ST10, SRJ11, SMRH13, SLB14, SLB$^+$16, TYH$^+$16].

Östergård's algorithm Östergård's algorithm [Öst02] is a branch and bound algorithm that uses Russian doll search as a pruning strategy: it considers smaller subinstances recursively and solves them in ascending order using previous solutions as upper bounds. This algorithm, which is the main component of the Cliquer software, is often used in practice and has been available online since 2003 [NÖ03]. Cliquer is also the software of choice to compute maximal cliques in the open source mathematical software SageMath [S$^+$17].

The `Cliquer(`G`)` algorithm described in Figure 2 is essentially the same as Algorithm 2 in [Öst02]. The algorithm first permutes the vertices of G according to some criteria. Let $v_1, \ldots, v_n$ be the enumeration of $V(G)$ induced by said permutation, and $V_i = \{v_i, \ldots, v_n\}$ for $i \in [n]$. In practice this permutation has a large impact on the running time of the algorithm, but for our analysis the knowledge of the specific order is irrelevant. In the main loop (lines 5–8) subgraphs of G are considered and at each iteration the size of a maximum clique containing only vertices of V_i is stored in *bounds*$[i]$. The algorithm keeps the best solution (largest clique) found so far in the global variable *incumbent* which is initially empty. The array *bounds* and the flag *found* are global variables. The current growing clique is stored in *solution* and passed as an argument of the subroutine `expand` together with the current subgraph $H \subseteq G$ being considered. The main subroutine `expand` recursively goes through all vertices of H from smallest to largest index. First note that if the size of the current growing clique plus $|H|$ is not larger than the current maximum clique (line 13) then this branch can be cut. Moreover, if v_i is the smallest-index vertex in H then $V(H) \subseteq V_i$ and *bounds*$[i]$ is an upper bound on the size of a maximum clique in H. This implies that this branch can be cut if the size of the current growing clique plus *bounds*$[i]$ is not larger than the current maximum clique (line 15). If it is larger, the algorithm branches on the vertex v_i. First v_i is taken to be part of the solution: it is added to (a copy of) the current growing solution, (a copy of) the graph is updated to contain only neighbours of v_i and a recursive call is made (lines 16–18). If the recursive call finds a clique larger than the current largest clique, it sets the flag *found* to true. This allows the algorithm can return to the main routine (line 8) since a maximum clique containing only vertices of V_i can be at most one unit larger than a maximum clique containing only vertices of

Algorithm 2 `Cliquer`(G) algorithm

```
1  Cliquer($G$):
2  **begin**
3      $G \leftarrow$ permute($G$)
4      $incumbent \leftarrow \emptyset$
5      **for** $i = n$ ***down to*** $1$ **do**
6          $found \leftarrow$ false
7          expand($G[V_i \cap N(v_i)], \{v_i\}$)
8          $bounds[i] \leftarrow |incumbent|$
9      **return** $incumbent$
10 expand($H, solution$):
11 **begin**
12     **while** $V(H) \neq \emptyset$ **do**
13         **if** $|solution| + |V(H)| \leq |incumbent|$ **then return**
14         $i \leftarrow \min\{j \mid v_j \in V(H)\}$
15         **if** $|solution| + bounds[i] \leq |incumbent|$ **then return**
16         $solution' \leftarrow solution \cup \{v_i\}$
17         $V' \leftarrow V(H) \cap N(v_i)$
18         expand($H[V'], solution'$)
19         **if** $found =$ true **then return**
20         $H \leftarrow H \setminus \{v_i\}$
21     **if** $|solution'| > |incumbent|$ **then**
22         $incumbent \leftarrow solution'$
23         $found \leftarrow$ true
24     **return**
```

V_{i+1}. If no larger clique was found, the algorithm then proceeds to the opposite branch choice, that is, taking vertex v_i to not be in the solution (line 20) and considering the next vertex in the ordering. If $V(H)$ is empty and a larger clique has been found, the best solution so far is updated and the flag *found* is set to true (lines 22–23).

We now argue that the running time of the `Cliquer`(G) algorithm is bounded from below by the size of a regular resolution refutation of $Clique_{\text{block}}(G,k)$ up to a constant factor. First note that a straightforward modification of the `Cliquer`(G) algorithm gives an algorithm that determines whether G contains a block-respecting k-clique.

Given a graph G that does not contain a block-respecting k-clique, the last call of the subroutine `expand` in the main loop (lines 5–8, when i is set to 1) can be represented by an ordered decision tree with labelled leafs. A decision tree is said to be ordered if there exists a linear ordering of the variables such that if x is queried before y then $x \prec y$. In our setting, the order is determined by the permutation of the vertices, and without loss of generality we assume $v_i \prec v_j$ if $i < j$. For each leaf, if R is the set of vertices identified as clique members by the branch leading to this leaf, then the leaf is labelled either by a pair (u, v) such that $u, v \in R$ and there is no edge between u and v or by an index $\ell \in [k]$ such that all vertices in the ℓth block are outside the clique, or by a vertex v_i such that $i = \min\{j \mid v_j \in N(R)\}$ and the largest clique containing only vertices of V_i has size at most $k - |R| - 1$. For each vertex v_i that labels some leaf, we construct the decision tree corresponding to the ith call of the subroutine `expand`.

In order to weave these decision trees into a read-once branching program, at each leaf labelled v_i we query to all non yet queried vertices v_j such that $j < i$ and v_j is in the same block as v_i. Let B_i denote the set of vertices. Observe that taking any vertex in B_i to be in the clique yields an immediate contradictions since $B_i \cap N(R) = \emptyset$ by definition of i. Moreover note that the branch leading to the leaf where all of B_i is taken to be outside the clique does not contain any query to vertices in V_i. We can therefore identify this leaf with the root of the decision tree corresponding to v_i and still maintain regularity. After repeating this procedure at every leaf labelled by some vertex, only leafs labelled by indices $\ell \in [k]$ and by pairs (u, v) remain, which have a direct correspondence to falsified clauses of $Clique_{\text{block}}(G, k)$. Therefore, the directed graph obtained by this process corresponds to a read-once branching program that solves the falsified clause search problem on $Clique_{\text{block}}(G, k)$ and the bound on the running time follows immediately.

Colour-based branch and bound algorithms We consider a class of algorithms which are arguably the most successful in practice. An extended survey together with a computational analysis of algorithms published until 2012 can be found in [Pro12] and an overview of algorithms reported since then in [McC17]. These algorithms are branch and bound algorithms that use colouring as a bounding—and often also as a branching—strategy. The basic idea is that if a graph can be coloured with ℓ colours then it does not contain a clique larger than ℓ.

The `MaxClique`(G) algorithm described in Figure 3, a generalized version of Algorithm 2.1 in [McC17], is a basic maximum clique algorithm which uses a colour-based branch and bound strategy. The algorithm keeps the best solution (largest clique) found so far in the global variable *incumbent* which is initially empty. The current growing clique is stored in *solution* and passed as an argument of the subroutine `expand` together with the current subgraph $H \subseteq G$ being considered. The subroutine `colourOrder`(H) (line 8) returns an ordering of the vertices in H, say $v_1, v_2, \ldots, v_n$, and for every $i \in [n]$ an upper bound on the number of colours needed to colour the graph induced by vertices v_1 to v_i. The vertices are then considered in reverse order. If the vertex v is being considered and the size of the current growing clique plus the (upper bound on the) number of colours needed to colour the remaining graph is not larger than the current maximum clique (line 11) then this branch can be cut. If it is larger, the algorithm branches on the vertex v. First v is taken to be part of the solution: v is added to (a copy of) the current growing solution, (a copy of) the graph is updated to contain only

Algorithm 3 MaxClique(G) algorithm

```
 1 MaxClique($G$):
 2 begin
 3     global $incumbent \leftarrow \emptyset$
 4     expand($G, \emptyset$)
 5     return $incumbent$
 6 expand($H, solution$):
 7 begin
 8     $(order, bounds) \leftarrow$ colourOrder($H$)
 9     while $V(H) \neq \emptyset$ do
10         $i \leftarrow |V(H)|$
11         if $|solution| + bounds[i] \leq |incumbent|$ then return
12         $v \leftarrow order[i]$
13         $solution' \leftarrow solution \cup \{v\}$
14         $V' \leftarrow V(H) \cap N(v)$
15         expand($H[V'], solution'$)
16         $H \leftarrow H \setminus \{v\}$
17     if $|solution'| > |incumbent|$ then $incumbent \leftarrow solution'$
18     return
```

neighbours of v and a recursive call is made (lines 13–15). If the recursive call finds a clique larger than the current largest clique, the best solution so far is updated (line 17). The algorithm proceeds to the opposite branch choice, that is, considering vertex v not in the solution (line 16). Returning to the loop the algorithm continues to consider the next vertex in the ordering.

It was reported in [CZ12] that it is possible to capture the algorithms for solving the maximum clique problem in [CP90, Fah02, TS03, TK07, KJ07, TSH$^+$10] in a same framework. The general algorithm they present is an iterative version of the MaxClique(G) algorithm. We observe that MaxClique(G) captures also the more recent algorithms in [ST10, SRJ11, SMRH13, SLB14, SLB$^+$16, TYH$^+$16]. The differences in these algorithms reside in the colouring procedure and in how the graph operations are implemented (see [Pro12, McC17] for details). For our purpose, that is, in order to show that the running time of these algorithms can be bounded from below by the length of the shortest regular resolution refutation of the k-clique formula, we assume that the colouring algorithm and the graph operations take constant time and prove the lower bound for this general framework. Moreover, we can assume that optimal colouring bounds and optimal ordering of vertices are given.

We now argue that the running time of the MaxClique(G) algorithm is bounded from below by the size of a regular resolution refutation of $Clique_{\text{block}}(G, k)$ up to a multiplicative factor of $2^k n^{O(1)}$. We first note that a straightforward modification of the `MaxClique`(G) algorithm gives an algorithm, which we refer to as `Clique`(G,k), that determines whether G contains a k-clique. Given a graph G that does not contain a k-clique, an execution of `Clique`(G,k) can be represented by a search tree with leafs labelled by a subgraph $H \subseteq G$ of potential clique-members and a number q such that the branch leading to this leaf has identified $k-q$ clique members, has not queried any vertex of H, and H is $(q-1)$-colourable. Note that a read-once branching program can simulate this search tree and, by Proposition A.3.3 and the equivalence between read-once branching programs and regular resolution, at each leaf establish that H does not contain a q-clique in size at most $2^q \cdot q^2 \cdot |V(H)|^2$. The bound on the running time follows directly. Observe that establishing that H does not contain a q-clique is done in a read-once fashion by querying only vertices of H. Since the vertices of H were not queried earlier on this branch, the whole branching program is read-once.

A.8 Concluding Remarks

In this paper we prove optimal average-case lower bounds for regular resolution proofs certifying k-clique-freeness of Erdős-Rényi graphs not containing k-cliques. These lower bounds are also strong enough to apply for several state-of-the-art clique algorithms used in practice.

The most immediate and compelling question arising from this work is whether the lower bounds for regular resolution can be strengthened to hold also for general resolution. A closer study of our proof reveals that there are several steps that rely on regularity. However, there is no connection per se between regular resolution and the abstract combinatorial property of graphs that we show to be sufficient to imply regular resolution lower bounds. Thus, it is tempting to speculate that this property, or perhaps some modification of it, might be sufficient to obtain lower bounds also for general resolution. If so, a natural next step would be to try to extend the lower bound further to the polynomial calculus proof system capturing Gröbner basis calculations.

Another interesting question is whether the lower bounds we obtain asymptotically almost surely for random graphs can also be shown to hold deterministically under the weaker assumption that the graph has certain pseudorandom properties. Specifically, is it possible to get an $n^{\Omega(\log n)}$ length lower bound for the class of Ramsey graphs? A graph on n vertices is called *Ramsey* if it has no set of $\lceil 2 \log_2 n \rceil$ vertices forming a clique or independent set. It is known that for sufficiently large n a random graph sampled from $\mathcal{G}(n,1/2)$ is Ramsey with high probability. Is it true that for a Ramsey graph G on n vertices the formula $Clique(G, \lceil 2 \log_2 n \rceil)$ requires (regular) resolution refutations of length $n^{\Omega(\log n)}$? Such a lower bound is known for tree-like resolution [LPRT17] and proving it for general resolution would have interesting consequences in other areas of proof complexity [DMS11].

Acknowledgements

This work has been a long journey, and different subsets of the authors want to acknowledge fruitful and enlightening discussions with different subsets of Christoph Berkholz, Olaf Beyersdorff, Nicola Galesi, Ciaran McCreesh, Toni Pitassi, Pavel Pudlák, Ben Rossman, Navid Talebanfard, and Neil Thapen. A special thanks to Shuo Pang for having pointed out an inaccuracy in the probabilistic argument in Section A.6 and having suggested a fix.

The first, second, and fourth authors were supported by the European Research Council under the European Union's Horizon 2020 Research and Innovation Programme / ERC grant agreement no. 648276 AUTAR. The third and fifth authors were supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611 as well as by Swedish Research Council grants 621-2012-5645 and 2016-00782, and the second author did part of this work while at KTH Royal Institute of Technology supported by the same grants. The last author was supported by the Russian Foundation for Basic Research.

B

Paper B

How Limited Interaction Hinders Real Communication (and What it Means for Proof and Circuit Complexity)

Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals

Full length version of the article published in Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS ’16), October 2016, pp. 295–304.

Abstract

We obtain the first true size-space trade-offs for the cutting planes proof system, where the upper bounds hold for size and total space for derivations with constant-size coefficients, and the lower bounds apply to length and formula space (i.e., number of inequalities in memory) even for derivations with exponentially large coefficients. These are also the first trade-offs to hold uniformly for resolution, polynomial calculus and cutting planes, thus capturing the main methods of reasoning used in current state-of-the-art SAT solvers.

We prove our results by a reduction to communication lower bounds in a round-efficient version of the real communication model of [Krajíček ’98], drawing on and extending techniques in [Raz and McKenzie ’99] and [Göös et al. ’15]. The communication lower bounds are in turn established by a reduction to trade-offs between cost and number of rounds in the game of [Dymond and Tompa ’85] played on directed acyclic graphs.

As a by-product of the techniques developed to show these proof complexity results, we obtain a separation between $\mathsf{monotone}\text{-}\mathsf{AC}^{i-1}$ and $\mathsf{monotone}\text{-}\mathsf{NC}^{i}$, and an exponential separation between $\mathsf{monotone}\text{-}\mathsf{AC}^{i-1}$ and $\mathsf{monotone}\text{-}\mathsf{AC}^{i}$, improving exponentially over the superpolynomial separation in [Raz and McKenzie ’99].

B.1 Introduction

Ever since the discovery of NP-completeness by Cook and Levin in [Coo71, Lev73], the problem of how hard it is to decide satisfiability of formulas in propositional logic has played a leading role in theoretical computer science. Although the conventional wisdom is that SAT should be a very hard problem—to the extent that the *Exponential Time Hypothesis* [IP01] concerning its worst-case complexity is a standard assumption used in many other hardness results—essentially no non-trivial lower bounds on the time complexity of the SAT problem are known.

A less ambitious goal is to ask for lower bounds if not only the running time but also the memory usage of the algorithm is restricted. Yet it took until [For00] to rule out a linear-time, logarithmic-space algorithm for SAT. Later research has shown that refuting unsatisfiable formulas on random-access machines cannot be done non-deterministically in simultaneous time $n^{4^{1/3}}$ and space $n^{o(1)}$ [DvMW11] and SAT cannot be decided deterministically in simultaneous time $n^{1.8}$ and space $n^{o(1)}$ [Wil08]. On Turing machines, no non-deterministic algorithm solving SAT in time T and space s can achieve $T \cdot s = n^2/\log^3 n$ [San01]. (See [vM07] for a good survey of the area with more details on this kind of results.)

For a problem that is believed to require exponential time, the results listed above might not seem very impressive. Yet they should not necessarily be viewed only as an illustration of the weaknesses of current techniques for proving lower bounds. It is important to realize that the adversary is formidable—applied research in the last 15–20 years has led to the development of amazingly efficient algorithms, so-called *SAT solvers*, that solve many real-world instances with millions of variables, and do so in linear time. Today, practitioners often think of SAT as an *easy problem to reduce to*, rather than a hard problem to reduce from (we refer the reader to [BHvMW09] for more on this fascinating topic).

Virtually the only tool currently available for a rigorous analysis of the performance of such SAT solvers is *proof complexity* [CR79], where one studies the methods of reasoning used by the corresponding algorithms. The transcript of the computations made can be viewed as a formal proof applying the relevant method of reasoning, and proof complexity analyses the resources needed when all computational choices are made optimally (i.e., non-deterministically). Even though this is quite a challenging adversarial setting, proof complexity has nevertheless managed to give tight exponential lower bounds on the worst-case running time for many approaches for SAT used in practice by lower-bounding proof size.

The focus of this paper is on time-space trade-offs in computational models describing current state-of-the-art SAT solvers. This research is partly driven by SAT solver running time and memory usage—in practice, space consumption can be almost as much of a bottleneck as running time—but is also motivated by the fundamental importance of time and space complexity in computational complexity.

B.1.1 Previous Work on Proof Complexity Trade-offs

In *resolution* [Bla37], which is arguably the most well-studied proof system in proof complexity, the input is an unsatisfiable formula in conjunctive normal form (CNF) and new disjunctive clauses are derived from this formula until an explicit contradiction is reached (in the form of the empty clause without literals). Resolution is also the method of reasoning underlying the currently most successful SAT solving paradigm based on so-called *conflict-driven clause learning (CDCL)* [BS97, MS99, MMZ^{+}01]. The question of time-space trade-offs for resolution was first raised by Ben-Sasson in 2002 (journal version in [Ben09]), who also obtained such trade-offs for the restricted subsystem of tree-like resolution. Size-space trade-offs for general, unrestricted resolution were later shown in [Nor09b, BN11, BBI16, BNT13].

In contrast to the trade-off results for random-access and Turing machines reviewed above, in these more limited models of computation one can obtain exponential lower bounds on proof size (corresponding to running time) for proofs in sublinear but polynomial space [Nor09b, BN11], and results in [BBI16, BNT13] even exhibit trade-offs where size has to be superpolynomial and space has to be superlinear simultaneously. Another difference is that these results are *true trade-offs* in the sense that it is actually possible to refute the formulas both in small size and small space, only not simultaneously. A third nice feature of the trade-offs are that the upper bounds are on proof size and total space, whereas the (sometimes tightly matching) lower bounds are on *length* and *formula space*, meaning that one only charges one time unit for each derivation step regardless of its complexity, and only one space unit per "formula" (for resolution: per clause) regardless of how large it is. Thus, the upper bounds are algorithmically achievable, while the lower bounds hold in a significantly stronger model.

A stronger proof system than resolution is *polynomial calculus* [CEI96, ABRW02], where the clauses of a formula are translated to multilinear polynomials and calculations inside the ideal generated by these polynomials (basically corresponding to a Gröbner basis computation) establishes unsatisfiability. Among other things, polynomial calculus captures CDCL solvers extended with reasoning about systems of linear equations mod 2. The first size-space trade-offs for polynomial calculus—which were not true trade-offs in the sense discussed above, however—were obtained in [HN12], and these results were further improved in [BNT13] to true trade-offs essentially matching the results cited above for resolution except for a small loss in parameters.

Another proof system that is also stronger than resolution and that has been the focus of much research is *cutting planes* [CCT87], which formalizes the integer linear programming algorithm in [Gom63, Chv73] and underlies so-called *pseudo-Boolean* SAT solvers. In cutting planes the clauses of a CNF formula are translated to linear inequalities, which are then manipulated to derive a contradiction. Thus, the question of Boolean satisfiability is reduced to the geometry of polytopes over the real numbers. Cutting planes is much more poorly understood than resolution and polynomial calculus, however, and size-space trade-offs have proven elusive. The results in [HN12] apply not only to resolution and polynomial calculus but also to cutting planes, and were improved further in [GP18] to hold for even stronger proof systems, but unfortunately are not true trade-offs in the sense discussed above.

The problem is that what is shown in [HN12, GP18] is only that proofs in small space for certain formulas have to be very large, but it is not established that these formulas can be refuted space-efficiently. In fact, for resolution it can be shown using techniques from [BN08] that such small-space proofs provably do not exist, and for polynomial calculus there is circumstantial evidence for a similar claim. As discussed in Section B.3, this turns out to be an inherent limitation of the technique used.

In a recent surprising paper [GPT15], it was shown that cutting planes can refute any formula in *constant* space if we only count the number of lines or formulas. Plugging this result into [HN12, GP18] yields a trade-off of sorts, since "small-space" proofs will always exist, but the catch is that such proofs will have exponentially large coefficients. This means that these trade-offs do not seem very "algorithmically relevant" in the sense that such proofs could hardly be found in practice, and saying that a proof with exponential-size coefficients has "constant space" somehow does not feel quite right.

B.1.2 Our Proof Complexity Contributions

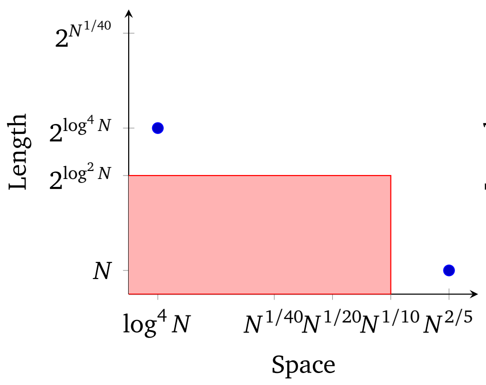
In this paper we report the first true, algorithmically realizable trade-offs for cutting planes, where the upper bounds hold for proof size and total space and the lower bounds apply to proof length and formula space (i.e., number of inequalities). The trade-offs also hold for resolution and polynomial calculus, making them the first trade-offs that hold for essentially all methods of reasoning used in the most successful SAT solvers to date.[1]

Below, we state two examples of the kind of trade-offs we obtain (referring the reader to Section B.2 for the missing formal definitions). In the rest of this section we will focus on cutting planes, since this proof system is the main target of this work. However, all the lower bounds stated also hold for polynomial calculus (and for the strictly weaker proof system resolution), and since all our upper bounds are actually proven in resolution they transfer to both polynomial calculus and cutting planes.

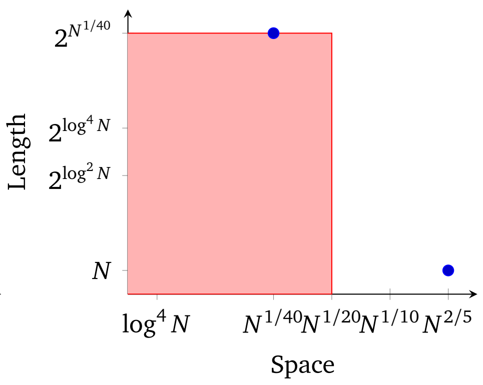
The first result is a "robust trade-off" that holds all the way from polylogarithmic to polynomial space as stated next.

Theorem B.1.1 (Informal). *There exists an explicitly constructible family of* 6*-CNF formulas* $\{F_N\}_{N=1}^{\infty}$ *of size* $\Theta(N)$ *such that:*

[1]We remark that this ignores the issue of formula *preprocessing techniques*, which are heavily used in most state-of-the-art SAT solvers, and some of which potentially require the full extended Frege proof system for a complete formal description (but can also sometimes cause a provable exponential *loss* in reasoning power). Since in practice SAT solvers fail to solve many of the combinatorial benchmark formulas that are hard for resolution, polynomial calculus, and cutting planes but easy for (even non-extended) Frege, however, and since in addition it is usually not hard to come up with formulas that foil any concrete preprocessing techniques actually used, this seems like a reasonable simplification.



(a) Robust trade-off



(b) Exponential trade-off

Figure B.1: Pictorial illustrations of trade-offs in Theorems B.1.1 and B.1.2

1. *F_N can be refuted by cutting planes with constant-size coefficients in size $O(N)$ and total space $O(N^{2/5})$.*

2. *F_N can be refuted by cutting planes with constant-size coefficients in total space $O(\log^4 N)$ and size $2^{O(\log^4 N)}$.*

3. *Any cutting planes refutation of F_N, even with coefficients of unbounded size, in formula space less than $N^{1/10-\epsilon}$ requires length greater than $2^{\Omega(\log^2 N)}$.*

The second trade-off holds over a smaller space range, but causes an exponential and not just superpolynomial blow-up in proof size.

Theorem B.1.2 (Informal). *There exists an explicitly constructible family of 6-CNF formulas $\{F_N\}_{N=1}^{\infty}$ of size $\Theta(N)$ such that:*

1. *F_N can be refuted by cutting planes with constant-size coefficients in size $O(N)$ and total space $O(N^{2/5})$.*

2. *F_N can be refuted by cutting planes with constant-size coefficients in total space $O(N^{1/40})$ and size $2^{O(N^{1/40})}$.*

3. *Any cutting planes refutation of F_N, even with coefficients of unbounded size, in formula space less than $N^{1/20-\epsilon}$ requires length greater than $2^{\Omega(N^{1/40})}$.*

See Figure B.1 for an illustration of these results, where blue dots denote provable upper bounds on time-space parameters of cutting planes refutations and the shaded red areas show ranges of parameters that are impossible to achieve.

B.1.3 Previous Work in Monotone Circuit Complexity

Since this paper also makes contributions to monotone circuit complexity, we next review some relevant background in this area. After superpolynomial lower bounds on the size of monotone circuits computing explicit functions were obtained in [Raz85, And85] (see also [BS90]), the first step towards the natural next goal of establishing a depth hierarchy for monotone circuits was taken in [KW90], proving that connectivity, which is in $\mathsf{monotone\text{-}NC}^2$, requires depth $\Omega(\log^2 n)$ for monotone circuits with fan-in 2. This implies a separation between $\mathsf{monotone\text{-}NC}^1$ and $\mathsf{monotone\text{-}NC}^2$. The same approach was used in [RM99] to prove a separation between $\mathsf{monotone\text{-}NC}^{i-1}$ and $\mathsf{monotone\text{-}NC}^i$ for every i. This result can be rephrased as saying that there is a family of Boolean functions $\{f^i\}$ such that f^i can be computed by monotone circuits of depth $\log^i n$, fan-in 2, and polynomial size but cannot be computed by any monotone circuit of depth $o(\log^i n)$ and fan-in 2.

Going into more details, the function in [RM99] that witnesses the separation between $\mathsf{monotone\text{-}NC}^{i-1}$ and $\mathsf{monotone\text{-}NC}^i$ can be computed by a monotone circuit of depth $\log^{i-1} n$, polynomial fan-in, and polynomial size, and therefore the separation is between $\mathsf{monotone\text{-}NC}^{i-1}$ and $\mathsf{monotone\text{-}AC}^{i-1}$. This separation was later refined in [Joh01] to circuits of semi-unbounded fan-in—i.e., with AND-gates of fan-in 2 and OR-gates of unbounded fan-in, leaving the question of a separation between $\mathsf{monotone\text{-}AC}^{i-1}$ and $\mathsf{monotone\text{-}NC}^i$ open.

We can also view the separation as between $\mathsf{monotone\text{-}AC}^{i-1}$ and $\mathsf{monotone\text{-}AC}^i$, since $\mathsf{monotone\text{-}AC}^{i-1}$ is contained in $\mathsf{monotone\text{-}NC}^i$, however, this separation only guarantees a superpolynomial circuit size lower bound. Furthermore, the function f^i only depends on $\log^{40i} n$ variables and so it can be computed by a monotone DNF of size $2^{\log^{40i} n}$, i.e., there is a quasipolynomial upper bound.

We remark that it is clearly not possible to prove a superpolynomial separation between $\mathsf{monotone\text{-}NC}^{i-1}$ and $\mathsf{monotone\text{-}NC}^i$ in view of the simple fact that circuits in these classes have fan-in 2, and hence it only makes sense to talk about superpolynomial versus exponential separations in the $\mathsf{monotone\text{-}AC}$ hierarchy. It should be noted that exponential separations between monotone circuits of bounded depth were previously known, but only for depth less than logarithmic. It was shown in [KPPY84] that the complete tree of depth k, arity $n^{1/k}$, and size $\Theta(n)$, with alternating levels of AND and OR, requires size $2^{\Omega(n^{1/k}/k)}$ to compute with circuits of depth $k-1$. This result was later reproven in [NW93] using the communication complexity of the pointer jumping function (see also [RY16]).

B.1.4 Our Monotone Circuit Complexity Contributions

In this paper we solve the open problem of [Joh01] by separating $\mathsf{monotone\text{-}AC}^{i-1}$ from $\mathsf{monotone\text{-}NC}^i$.

Theorem B.1.3. *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth $\log^i n$, fan-in 2, and size $O(n)$, but for which every monotone circuit of depth $O(\log^{i-1} n)$ requires superpolynomial size.*

In addition we establish an exponential separation in the monotone-AC hierarchy. More precisely, for each $i \in \mathbb{N}$ we exhibit a Boolean function f^i that can be computed by monotone circuits of depth $\log^i n$ but such that every monotone circuit of depth at most $O(\log^{i-1} n)$ requires size $2^{n^{\Omega(1)}}$ (where the hidden constant in the lower bound depends inversely on that in the upper bound).

Theorem B.1.4. *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth $\log^i n$, fan-in $n^{4/5}$, and size $O(n)$, but for which every monotone circuit of depth $q \log^{i-1} n$ requires size $2^{\Omega(n^{1/(10+4\epsilon)q})}$.*

B.1.5 Discussion of Techniques

Let us now briefly discuss the techniques we use to establish the above results, focusing for concreteness on Theorems B.1.1 and B.1.2. These theorems are proven by a careful chain of reductions as follows.

1. Our first step is to use the connection made explicit in [HN12], and also used in [GP18], that short and space-efficient proofs for a CNF formula F can be converted to efficient communication protocols for the *falsified clause search problem* for F. Going beyond [HN12, GP18], however, we make the simple but absolutely crucial additional observation that protocols obtained in this way are also round-efficient. Furthermore, in contrast to [HN12, GP18] we do not study randomized communication, but instead focus on the *real communication model* introduced by Krajíček [Kra98] with the purpose of getting a tighter correspondence with cutting planes.

2. We next generalize the communication-to-decision-tree simulation theorem for *composed search problem* in the celebrated paper by Göös et al. [GPW15] to the real communication model, and then extend it further to be able to handle rounds using the *parallel decision trees* introduced by Valiant [Val75b]. This part is inspired by [BEGJ00], where the simulation theorem in the precursor [RM99] of [GPW15] was proven for real communication but without taking round efficiency into account.

3. To leverage this machinery we need a base CNF search problem, and just as in [BN11, GP18, HN12, BNT13] (and many other papers) the *pebbling formulas* Peb_G from [BW01] turn out to be handy here, provided that they are defined over appropriately chosen directed acyclic graphs G. These formulas are then *lifted* (corresponding to composition of search problems) as described in [BHP10], though the parameters of the lifting are different (and unfortunately significantly worse than in [HN12]).

4. The following step is the relatively straightforward observation that efficient parallel decision trees for formulas Peb_G yield good strategies in the pebble game of Dymond and Tompa [DT85] played on the underlying graph G. At the same time, this is a somewhat unexpected twist, since in previous papers such as [BN08, BN11, BNT13] size and space lower bounds for pebbling formulas always followed from the *black-white pebble game* [CS76] on G, but we cannot make use of that latter game here.

5. Since we have to use the Dymond–Tompa game rather than the black-white pebble game, as a consequence we also have to use different graphs than in [BN11, HN12, BNT13]—in particular, modifying the construction of graphs with good black-white pebbling trade-offs in [LT82]—and as a concluding step we prove Dymond–Tompa trade-offs for these graphs.

Putting all these pieces together, we obtain a general theorem saying that graphs with Dymond–Tompa trade-offs yield explicit 6-CNF formulas with size-space trade-offs for cutting planes (and polynomial calculus and resolution). Theorem B.1.4 follows by a similar chain of reductions.

B.1.6 Paper Outline

The rest of this paper is organized as follows. In Section B.2 we give a more detailed overview of the steps in the proofs of our main theorems, introducing formal definitions of the concepts discussed above as need arises. In Section B.3 we translate proofs into communication protocols. The heart of the paper is then in Section B.4, where we establish that communication protocols for lifted search problems can be simulated by decision trees for the original search problems. In Section B.5 we show how decision trees for our search problem for pebbling formulas can be converted to Dymond–Tompa game strategies for the corresponding graphs, and in Section B.6 we show Dymond–Tompa trade-offs. After having established the upper bounds needed for our proof complexity trade-offs in Section B.7, we put all the pieces together in Section B.8. Section B.9 then discusses how we can use the same tools to obtain circuit complexity separations. Finally, we make some concluding remarks in Section B.10.

B.2 Preliminaries and Proof Overview

In this section, we describe which components are needed for our results stated in Section B.1 and how they fit together. Our goal is to give an accessible high-level outline of the proofs, but still make clear what are the main technical points in the arguments and also indicate some of the challenges that have to be overcome.

Let us start by reviewing the concepts we need from proof complexity. Throughout this paper all logarithms are to base 2 unless otherwise specified, and we write $[n]$ to denote the set $\{1, 2, \dots, n\}$.

B.2.1 Proof Complexity Basics and Cutting Planes

For x a Boolean variable, a *literal over x* is either the variable x itself or its negation, denoted $\overline{x}$. It will also be convenient to use the notation $x^1 = x$ and $x^0 = \overline{x}$. A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals and a *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We will think of clauses and CNF formulas as sets, so that the ordering is inconsequential and there are no repetitions. A *k-CNF formula* is a CNF formula consisting of clauses containing at most k literals.

We write α, β to denote truth value assignments, i.e., functions to $\{0,1\}$, where we identify 0 with false and 1 with true (thus, x^b is the literal satisfied by setting $x = b$). We have the usual semantics that a clause is true under α, or *satisfied* by α, if at least one literal in it is true, and a CNF formula is satisfied if all clauses in it are satisfied. We write $\perp$ to denote the empty clause without literals, which is false under all truth value assignments.

Following [ABRW02, ET01], we view a proof of unsatisifiability of a CNF formula F, or *refutation* of F, as a non-deterministic computation, with a special read-only input tape from which the clauses of the formula F being refuted (which we refer to as *axioms*) can be downloaded and a working memory where all derivation steps are made. In a *cutting planes (CP)* derivation, memory configurations are sets of linear inequalities $\sum_j a_j x_j \geq c$ with $a_j, c \in \mathbb{Z}$. We translate clauses C to linear inequalities $L(C)$ by identifying the clause $\bigvee_j x_j^{b_j}$ with the inequality $\sum_j (-1)^{1-b_j} x_j \geq 1 - \sum_j (1 - b_j)$. A CP refutation of F is a sequence of configurations $(\mathbb{L}_0, \ldots, \mathbb{L}_\tau)$ such that $\mathbb{L}_0 = \emptyset$, the inequality $0 \geq 1$ occurs in $\mathbb{L}_\tau$, and for $t \in [\tau]$ we obtain $\mathbb{L}_t$ from $\mathbb{L}_{t-1}$ by one of the following rules:

Axiom download $\mathbb{L}_t = \mathbb{L}_{t-1} \cup \{L\}$ for L being either the encoding $L(C)$ of an *axiom clause* $C \in F$ or a *variable axiom* $x_j \geq 0$ or $-x_j \geq -1$ for any variable x_j.

Inference $\mathbb{L}_t = \mathbb{L}_{t-1} \cup \{L\}$ for L inferred by *addition* $\dfrac{\sum_j a_j x_j \geq c \qquad \sum_j b_j x_j \geq d}{\sum_j (a_j + b_j) x_j \geq c + d}$,

multiplication $\dfrac{\sum_j a_j x_j \geq c}{\sum_j k a_j x_j \geq kc}$, or *division* $\dfrac{\sum_j k a_j x_j \geq c}{\sum_j a_j x_j \geq \lceil c/k \rceil}$ for $k \in \mathbb{N}^+$.

Erasure $\mathbb{L}_t = \mathbb{L}_{t-1} \setminus \{L\}$ for some $L \in \mathbb{L}_{t-1}$.

The *length* of a CP refutation is the number of linear inequalities L appearing in download and inference steps, counted with repetitions. We obtain the *size* of a refutation by also summing the sizes of the coefficients and constant terms in the inequalities, i.e., each inequality $\sum_j a_j x_j \geq c$ contributes $\log|c| + \sum_j \log|a_j|$. The *formula space* of a configuration $\mathbb{L} = \{\sum_j a_{i,j} x_{i,j} \geq c_i \mid i \in [s]\}$ is the number of inequalities s in it, and the *total space* of $\mathbb{L}$ is $\sum_{i \in [s]} \bigl(\log|c_i| + \sum_j \log|a_{i,j}|\bigr)$. We obtain the formula space or total space of a refutation by taking the maximum over all configurations in it. Finally, the length, size, formula space, and total space of refuting a formula F is obtained by taking the minimum over all CP refutations of the formula with respect to the corresponding complexity measure.

B.2.2 Composed Search Problems and Lifted CNF Formulas

Informally speaking, the idea behind *lifting*, or *composition*, is to take a relation over some domain and extend it to tuples from the same domain by combining it with a *selector* function that determines on which coordinates from the tuples the relation should be evaluated.

Let S be any relation on the Cartesian product $A \times Q$. We will think of S as a *search problem* with input domain A and output range Q, where on any input $a \in A$ the task is to find some $q \in Q$ such that $(a, q) \in S$ (assuming that S is such that there always exists at least one solution). Throughout this paper, we will have $A = \{0, 1\}^m$ for some $m \in \mathbb{N}^+$, so for simplicity we fix A to be such a domain from now on.

For any $\ell \in \mathbb{N}^+$, we define the *lift of length ℓ of S* to be a new search problem $\mathit{Lift}_\ell(S) \subseteq \left([\ell]^m \times \{0, 1\}^{m \cdot \ell}\right) \times Q$ with input domain $[\ell]^m \times \{0, 1\}^{m \cdot \ell}$ and output range Q such that for any $x \in [\ell]^m$, any bit-vector $\{y_{i,j}\}_{i \in [m], j \in [\ell]}$, and any $q \in Q$, it holds that

$$(x, y, q) \in \mathit{Lift}_\ell(S) \quad \text{if and only if} \quad \left((y_{1,x_1}, y_{2,x_2}, \ldots, y_{m,x_m}), q\right) \in S \ . \tag{B.1}$$

In what follows, we will refer to the coordinates of the x-vector as *selector variables* and those of the y-vector as *main variables*, and we will sometimes use the notation

$$\mathsf{select}(x_i, y_i) = y_{i,x_i} \tag{B.2}$$

to denote the bit in y_i selected by x_i. We extend this notation to vectors to write $\mathsf{select}(x, y) = y_x = (y_{1,x_1}, \ldots, y_{m,x_m})$.

As in [HN12, GP18], we obtain our results by studying lifted search problems defined in terms of CNF formulas and proving communication lower bounds for such problems. Syntactically speaking, however, these objects are not themselves CNF formulas, which is what we use to feed to our proof system. Therefore, we need an additional step which translates the lifted search problems back to CNF as follows.

Definition B.2.1 (Lifted formula [BHP10]). Given $\ell \in \mathbb{N}^+$ and a CNF formula F over variables $u_1, \ldots, u_n$, the *lift of length ℓ of F*, denoted $\mathit{Lift}_\ell(F)$, is the formula over variables $\{x_{i,j}\}_{i \in [n], j \in [\ell]}$ (*selector variables*) and $\{y_{i,j}\}_{i \in [n], j \in [\ell]}$ (*main variables*) containing the following clauses:

- For every $i \in [n]$, an *auxiliary clause*

$$x_{i,1} \lor x_{i,2} \lor \cdots \lor x_{i,\ell} \ . \tag{B.3a}$$

- For every clause $C \in F$, where $C = u_{i_1} \lor \cdots \lor u_{i_s} \lor \overline{u}_{i_{s+1}} \lor \cdots \lor \overline{u}_{i_t}$ for some $i_1, \ldots, i_t \in [n]$, and for every tuple $(j_1, \ldots, j_t) \in [\ell]^t$, a *main clause*

$$\overline{x}_{i_1,j_1} \lor y_{i_1,j_1} \lor \cdots \lor \overline{x}_{i_s,j_s} \lor y_{i_s,j_s} \lor \overline{x}_{i_{s+1},j_{s+1}} \lor \overline{y}_{i_{s+1},j_{s+1}} \lor \cdots \lor \overline{x}_{i_t,j_t} \lor \overline{y}_{i_t,j_t} \ . \tag{B.3b}$$

Intuitively, we can think of the selector variables as encoding the vector $x \in [\ell]^m$ in the lifted search problem (B.1). Since $\overline{x}_{i,j} \vee y_{i,j}$ is equivalent to the implication $x_{i,j} \rightarrow y_{i,j}$, we can rewrite (B.3b) as

$$(x_{i_1,j_1} \rightarrow y_{i_1,j_1}) \vee \cdots \vee (x_{i_t,j_t} \rightarrow \overline{y}_{i_t,j_t}) \ , \tag{B.4}$$

from which we can see that for every clause C the auxiliary clauses encode that there is at least one choice of selector variables $x_{i,j}$ which are all true, and for this choice of selector variables the $y_{i,j}$-variables in the lifted main clause will play the role of the u_i-variables, giving us back the original clause C. It is easily verified that F is unsatisfiable if and only if $H = \mathit{Lift}_\ell(F)$ is unsatisfiable, and that if F is a k-CNF formula with m clauses, then H is a $\max(2k, \ell)$-CNF formula with at most $m\ell^k + n$ clauses. A small technical issue for us compared to [HN12, GP18] is that $\ell \gg k$ will not be constant, but we can convert the wide clauses in (B.3a) to constant width using extension variables, and so we will just ignore this issue in our proof overview.

B.2.3 Pebbling Contradictions

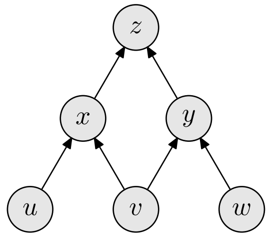
An important role in many proof complexity trade-off results is played by so-called *pebbling contradictions*. For our purposes it suffices to say that they are defined in terms of directed acyclic graphs (DAGs) G, where for simplicity we assume that all vertices have indegree 0 or 2. We refer to vertices with indegree 0 as *sources* and assume that there is a unique *sink* vertex with outdegree 0. What the pebbling contradiction over G says is that the sources are true and that truth propagates from predecessors to successors, but that the sink is false. The formal definition follows.

Definition B.2.2 (Pebbling contradiction [BW01]). Let G be a DAG with sources S and a unique sink z, and with all non-sources having indegree 2. Then the *pebbling contradiction* over G, denoted Peb_G, is the conjunction of the following clauses over variables $\{v \mid v \in V(G)\}$:

- for every source vertex $s \in S$, a unit clause s (*source axioms*),
- For all non-sources w with immediate predecessors u, v, a clause $\overline{u} \vee \overline{v} \vee w$ (*pebbling axioms*),
- for the sink z, the unit clause $\overline{z}$ (*sink axiom*).

If G has n vertices, the formula Peb_G is an unsatisfiable 3-CNF formula with $n+1$ clauses over n variables. For an example of a pebbling contradiction, see the CNF formula in Figure B.2b defined in terms of the graph in Figure B.2a.

To make the connection back to Definition B.2.1, in Figure B.3 we present the lift of length 2 of the CNF formula in Figure B.2b, with the auxiliary clauses at the top of the left column followed by the main clauses one by one, listed for all tuples of selector indices (with the only difference that since the variables in this formula are u, v, w, x, y, z



$$
\begin{aligned}
&u \\
\wedge\ &v \\
\wedge\ &w \\
\wedge\ &(\overline{u} \vee \overline{v} \vee x) \\
\wedge\ &(\overline{v} \vee \overline{w} \vee y) \\
\wedge\ &(\overline{x} \vee \overline{y} \vee z) \\
\wedge\ &\overline{z}
\end{aligned}
$$

(a) Pyramid graph Π_2 of height 2.

(b) Pebbling contradiction Peb_{Π_2}.

Figure B.2: Example pebbling contradiction for the pyramid of height 2.

rather than $u_1, \ldots, u_n$, we denote the main variables by $y_{u,j_1}, y_{v,j_2}, y_{w,j_3}$, et cetera, rather than $y_{i_1,j_1}, y_{i_2,j_2}, \ldots$). We will refer to the main clauses in Figure B.2b as source axioms, pebbling axioms, and sink axioms, respectively, when they have been obtained by lifting of the correspondingly named axioms in the pebbling contradiction.

B.2.4 Real Communication and Falsified Clause Search Problems

For our communication complexity results we study a two-player communication model, referring to the players as *Alice* and *Bob* following tradition. We first briefly discuss the basic deterministic model, and then explain how we need to extend it, directing the reader to [KN97] for any omitted standard communication complexity facts.

In the communication problem of computing a function $f : X \times Y \to Q$, Alice is given an input $x \in X$, Bob is given an input $y \in Y$, and they are required to find $f(x, y)$ while minimizing the communication between them. A communication protocol is a binary tree where Alice and Bob start at the root, every node specifies who is going to speak, the value of the spoken bit is only a function of the node v and the input x if Alice speaks or y if Bob does, and leaves are labelled by correct values $f(x, y)$. Similarly, for any relation $S \subseteq X \times Y \times Q$, the communication problem for S is one in which Alice is given $x \in X$, Bob is given $y \in Y$, and they are required to communicate to find some q such that $(x, y, q) \in S$. The cost of a protocol is the maximum number of bits communicated on any input, and the number of rounds is the maximum number of alternations between Alice and Bob speaking.

In order to obtain trade-offs for cutting planes, we need to study the more general *real communication* model in [Kra98], where Alice and Bob interact via a referee, and also introduce the concept of rounds in this model. It is convenient to describe the protocol as a (non-binary) tree, where at node v in the protocol tree Alice and Bob send k_v real numbers $\phi_{v,1}(x), \ldots, \phi_{v,k_v}(x)$ and $\psi_{v,1}(y), \ldots, \psi_{v,k_v}(y)$, respectively, to

$$
\begin{aligned}
& (x_{u,1} \lor x_{u,2}) \\
\land\, & (x_{v,1} \lor x_{v,2}) \\
\land\, & (x_{w,1} \lor x_{w,2}) \\
\land\, & (x_{x,1} \lor x_{x,2}) \\
\land\, & (x_{y,1} \lor x_{y,2}) \\
\land\, & (x_{z,1} \lor x_{z,2}) \\
\land\, & (\overline{x}_{u,1} \lor y_{u,1}) \\
\land\, & (\overline{x}_{u,2} \lor y_{u,2}) \\
\land\, & (\overline{x}_{v,1} \lor y_{v,1}) \\
\land\, & (\overline{x}_{v,2} \lor y_{v,2}) \\
\land\, & (\overline{x}_{w,1} \lor y_{w,1}) \\
\land\, & (\overline{x}_{w,2} \lor y_{w,2}) \\
\land\, & (\overline{x}_{u,1} \lor \overline{y}_{u,1} \lor \overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{x,1} \lor y_{x,1}) \\
\land\, & (\overline{x}_{u,1} \lor \overline{y}_{u,1} \lor \overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{x,2} \lor y_{x,2}) \\
\land\, & (\overline{x}_{u,1} \lor \overline{y}_{u,1} \lor \overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{x,1} \lor y_{x,1}) \\
\land\, & (\overline{x}_{u,1} \lor \overline{y}_{u,1} \lor \overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{x,2} \lor y_{x,2}) \\
\land\, & (\overline{x}_{u,2} \lor \overline{y}_{u,2} \lor \overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{x,1} \lor y_{x,1}) \\
\land\, & (\overline{x}_{u,2} \lor \overline{y}_{u,2} \lor \overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{x,2} \lor y_{x,2}) \\
\land\, & (\overline{x}_{u,2} \lor \overline{y}_{u,2} \lor \overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{x,1} \lor y_{x,1}) \\
\land\, & (\overline{x}_{u,2} \lor \overline{y}_{u,2} \lor \overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{x,2} \lor y_{x,2})
\end{aligned}
$$

$$
\begin{aligned}
\land\, & (\overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{w,1} \lor \overline{y}_{w,1} \lor \overline{x}_{y,1} \lor y_{y,1}) \\
\land\, & (\overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{w,1} \lor \overline{y}_{w,1} \lor \overline{x}_{y,2} \lor y_{y,2}) \\
\land\, & (\overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{w,2} \lor \overline{y}_{w,2} \lor \overline{x}_{y,1} \lor y_{y,1}) \\
\land\, & (\overline{x}_{v,1} \lor \overline{y}_{v,1} \lor \overline{x}_{w,2} \lor \overline{y}_{w,2} \lor \overline{x}_{y,2} \lor y_{y,2}) \\
\land\, & (\overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{w,1} \lor \overline{y}_{w,1} \lor \overline{x}_{y,1} \lor y_{y,1}) \\
\land\, & (\overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{w,1} \lor \overline{y}_{w,1} \lor \overline{x}_{y,2} \lor y_{y,2}) \\
\land\, & (\overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{w,2} \lor \overline{y}_{w,2} \lor \overline{x}_{y,1} \lor y_{y,1}) \\
\land\, & (\overline{x}_{v,2} \lor \overline{y}_{v,2} \lor \overline{x}_{w,2} \lor \overline{y}_{w,2} \lor \overline{x}_{y,2} \lor y_{y,2}) \\
\land\, & (\overline{x}_{x,1} \lor \overline{y}_{x,1} \lor \overline{x}_{y,1} \lor \overline{y}_{y,1} \lor \overline{x}_{z,1} \lor y_{z,1}) \\
\land\, & (\overline{x}_{x,1} \lor \overline{y}_{x,1} \lor \overline{x}_{y,1} \lor \overline{y}_{y,1} \lor \overline{x}_{z,2} \lor y_{z,2}) \\
\land\, & (\overline{x}_{x,1} \lor \overline{y}_{x,1} \lor \overline{x}_{y,2} \lor \overline{y}_{y,2} \lor \overline{x}_{z,1} \lor y_{z,1}) \\
\land\, & (\overline{x}_{x,1} \lor \overline{y}_{x,1} \lor \overline{x}_{y,2} \lor \overline{y}_{y,2} \lor \overline{x}_{z,2} \lor y_{z,2}) \\
\land\, & (\overline{x}_{x,2} \lor \overline{y}_{x,2} \lor \overline{x}_{y,1} \lor \overline{y}_{y,1} \lor \overline{x}_{z,1} \lor y_{z,1}) \\
\land\, & (\overline{x}_{x,2} \lor \overline{y}_{x,2} \lor \overline{x}_{y,1} \lor \overline{y}_{y,1} \lor \overline{x}_{z,2} \lor y_{z,2}) \\
\land\, & (\overline{x}_{x,2} \lor \overline{y}_{x,2} \lor \overline{x}_{y,2} \lor \overline{y}_{y,2} \lor \overline{x}_{z,1} \lor y_{z,1}) \\
\land\, & (\overline{x}_{x,2} \lor \overline{y}_{x,2} \lor \overline{x}_{y,2} \lor \overline{y}_{y,2} \lor \overline{x}_{z,2} \lor y_{z,2}) \\
\land\, & (\overline{x}_{z,1} \lor \overline{y}_{z,1}) \\
\land\, & (\overline{x}_{z,2} \lor \overline{y}_{z,2})
\end{aligned}
$$

Figure B.3: Lifted formula $\mathit{Lift}_2\big(\mathit{Peb}_{\Pi_2}\big)$ of length 2 obtained from the pebbling contradiction over Π_2.

the referee. The referee announces the results of the comparisons $\phi_{v,i}(x) \leq \psi_{v,i}(y)$ for $i \in [k_v]$ as a k_v-bit binary string, after which the players move to the corresponding next node in the protocol tree. The number of rounds r of a protocol is the depth of the tree and the cost c is the total number of comparisons made by the referee for any input. It is easy to see that this model can simulate standard deterministic communication (for instance, if Alice wants to send a message, she sends the complement of that message to the referee and Bob sends a list of the same length with all entries $1/2$) and is in fact strictly stronger (since equality can be solved with just two bits of communication).

The communication problem that we are interested in is the *(falsified) clause search problem*. This is the problem of, given an unsatisfiable CNF formula F and a truth value assignment α, finding a clause $C \in F$ falsified by α. We denote this problem by *Search*(F). In fact, from a communication complexity point of view we will be interested in lifts of this search problem *Lift*(*Search*(F)), while for our proof complexity trade-offs the perspective is slightly different in that we need to study the CNF formula *Lift*(F) from Definition B.2.1 and relate the hardness of this formula to the communication complexity of the falsified clause search problem *Search*($Lift(F)$). Happily, this distinction does not really matter to us, since a good communication protocol for *Search*($Lift(F)$) can also be used to solve *Lift*($Search(F)$), and hence a lower bound for the latter communication problem applies also to the former, as stated formally in the next observation.

Observation B.2.3. *Suppose that F is an unsatisfiable CNF formula. Then any two-player real communication protocol for $Search(Lift_\ell(F))$ where all selector variables $x_{i,j}$ in the same block are given to the same player can be adapted to a protocol for $Lift_\ell(Search(F))$ with the same parameters.*

We refer to, e.g., [HN12] for the easy proof (which is independent of the concrete communication model under consideration). Thanks to this observation, we can freely switch perspectives between $Lift_\ell(Search(F))$ and $Search(Lift_\ell(F))$ when we want to prove lower bounds for the latter problem. The reason that such lower bounds are interesting, in turn, is that if a CNF formula H has a CP refutation in short length and small space, then such a proof can be used to construct a round- and communication-efficient protocol for $Search(H)$.

Lemma B.2.4. *If a CNF formula H can be refuted by cutting planes in length L and formula space s, then for any partition of the variables of H between Alice and Bob there is a real communication protocol solving $Search(H)$ in $\lceil \log L \rceil$ rounds with total communication cost at most $s \cdot \lceil \log L \rceil$.*

Sketching the proof very briefly, given a truth value assignment α Alice and Bob can do binary search over the refutation $(\mathbb{L}_0 = \emptyset, \mathbb{L}_1, \ldots, \mathbb{L}_L)$ of H until they find a $t \in [L]$ such that $\mathbb{L}_t$ evaluates to true under α but $\mathbb{L}_{t-1}$ evaluates to false. Then the derivation step at time t must be a download of an axiom $C \in H$ falsified by α. For the details we can reuse the proof from [HN12] verbatim, just adding the one simple but absolutely crucial observation that the protocol obtained in this way is also round-efficient, since all communication needed to evaluate a particular configuration $\mathbb{L}_t$ can be performed in parallel.

It is worth noting that although we state Lemma B.2.4 for cutting planes here, there is nothing that really uses the syntactic properties of the cutting planes refutation. Thus, the proof works equally well for resolution, polynomial calculus, or any proof system for which configurations can be evaluated by round-efficient protocols where the communication scales as the space of the configuration.

B.2.5 Simulations of Protocols by Parallel Decision Trees

A *parallel decision tree* [Val75b] for a search problem $S \subseteq \{0,1\}^m \times Q$ is a tree T such that each node v is labelled by a set of variables V_v and has exactly one outgoing edge for each of the $2^{|V_v|}$ possible assignments to these variables, and such that for every $\alpha \in \{0,1\}^m$ the path from the root of T defined by the edges consistent with α ends at a leaf labelled by some $q \in Q$ such that $(\alpha, q) \in S$ (where again the tacit assumption is that S is such that such a solution always exists). The *number of queries* of T is the maximal sum of set sizes $|V_v|$ along any path in T, and the *depth* of T is the length of a longest path.

Any decision tree T for a search problem S can be simulated by a communication protocol for the lifted problem *Lift*(S) in a straightforward way, where if T wants to query the ith variable Alice and Bob can communicate to find y_{i,x_i} and then walk in T according to this value. Such a walk will end in a leaf labelled by a q such that $\big((y_{1,x_1}, y_{2,x_2}, \dots, y_{m,x_m}), q\big) \in S$, i.e., a solution to the lifted search problem, and thus the query complexity of the original search problem provides an upper bound on the communication cost of the lifted problem. If in addition there is a parallel decision tree with small depth, then a protocol simulating such a tree will also be round-efficient. The main technical result of our paper is that simulating such a parallel decision tree is essentially the best any round-efficient protocol can do (provided that the lifting of the search problem is done with appropriate parameters).

Theorem B.2.5 (Simulation theorem). *Let S be a relation with domain $\{0,1\}^m$ and let $\ell = m^{3+\epsilon}$ for some constant $\epsilon > 0$. If there is an r-round real communication protocol in cost c that solves Lift$_\ell(S)$, then there is a parallel decision tree in depth r solving S using $\mathrm{O}(c/\log\ell)$ queries.*

We remark that similar simulation theorems have previously been shown for both deterministic communication [RM99, GPW15] and real communication [BEGJ00], but unfortunately they fail to take round efficiency into account. Our proof of Theorem B.2.5 follows the approach in these papers to build a decision tree for the original problem that simulates the communication protocol for the lifted problem. In order to obtain an efficient simulation we have to maintain (in an amortized sense) that the decision tree queries a variable only when a noticeable amount of communication has taken place. To prove that the decision tree constructed in this way is correct, we need to show that at the end of the simulation there exists a pair of inputs to Alice and Bob that are compatible both with the transcript and with a lift of the original input. Towards this end, during the simulation we maintain a set of such compatible inputs, which must not be allowed to shrink too fast.

In order for the proof to work we need to be able to handle two kinds of events: *communication events*, where we simulate the players communicating; and *query events*, where the decision tree under construction queries some variable and gets its actual value. Both of these events force us to prune the set of compatible communication inputs. In the first case we want to choose a communication message that removes as few inputs as possible, whereas in the second case we have to restrict the communication inputs to a subset that is compatible with the value returned by the decision tree query. We make sure to query a variable only when the transcript "reveals too much information" about Alice's and Bob's lifted input related to that variable, and thanks to this we can argue that query events do not happen too often and that the amount of communication provides an upper bound on the total number of queries.

Extending these techniques to round-efficient protocols and simulations by parallel decision trees causes significant additional complications, however. Very briefly, one issue is that we cannot let the tree query an individual variable as soon as sufficient information has been "revealed" about it during the simulation, but have to wait until we can issue a whole set of queries corresponding to a single message of the protocol. This makes it challenging to maintain a set of compatible inputs for variables we have not yet been allowed to query. Another issue is that, in contrast to deterministic communication protocols, real protocols do not partition the input domain into combinatorial rectangles. While this is not a big problem for a single comparison by the referee, it becomes more challenging when we want to handle a round consisting of many simultaneous comparisons.

B.2.6 From Decision Trees to Dymond–Tompa Trade-offs

The *Dymond–Tompa game* [DT85][2] is played in rounds on a DAG G by two players *Pebbler* and *Challenger*. In the first round, Pebbler places pebbles on a non-empty subset of vertices of G including the unique sink z and Challenger picks some vertex in this set. In all subsequent rounds, Pebbler places pebbles on some non-empty subset of vertices not yet containing pebbles, and Challenger either challenges a vertex in this new set (*jumps*) or re-challenges the previously chosen vertex (*stays*). This repeats until at the end of a round Challenger is standing on a vertex with all immediate predecessors pebbled (or on a source, for which the condition vacuously holds), at which point the game ends. Intuitively, Challenger is challenging Pebbler to "catch me if you can" and wants to play for as many rounds as possible, whereas Pebbler wants to "surround" Challenger as quickly as possible. We say that Pebbler *wins the r-round Dymond–Tompa game on G in cost c* if there is a strategy such that Pebbler can always finish the game in at most r rounds placing a total of at most c pebbles regardless of how Challenger plays.

In order to obtain lower bounds on the query complexity of parallel decision trees of bounded depth, we use an adversary argument and describe strategies that give as unhelpful answers as possible for variables queried by the decision trees. If we specialize this to the clause search problem for pebbling contradictions Peb_G, such adversary strategies are equivalent to Challenger strategies in the Dymond–Tompa game on G. For standard binary decision trees and the Dymond–Tompa game with unlimited number of rounds this was proven in [Cha13],[3] and we show that this equivalence extends also to our more general setting where decision trees can issue queries in parallel and we account for the number of rounds in the Dymond–Tompa game.

[2]We give a slightly different, but essentially equivalent, description of the Dymond–Tompa game that is closer to recent papers such as [Cha13, CLNV15].

[3]This game on decision trees is called the *Raz–McKenzie game* in [Cha13].

Lemma B.2.6. *If there is a parallel decision tree for $\mathit{Search}(\mathit{Peb}_G)$ in depth r using at most c queries, then Pebbler has a winning strategy in the r-round Dymond–Tompa game on G in cost at most $c + 1$.*

It follows from this lemma that round-cost trade-offs for Dymond–Tompa pebbling implies depth-query trade-offs for parallel decision trees. To conclude the proof of the lower bound in our trade-off results, we need to find a family of graphs for which we can prove lower bounds for the cost of Pebbler strategies in the Dymond–Tompa game with bounded number of rounds. Towards this end, we establish that graphs that satisfy a certain connectivity property possess trade-offs between number of rounds and cost, and then exhibit such graphs. These graphs were inspired by graphs for which black-white pebbling trade-offs were shown in [LT82], but we need to make some modifications in order to obtain Dymond–Tompa trade-offs. Round-cost trade-offs have been previously studied in [KS90] but for a different range of parameters.

Lemma B.2.7. *For any $n, r \in \mathbb{N}^+$ there exists an explicitly constructible DAG $G(n, r)$ with $\mathrm{O}(rn \log n)$ vertices such that the cost of the r-round Dymond–Tompa game on $G(n, r)$ is at least $\Omega(n)$.*

The graph $G(n, r)$ is structured in $r + 1$ layers and we obtain the lemma by designing a strategy for Challenger such that as long as Pebbler does not place too many pebbles, Challenger can make sure that in the ith round the challenged pebble is above the ith layer. Hence, the game does not end within r rounds.

B.2.7 Proofs of Main Theorems

Combining all the components discussed above we can now prove the following trade-off lower bound.

Theorem B.2.8. *Let G be a DAG over m vertices such that any winning strategy for Pebbler in the r-round Dymond–Tompa game on G has cost $\Omega(c)$, and let $\epsilon > 0$ and $\ell = m^{3+\epsilon}$. Then $\mathit{Lift}_\ell(\mathit{Peb}_G)$ is a 6-CNF formula over $\Theta(m^{4+\epsilon})$ variables and $N = \Theta(m^{10+3\epsilon})$ clauses such that any cutting planes refutation of it in formula space less than $\frac{c}{r} \log N$, even with coefficients of unbounded size, requires length at least $2^{\Omega(r)}$.*

Proof. Suppose for the sake of contradiction that there is a cutting planes refutation of $\mathit{Lift}_\ell(\mathit{Peb}_G)$ in length $2^{\mathrm{o}(r)}$ and formula space less than $\frac{c}{r} \log N$. By Lemma B.2.4 this implies that there is a real communication protocol that solves $\mathit{Lift}_\ell(\mathit{Search}(\mathit{Peb}_G))$ in $\mathrm{o}(r)$ rounds and total cost $\mathrm{o}(c \log N)$. Using Theorem B.2.5 we obtain a parallel decision tree computing $\mathit{Search}(\mathit{Peb}_G)$ using $\mathrm{o}(c)$ queries and depth $\mathrm{o}(r)$. But if so, by Lemma B.2.6 Pebbler wins the $\mathrm{o}(r)$-round Dymond–Tompa game on G in cost $\mathrm{o}(c)$, which contradicts the assumption of the theorem. □

In order to attain our trade-off results we also need upper bounds on refutations of these formulas. Small-size upper bounds follow by essentially the same approach of lifting black pebbling upper bounds as in [BN11, HN12], although more care is needed since our lifts are of non-constant length. For the small-space refutations, this technique does not work because the space loss due to the large lift length is larger than the upper bound we are aiming for. Luckily, we can instead prove upper bounds in the Dymond–Tompa game with unlimited rounds and then convert them into refutations in small space. Theorems B.1.1 and B.1.2 then follow from Theorem B.2.8 applied to an appropriate family of graphs that exhibit Dymond–Tompa trade-offs as in Lemma B.2.7.

The tools we have developed also allow us to prove the monotone circuit separation in Theorem B.1.4. The function that witnesses the separation is inspired by the PYRAMID-GEN function of [RM99] adapted to the graphs in Lemma B.2.7. Then we translate the Dymond–Tompa trade-off into a lower bound for deterministic communication protocols with few rounds, which we then transform into a lower bound for circuits of small depth via the Karchmer–Wigderson game [KW90].

B.3 From Proofs to Communication Protocols

As mentioned in the preliminaries, length space trade-offs for a given proof system can be obtained via reduction to the falsified clause search problem. Exactly which communication model to consider for the search problem depends on the proof system. Given a sequential proof system $\mathcal{P}$ and a communication model $\mathcal{M}$, let $c_{\mathcal{P},\mathcal{M}}$ and $r_{\mathcal{P},\mathcal{M}}$ be the maximum cost and the maximum number of rounds, respectively, required to evaluate a line/formula of any configuration.

The idea behind the reduction is to, given a refutation as a sequence of configurations, do a binary search in this sequence in order to find two consecutive configurations such that the first is evaluated to true and the second to false. Since the proof system is sound, this false configuration must be due to an axiom download and this axiom must be falsified. Finally, observing that each line/formula of a configuration can be evaluated in parallel, we get the following lemma.

Lemma B.3.1. *Let π be a refutation in a sequential proof system $\mathcal{P}$ of a CNF formula F in length L and formula space s. Then, in any (deterministic) communication model $\mathcal{M}$ and for any partition of the variables of F between Alice and Bob there is a communication protocol that solves Search(F) in $r_{\mathcal{P},\mathcal{M}} \cdot \lceil \log L \rceil$ rounds with total communication cost at most $s \cdot c_{\mathcal{P},\mathcal{M}} \cdot \lceil \log L \rceil$.*

Proof. Suppose Alice and Bob are each given a part of an assignment α to F. Fix a $\mathcal{P}$-refutation $\pi = \{\mathbb{D}_0, \mathbb{D}_1, \ldots, \mathbb{D}_L\}$ of F as in the statement of the lemma. By definition of refutation, it must be the case that $\mathbb{D}_0 = \emptyset$ and $\perp \in \mathbb{D}_L$.

Alice and Bob consider the configuration $\mathbb{D}_{L/2}$ in the refutation at time $L/2$ and with joint efforts (involving some communication, which we will discuss shortly) evaluate the truth value of $\mathbb{D}_{L/2}$ under the assignment α. If $\mathbb{D}_{L/2}$ is true under α, they continue their search in the subderivation $\{\mathbb{D}_{L/2}, \mathbb{D}_1, \ldots, \mathbb{D}_L\}$. If $\mathbb{D}_{L/2}$ is false, the search continues in the first half of the refutation $\{\mathbb{D}_0, \mathbb{D}_1, \ldots, \mathbb{D}_{L/2}\}$ Note that $\mathbb{D}_0 = \emptyset$ is vacuously true under any assignment, and since $\bot \in \mathbb{D}_L$ this last configuration evaluates to false under any assignment. The binary search is carried out so as to maintain that the first configuration in the current subderivation under consideration evaluates to true and the last one evaluates to false under the given assignment α. Hence, after at most $\lceil \log L \rceil$ steps Alice and Bob find a $t \in [L]$ such that $\mathbb{D}_{t-1}$ is true under α but $\mathbb{D}_t$ is false. Since the proof system is sound, the derivation step to get from $\mathbb{D}_{t-1}$ to $\mathbb{D}_t$ cannot have been an inference or erasure, but must be a download of some axiom clause $C \in H$. This clause C must be false under α, and so Alice and Bob give C as their answer.

Now all that remains is to discuss how much communication is needed to evaluate a configuration in the refutation. Every line/formula in the configuration can be evaluated with cost at most $c_{\mathcal{P},\mathcal{M}}$ and in at most $r_{\mathcal{P},\mathcal{M}}$ rounds. Moreover, the $r_{\mathcal{P},\mathcal{M}}$ rounds to evaluate each line can be done in parallel by simply concatenating, at each round i, all the ith messages of the protocol for evaluating each line of the configuration. Since each configuration has at most s lines, it can be evaluated with cost at most $s \cdot c_{\mathcal{P},\mathcal{M}}$ and in at most $r_{\mathcal{P},\mathcal{M}}$ rounds. $\square$

We note that, for randomized communication models (which we do not use in this paper, but are used for example in [HN12, GP18]), the above theorem holds if $c_{P,M}$ and $r_{P,M}$ are defined to be the maximum cost and the maximum number of rounds, respectively, required to evaluate a line/formula of any configuration with high enough probability of success so that the union bound of the probability of error over all the evaluations of configurations is small enough.

Ideally, given a proof system P we want a communication model M such that $r_{P,M}$ and $c_{P,M}$ are constants, or at least a slow growing function. We only consider semantic versions of proof systems, where we specify the format of proof lines and use the fact that derivation rules, whichever they are, are semantically sound (as defined in [ABRW02]).

For example, if P is resolution, where lines are clauses of the form $\bigvee_j x_j^{b_j}$, then Alice and Bob can evaluate a line in two rounds and two bits of communication in the deterministic communication model.

If we consider polynomial calculus over any field $\mathbb{F}$, where lines are polynomials of the form $\sum_m \prod_j a_{m,j} x_j$ but the space measure is the number of monomials, Alice and Bob first check that the assignment is $\{0,1\}$-valued—and immediately output a falsified axiom otherwise—and then run the binary search protocol, where they can evaluate a monomial in two rounds and two bits of communication in the deterministic communication model.

For cutting planes with bounded coefficients, Alice and Bob can evaluate a line in two rounds and either $O(\log N)$ bits of communication in the deterministic communication model if the bound is a polynomial in the size of the formula or $O(\log L)$ bits if the bound is a polynomial in the size of the refutation.

For unrestricted cutting planes, Alice and Bob can evaluate a line in one round and one comparison in the real communication model.

The small but key difference from previous papers [HN12, GP18] is that we explicitly consider rounds. The theorem states that if there exists a refutation in small space and small length, then there is a communication protocol that solves the falsified clause search problem not only with small communication cost, but also in few rounds.

It seems unlikely that the techniques used so far (without rounds) would yield true trade-offs; let us discuss why. For a trade-off of the form $s \cdot \log L \geq x$ to be a true trade-off there must exist a refutation in small space and another one in small length. The formulas for which trade-offs have been proven are lifted versions of pebbling formulas, which have refutations in small (linear) length, but not necessarily small space: the black-white pebbling number is a lower bound for resolution space, as proven in [BN11].

For the pyramid graphs studied in [HN12, GP18], the black-white pebbling number is asymptotically equal to the Dymond–Tompa price, which in turn is an upper bound for the communication complexity, as we argued in Section B.2. Therefore, for the resolution proof system, the apparent trade-off is actually $s \cdot \log L = \Omega(s)$, giving only an uninformative length lower bound for the feasible range of space, and so the formula properties are better described by a space lower bound rather than a trade-off.

It seems plausible that the black-white pebbling number is also a lower bound for polynomial calculus space and cutting planes total space, and thus the "trade-offs" between PC-space or CP-total space and length, might also turn out to be unconditional space lower bounds.

Even if we consider other graph families, the best separation between black-white pebbling number and Dymond–Tompa price so far is logarithmic in the size of the graph [CLNV15], which still does not give meaningful results for resolution. It seems more promising to search for trade-offs in graphs where the black-white pebbling number is small but nonetheless have trade-offs in resolution, established by some means other than communication complexity.

To keep the discussion short and focused we only mention that trade-offs have been proven for stacks of superconcentrators [BN11] and Carlson–Savage graphs [Nor12]. Yet in both cases the Dymond–Tompa price is too small to give meaningful trade-offs: in the first case, it is enough to note that the Dymond–Tompa price is at most the depth, and a stack of superconcentrators has small depth; for Carlson–Savage graphs, the proof is similar, but the depth argument is not enough.

To sum up, we showed that the graphs for which the previous techniques yield trade-offs are likely to have unconditional space lower bounds (but we cannot prove it), and that for graph families that may have trade-offs—and indeed we prove that this is the case for a special family of superconcentrators—the previous techniques cannot prove them.

B.4 From Real Communication to Parallel Decision Trees

We have reached the heart of the reduction. This is by far the most intriguing and also the most difficult part of the paper. The reader that first wishes to have an overview of the whole proof should skip Sections B.4.1 and B.4.2.

To prove Theorem B.2.5 we use the same high-level approach of [RM99, BEGJ00, GPW15]: we build a decision tree that simulates a protocol computing the composed search problem $\mathit{Lift}(S)$ and it only queries a variable when, in a certain sense, the transcript reveals too much information about the index for that variable.

More precisely, we keep two sets of inputs A and B that are compatible with the communication so far and that are large enough. Additionally for A we enforce that for each coordinate i that we have not queried yet, if we fix every other coordinate to some value, there are still many choices for what to set the index x_i to.

To maintain the invariant, we need to handle two kinds of events: communication events where we guess a message and restrict A and B to the new compatible inputs, with some additional cleanup, and query events, where some coordinate i becomes too dependent on other coordinates. Since for each coordinate there are more choices for y_i than Bob can expect to communicate, we will be able to find an input for the players such that $\mathsf{select}(x_i, y_i)$ agrees with z_i, and then restrict A and B appropriately.

At the end of the protocol, if we were to query the remaining variables, we would have a pair of inputs (x, y) that are compatible with the transcript, therefore the protocol answers correctly, and such that $\mathsf{select}(x_i, y_i) = z_i$ in every coordinate, therefore the answer is also correct for the decision tree.

To argue that we do not make too many queries we keep a density function that measures how many choices we have for Alice's input over not queried coordinates. This function increases on communication, decreases after a query, and is nonnegative, which gives us a bound on the number of queries in terms of communication.

The description up to this point is common to all flavours of the simulation theorem, with or without rounds and with deterministic or real communication. The differences will surface once we try to implement it.

The first challenge we encounter is when exactly should we query a variable. If we do not have any bound on depth, then we can do that as soon as we detect that the invariant is broken and we need to restore it. Since we want to measure the effect of rounds, however, this exact approach will not work for us, because we might need to query a variable mid-message. Indeed, if Alice sends the message x_1x_2, we would first simulate sending some bits of x_1, then query z_1 and restrict the inputs to those such that $\mathsf{select}(x_1, y_1) = z_1$, keep simulating sending bits of x_1 and x_2, then query z_2 and restrict the inputs, and finish sending bits. We had to use two rounds of queries in a single round of communication.

It seems natural to delay querying the variables until the end of the message, but now we have another problem. Assume that Bob had sent that the 0-th bit of y_1 is a 0, and that Alice's message is x_1. If we guess that the message is $x_1 = 0$ but after we query z_1 we get that $z_1 = 1$, then the inputs compatible with the communication stop being compatible with the gadget select.

Our solution is to indeed delay querying the variables until the end of the message but still restrict the inputs as soon as the invariant breaks, in a way that, after fixing x_1, $\mathsf{select}(x_1, y_1)$ can take any value. During the interval of communication between the restriction and the query we keep a set C that acts as a proxy for B over the coordinates that we have not queried (and do not intend to). This is a harder task, but still feasible because only Alice speaks during this time, so we do not need to know B precisely.

The second challenge is to adapt the simulation to a real communication protocol. As opposed to deterministic protocols, the set of compatible inputs does not necessarily form a rectangle. However, as observed by [Joh98], the result of one comparison splits the matrix of inputs in such a way that one quadrant—thus a rectangle—is monochromatic, and [BEGJ00] uses this fact to decide the outcome of each comparison.

Since we want to query variables only at the end of each round we might not know what B is at the time we want to extract a rectangle from a comparison matrix, and unfortunately the proxy does not help. Therefore we need to extract rectangles from the input when we do know B, this is before we start simulating the message. We could easily extract a rectangle of size a $2^{-k} \times 2^{-k}$-fraction of the inputs, where k is the size of the message, but that would be equivalent to simulating the message in one go, which we argued does not work in the deterministic case.

Our solution is to extract a rectangle of size a $1/4 \times 2^{-k \log k}$-fraction of the inputs. Even though this is equivalent to simulating a long Bob message at once, it has the advantage that the equivalent message for Alice is very short, so we can still use the very same techniques to recover the invariants as in the deterministic case.

B.4.1 Simulation of Deterministic Communication Protocols by Decision Trees

We begin by proving the simulation theorem for the more common deterministic communication model. Throughout this section we assume that the arity of the select function is $\ell = m^{3+\epsilon}$ for some small constant $\epsilon > 0$, where m is the size of the input.

Theorem B.4.1. *If there is a deterministic communication protocol computing Lift(S) using communication c and r rounds, then there is a parallel decision tree computing S using* $\mathrm{O}(c/\log \ell)$ *queries and depth r.*

We mention in Section B.2 that we have to use a lift of polynomial length as opposed to constant length; this is needed for the simulation theorem to apply. In [GPW15] and [RM99] the lift is of size m^{20}, while in [BEGJ00] the lift is of size m^{14}, and a more careful analysis shows that $m^{4+\epsilon}$ is enough. Using a combinatorial approach to the analysis we can lower the lift length to $m^{3+\epsilon}$, but getting beyond this seems hard.

To be able to prove Theorem B.4.1 we need to introduce some notation. Let $\gamma = 1/(3+\epsilon)$, δ, λ, μ be numbers strictly between 0 and 1 such that the inequalities

$$\lambda - \gamma > \mu \tag{B.5a}$$

$$\mu + \delta - 1 > \gamma \tag{B.5b}$$

$$\gamma + \delta < 1 \tag{B.5c}$$

hold. Note that a solution exists iff $\gamma < 1/3$. For concreteness, we can think of $\gamma = 1/3 - 2\xi$, $\lambda = 1 - \xi$, $\mu = 2/3$, and $\delta = 2/3$ for some $\xi > 0$.

Let Π be an r-round deterministic communication protocol computing $\mathit{Lift}(S)$ in cost c. Let X and Y be inputs to Alice and Bob. Let X^v and Y^v be the inputs compatible with node v of the protocol tree. We are going to keep two sets $A \subseteq [\ell]^m$ and $B \subseteq \{0,1\}^{\ell m}$ of inputs that are compatible with the communication so far, this is $A \subseteq X^v$ and $B \subseteq Y^v$, but have been cleaned up.

We will often be interested in the coordinates that the decision tree has not yet queried, which we denote $I \subseteq [m]$. We denote a vector with coordinates in I by $x_I = \{x_i : i \in I\}$ and, as a reminder, we denote a set of such vectors by S_I. We denote the projection of a set to I coordinates by $\pi_I(S_J) = \{x_I \in \{0,1\}^I : x \in S_J \text{ for some } x_{J \setminus I} \in \{0,1\}^{J \setminus I}\}$.

In order to formalize the property of having little information about a coordinate, we define $\mathit{Graph}_i(A_I)$ as the bipartite graph where left vertices x_i are elements of $[\ell]$, right vertices $x_{I\setminus\{i\}}$ are elements of $[\ell]^{|I|-1}$, and there is an edge between two vertices if $x_I \in A_I$. Analogously, let $\mathit{Graph}_i(B_I)$ be the bipartite graph where left vertices y_i are elements of $\{0,1\}^\ell$, right vertices $y_{I\setminus\{i\}}$ are elements of $\{0,1\}^{\ell(|I|-1)}$, and there is an edge between two vertices if $y_I \in B_I$. Now let $\mathit{MinDeg}_i(A_I)$ and $\mathit{AvgDeg}_i(A_I)$ be the minimum and average right degree of $\mathit{Graph}_i(A_I)$, both taken over the set of vertices of positive degree. We consider that we do not know too much about a coordinate i if $\mathit{AvgDeg}_i(A_I) \geq \ell^\lambda$. Moreover, we say A_I is *thick* if $\mathit{MinDeg}_i(A_I) \geq \ell^\mu$ for all $i \in I$, a property we will make sure to keep throughout the simulation. Observe that, since $|A_I|$ is the number of edges in $\mathit{Graph}_i(A_I)$ and $|\pi_{I\setminus\{i\}}(A_I)|$ is the number of right vertices with positive degree, the definition of average degree is equivalent to

$$\mathit{AvgDeg}_i(A_I) = \frac{|A_I|}{|\pi_{I\setminus\{i\}}(A_I)|} , \tag{B.6}$$

which is more convenient to work with.

A useful observation is that minimum degree (and therefore thickness) is monotone with respect to projections.

Observation B.4.2 ([RM99]). *$\mathit{MinDeg}_j(\pi_{I\setminus\{i\}}(A)) \geq \mathit{MinDeg}_j(\pi_I(A))$ for all $j \in I \setminus \{i\}$.*

Proof. For each index $j \in I \setminus \{i\}$, if there is an edge between $x_{I\setminus\{j\}}$ and x_j in $\mathit{Graph}_j(A_I)$, then there is also an edge between $x_{I\setminus\{i,j\}}$ and x_j in $Graph_j(\pi_{I\setminus\{i\}}(A_I))$. Formally,

$$\begin{aligned} N_j(x_{I\setminus\{i,j\}}) &= \{x_j : x_{I\setminus\{i\}} \in \pi_{I\setminus\{i\}}(A)\} = \{x_j : \exists x_i \; x_I \in \pi_I(A)\} &\text{(B.7)}\\ &= \bigcup_{x_i}\{x_j : x_I \in \pi_I(A)\} = \bigcup_{x_i} N_j(x_{I\setminus\{j\}}) \ , &\text{(B.8)}\end{aligned}$$

so for any x_i with positive right degree we have

$$|N_j(x_{I\setminus\{i,j\}})| \geq |N_j(x_{I\setminus\{j\}})| \geq MinDeg_j(\pi_I(A)) \ . \qquad \square$$

Finally, we define two density measures for inputs over non-queried coordinates. For $A_I \subseteq [\ell]^{|I|}$, let $\alpha(A_I) = -\log\frac{|A_I|}{\ell^{|I|}} = |I|\log\ell - \log|A_I|$. Analogously, for $B_I \subseteq \{0,1\}^{\ell|I|}$, let $\beta(B_I) = -\log\frac{|B_I|}{2^{\ell|I|}} = |I|\ell - \log|B_I|$. We will make sure to keep these measures small, so that at any point there are many inputs compatible with the communication so far.

In fact, density even increases with projections.

Observation B.4.3 ([GPW15]). $\alpha(\pi_{I\setminus\{i\}}(A)) = \alpha(\pi_I(A)) - \log\ell + \log AvgDeg_i(\pi_I(A))$

Proof. By definition of α, this is equivalent to $|\pi_{I\setminus\{i\}}(A)| = |\pi_I(A)|/AvgDeg_i(\pi_I(A))$, which follows from the definition of average degree (B.6). $\square$

We have an auxiliary procedure prune, which we use to restore the thickness of A after Alice speaks, with the following properties.

Lemma B.4.4 (Thickness Lemma [RM99]). *If $AvgDeg_i(\pi_I(A)) \geq \ell^\lambda/4$ for all $i \in I$, then the return value A' of* prune(A, I) *satisfies*

1. *$\pi_I(A')$ is thick;*
2. *$\alpha(\pi_I(A')) \leq \alpha(\pi_I(A)) + 1$.*

To restrict the inputs on one coordinate i to a set $U \subseteq [\ell]$ we write $\rho_{i,U}(A) = \{x \in A : x_i \in U\}$, and similarly for $V \subseteq \{0,1\}^\ell$, we have $\rho_{i,V}(B) = \{y \in B : y_i \in V\}$. The set of b-monochromatic colourings of U is $V^b(U) = \{w \in \{0,1\}^\ell : \forall j \in U \; w_j = b\}$.

We have another auxiliary procedure `project`, which we use to pick the appropriate U after we make a query in order to recover the density of A with respect to the remaining coordinates, with the following properties. Note that in the description of Algorithm B.4 we employ sets C_I that act as a proxy for $\pi_I(B)$. We denote ${C_{I\setminus\{i\}}}^{(b)}(U) = \pi_{I\setminus\{i\}}(\rho_{i,V^b(U)}(C_I))$.

Lemma B.4.5 (Projection Lemma). *If $\pi_I(A)$ is thick, and $\beta(C_I) \leq 2\ell^\gamma\log^2\ell$, then the return value U of* `project`(A, C_I, I, i) *satisfies*

1. *$\pi_{I\setminus\{i\}}(\rho_{i,U}(A))$ is thick ;*
2. *$\alpha(\pi_{I\setminus\{i\}}(\rho_{i,U}(A))) \leq \alpha(\pi_I(A)) - \log\ell + \log AvgDeg_i(\pi_I(A))$;*

```
let $A = [\ell]^m$, $B = \{0,1\}^{\ell m}$, $I = [m]$, $v$ be the root of $\Pi$
while $v$ is not a leaf do
    let $Q = \emptyset$, $C_I = \pi_I(B)$
    while $v$ is a node where Alice speaks do
        while $\exists i \in I$ such that $AvgDeg_i(\pi_I(A)) < \ell^\lambda$ do
            let $U_i$ = project ($A$, $C_I$, $I$, $i$)
            let $A = \rho_{i,U_i}(A)$, $C_{I\setminus\{i\}} = {C_{I\setminus\{i\}}}^{(0)}(U_i) \cap {C_{I\setminus\{i\}}}^{(1)}(U_i)$
            let $I = I \setminus \{i\}$, $Q = Q \cup \{i\}$
        let $b = \mathrm{argmax}|\pi_I(A \cap X^{v_b})|$
        let $A = \mathtt{prune}(A \cap X^{v_b}, I)$, $v = v_b$
    query coordinates $Q$ to get string $z_Q$
    for $i \in Q$ do
        let $B = \rho_{i,V}(B)$, where $V = V^{z_i}(U_i)$
    while $v$ is a node where Bob speaks do
        let $b = \mathrm{argmax}|\pi_I(B \cap Y^{v_b})|$
        let $B = B \cap Y^{v_b}$, $v = v_b$
return the answer at $v$
```

Figure B.4: Procedure eval(Π,z)

3. $\beta({C_{I\setminus\{i\}}}^{(0)}(U) \cap {C_{I\setminus\{i\}}}^{(1)}(U)) \leq \beta(C_I) + 1$.

The decision tree that witnesses Theorem B.4.1 is Algorithm eval. The difference from the algorithm in [RM99, GPW15] is that, when we need to restrict the sets A and B, instead of making a query we consider both possible outcomes for Bob. We can delay committing to either outcome until the moment before Bob starts to speak, at which point we make all queries at once. We assume that Q is a queue, this is, its elements are sorted in insertion order. We also assume that argmax decides arbitrarily in case of tie, and observe that if $b = \mathrm{argmax}|\pi_I(A \cap X^{v_b})|$, then $\alpha(\pi_I(A \cap X^{v_b})) \leq \alpha(\pi_I(A)) + 1$.

Lemma B.4.6 (Main Lemma). *If Π is a protocol that computes $Lift(S)$ using communication $c < \frac{m}{2}(1-\lambda)\log \ell$ and r rounds, then* eval *computes S using at most $2c/(1-\lambda)\log \ell$ queries and depth r.*

Observe that Theorem B.4.1 follow from this lemma, since if $c \geq \frac{m}{2}(1-\lambda)\log \ell$ then a parallel decision tree that queries all variables in depth 1, satisfies the theorem. Before proving Lemma B.4.6, let us consider some possible protocols and what Algorithm B.4 does.

Consider the trivial protocol where Alice in one round sends all of her input to Bob, who outputs the answer. To be fair, this is a bit too much communication for Lemma B.4.6 to apply, but let us look over this fact and try to get an intuition of what happens. While Alice is speaking, the algorithm has to commit to the values of her coordinates, and therefore, for all coordinates i, $\mathit{AvgDeg}_i(\pi_I(A))$ eventually becomes too small and i is marked to be queried. After she finishes speaking the algorithm queries all coordinates and restricts Bob's input to be compatible with the queries and with Alice's message, i.e., it only keeps inputs that have the queried value on the corresponding position. Finally, the algorithm answers what Bob would output for any of the compatible inputs. Note that the decision tree in this case is the depth-1 decision tree that queries all coordinates at once and answers accordingly.

Another possible protocol is the one that follows a parallel decision tree T, as explained in Section B.2: Alice and Bob communicate to find the value of the coordinates queried on each node and then continue on T according to these values. For this protocol, Algorithm B.4 marks to be queried all the coordinates Alice and Bob talk about because the corresponding average degree gets too low. Note that the final decision tree is exactly the same as the one Alice and Bob were following.

Now we consider what happens in a more extreme case. Suppose the protocol is very unbalanced in the sense that Alice's first bit is a 0 if her first coordinate is 42, and otherwise is a 1. In this case, when at line 9 the algorithm chooses a bit for Alice to speak, it will always choose 1 since it is compatible with the most inputs.

Proof of Lemma B.4.6. Let R^v be the rectangle of inputs compatible with node v, and let c_v^A (resp. c_v^B) be the number of bits sent by Alice (resp. Bob) up to node v. Let χ be the number of queries so far, i.e., $\chi = m - |I|$. We show that the following invariants hold throughout the algorithm:

1. $\pi_I(A)$ is thick;
2. $A \times B \subseteq R^v$;
3. $\chi \leq 2c_v^A/(1-\lambda)\log \ell$;
4. $\beta(C_I) \leq \chi + c_v^B$;

and the following invariant holds at the beginning of each round:

5. $\beta(\pi_I(B)) \leq \chi + c_v^B$;
6. $\mathsf{select}(x_i, y_i) = z_i$ for all $(x,y) \in A \times B$ and $i \notin I$.

All six invariants are true at the beginning of the algorithm. To prove invariants 1 through 4, we assume they hold up to the current point of the algorithm and prove they hold after executing the next line.

The set A is modified at lines 7 and 10, both of which ensure that $\pi_I(A)$ is thick, therefore invariant 1 holds. We need to argue, though, that the assumptions of the corresponding lemmas hold and therefore we are allowed to apply them. For line 7 we need to argue that Lemma B.4.5 applies, and that is the case, since we assume invariant 1 holds before this point, and since invariants 3 and 4 together with the assumption of Lemma B.4.6 that $c_v^A + c_v^B \leq m \log \ell = \ell^\gamma \log \ell$ imply that $\beta(C_I) \leq \chi + c_v^B \leq 2c_v^A/(1-\lambda)\log \ell + c_v^B \leq c_v^A + c_v^B \leq \ell^\gamma \log \ell$. For line 10 we need to argue that Lemma B.4.4 applies and, indeed, we have that

$$AvgDeg_i(\pi_I(A \cap X^{v_b})) = \frac{|\pi_I(A \cap X^{v_b})|}{|\pi_{I\setminus\{i\}}(A \cap X^{v_b})|} \geq \frac{\frac{1}{2}|\pi_I(A)|}{|\pi_{I\setminus\{i\}}(A)|} = \frac{1}{2} AvgDeg_i(\pi_I(A)) \geq \ell^\lambda/2 \ . \tag{B.9}$$

We note that the conditions for Lemmas B.4.4 and B.4.5 to apply are more relaxed than what we prove. This is so because we will apply the same lemmas when dealing with real communication and there we will need this extra slack.

For invariant 2, first note that A and B never increase. Moreover, the rectangle of compatible inputs R^v changes when v is modified at lines 10 and 16, and in both cases we restrict A (resp. B) to a subset of X^b (resp. Y^b).

To see that we make at most $2c_v/(1-\lambda)\log \ell$ queries, we observe that each query comes from a call to `project` in line 6, which decreases α by at least $(1-\lambda)\log \ell$ because $AvgDeg_i(A_I) < \ell^\lambda$. However, $\alpha(\pi_I(A)) \leq 2c_v^A$ since, at line 9, b is chosen so that $\alpha(\pi_I(A \cap X^{v_b})) \leq \alpha(\pi_I(A)) + 1$ and thus, by Lemma B.4.4, α increases by at most 2 at line 10, i.e., after one bit of communication. Since $\alpha \geq 0$ at all times by definition, the upper bound in invariant 3 follows.

Note that C_I is only updated at lines 3 and 7. At line 3 it holds by the fact that invariant 5 is true at the beginning of a round. At line 7, Lemma B.4.5 guarantees that $\beta({C_{I\setminus\{i\}}}^{(0)}(U) \cap {C_{I\setminus\{i\}}}^{(1)}(U)) \leq \beta(C_I) + 1$. Note that it is possible to apply Lemma B.4.5 as argued before. Therefore, invariant 4 follows.

To prove that invariant 5 holds at the end of a round we distinguish whether Alice or Bob spoke. If Bob spoke, B was updated at line 16 and the invariant clearly holds since each bit b that Bob says is chosen such that $\beta(\pi_I(B \cap Y^{v_b}))$ increases by at most 1.

If Alice spoke, B is updated in line 13. Let $Q = \{i_1, \ldots, i_{|Q|}\}$. Let I_0 be the non-queried coordinates at the beginning of the round and $I_\eta = I_{\eta-1} \setminus \{i_\eta\}$, for $\eta \in [|Q|]$. Moreover, let B_0 be the value of B at the beginning of the round and $B_\eta = \rho_{i,V}(B_{\eta-1})$, for $\eta \in [|Q|]$ and $i = i_\eta$. We prove by induction over η that $\pi_{I_\eta}(B_\eta) \supseteq C_{I_\eta}$. Recall that C_{I_η} is a set whose elements are vectors with coordinates in I_η and is not the projection of a set C to I_η. Therefore, the subscript serves not only as a reminder of the form of its elements, but also as an identifier of the set. At the beginning of the round $\pi_{I_0}(B_0) = C_{I_0}$. At each iteration,

$$\pi_{I_\eta}(B_\eta) = \pi_{I_\eta}(\rho_{i,V}(B_{\eta-1})) \supseteq \pi_{I_\eta}(\rho_{i,V}(\pi_{I_{\eta-1}}(B_{\eta-1}))) \tag{B.10}$$

$$\supseteq \pi_{I_\eta}(\rho_{i,V}(C_{I_{\eta-1}})) = {C_{I_{\eta-1}}}^{(z_i)}(U_i) \supseteq C_{I_\eta} \tag{B.11}$$

so at the end $\beta(\pi_{I_{|Q|}}(B_{|Q|})) \leq \beta(C_{I_{|Q|}}) \leq \chi + c_v^B$, where the last inequality follows from invariant 4.

For invariant 6, recall that A and B never increase. Furthermore, each time I is modified at line 8, we add to Q the coordinate i for which invariant 6 no longer holds (since the element i was removed from I and it has not been queried yet). Then we restore the invariant before the next iteration by restricting B at line 13. Indeed, if $(x,y) \in A \times B$, then $x_i \in U_i$ and $y_i \in V^{z_i}(U_i)$, so by definition of V it holds that $y_{i x_i} = z_i$.

We have finished proving the invariants. From now on we assume that the algorithm ended and reached a leaf v. It is clear that the decision tree has depth r and the total number of queries is at most $2c/(1-\lambda)\log \ell < m$ by invariant 3. It remains to prove correctness, that is, for any $z \in \{0,1\}^m$ that we fix, the decision tree answered $S(z)$.

In order to prove this, we show that there exists an input $(x,y) \in R_v$ such that $\mathsf{select}(x,y) = z$, which implies $\mathsf{eval}(\Pi, z) = \Pi(x,y) = (\mathit{Lift}(S))(x,y) = S(\mathsf{select}(x,y)) = S(z)$. This is done by restricting A and B to A' and B' so that any input in A' and any input in B' are compatible with z and finally arguing that A' and B' are non-empty. For every queried coordinate i, we have already restricted A and B so that for any $x \in A$ and any $y \in B$, $\mathsf{select}(x_i, y_i) = z_i$; thus we are left to deal with the non-queried coordinates.

Note that $\mathit{Graph}_i(\pi_{\{i\}}(A))$ has only one vertex on the right with label $\emptyset$, and an edge from this vertex to an element $x_i \in [\ell]$ if $x_i \in \pi_{\{i\}}(A)$. Although $\mathit{Graph}_i(\pi_\emptyset(A))$ is not defined (which is the reason we have to apply Claims B.4.7 and B.4.8 directly and not Lemma B.4.5), the projection $\pi_\emptyset(A)$ is well-defined and it is either the empty set, in which case A is empty, or it is the singleton set containing the empty string, in which case there exists at least one element $x \in A$. The same holds for B, $\pi_\emptyset(B)$ is either empty or it is the singleton set containing the empty string. Therefore, by the definition of α and β, if $\alpha(\pi_\emptyset(A))$ (resp. $\beta(\pi_\emptyset(B))$) is finite, then A (resp. B) is non-empty.

We start with $A' = A$, $B' = B$ and $I' = I$ such that $\pi_{I'}(A')$ is thick, $\beta(\pi_{I'}(B')) \leq \chi + c_v^B \leq \ell^\gamma \log \ell$ and I' is non-empty since we queried less than m coordinates. While I' is not empty, we choose $i \in I'$ and apply Claims B.4.7 and B.4.8 to get a set U such that $\pi_{I' \setminus \{i\}}(\rho_{i,U}(A')) = \pi_{I' \setminus \{i\}}(A')$ and $\beta(\pi_{I' \setminus \{i\}}(B')^{(0)}(U) \cap \pi_{I' \setminus \{i\}}(B')^{(1)}(U)) \leq \beta(\pi_I(B')) + 1$. We then set $A' = \rho_{i,U}(A')$ and $B' = \rho_{i,V}(B')$, where $V = V^{z_i}(U)$, and thus, by definition of V, for all $x \in A'$ and $y \in B'$ it holds that $\mathsf{select}(y_i, x_i) = z_i$. Finally we set $I' = I' \setminus \{i\}$ and repeat. Note that these claims can indeed be applied while I' is not empty, since the fact that thickness is monotone (Observation B.4.2) implies $\pi_{I'}(A')$ is thick, and since $\beta(\pi_{I'}(B')) \leq c_v^B + (\chi + |I|) \leq c + m < 2\ell^\gamma \log \ell$, because $\beta(\pi_{I'}(B'))$ is increasing by at most 1 for every $i \in I$.

At the end of this process, the set B' is such that $\beta(\pi_{I'}(B')) \leq 2\ell^\gamma \log \ell < \infty$, and thus B' is non-empty. Moreover, we have that $\pi_\emptyset(A') = \pi_\emptyset(A)$, and since A is non-empty (because $\alpha(\pi_I(A)) \leq 2c_v^A$), $\pi_\emptyset(A')$ is the singleton set containing the empty string and, therefore, A' is non-empty. □

Now we explain the auxiliary procedures. Procedure `prune` just removes vertices with too small degree. Thus, we only need to argue that this process does not last for too long.

Proof of Lemma B.4.4. $A'_I = \pi_I(A')$ is thick by construction. It remains to show that $|A'_I| \geq |\pi_I(A)|/2$. For each coordinate $i \in I$ we observe that, by the definition of average degree, there are at most $|\pi_I(A)|/\mathit{AvgDeg}_i(\pi_I(A))$ right vertices of positive degree in

```
1 let $A_I = \pi_I(A)$
2 while $\exists i$ such that $MinDeg_i(A_I) < \ell^\mu$ do
3     let $x'_{I\setminus\{i\}} \in V(Graph_i(A_I))$ be a right vertex of degree less than $\ell^\mu$
4     let $A_I = \{x \in A_I : x_{I\setminus\{i\}} \neq x'_{I\setminus\{i\}}\}$
5 return $\{x \in A : x_I \in A_I\}$
```

Figure B.5: Procedure prune(A, I)

```
1 let $U \subseteq [\ell]$ be a set such that
2     $\pi_{I\setminus\{i\}}(\rho_{i,U}(A)) = \pi_{I\setminus\{i\}}(A)$
3     $\beta({C_{I\setminus\{i\}}}^{(0)}(U) \cap {C_{I\setminus\{i\}}}^{(1)}(U)) \geq \beta(C_I) + 1$
4 return $U$
```

Figure B.6: Procedure project(A, C_I, I, i)

$Graph_i(\pi_I(A))$, and, moreover, since $Graph_i(A'_I) \subseteq Graph_i(\pi_I(A))$, there are at most $|\pi_I(A)|/AvgDeg_i(\pi_I(A))$ right vertices of positive degree in $Graph_i(A'_I)$. Clearly this is an upper bound on the number of iterations of the procedure prune for coordinate i, since every iteration removes a right vertex of positive degree.

Since $AvgDeg_i(\pi_I(A)) \geq \ell^\lambda/4$ for all $i \in I$, the total number of iterations for each coordinate is at most $|\pi_I(A)|/AvgDeg_i(\pi_I(A)) \leq 4|\pi_I(A)|/\ell^\lambda$, which makes a total of at most $4|I||\pi_I(A)|/\ell^\lambda \leq 4\ell^{\gamma-\lambda}|\pi_I(A)|$ iterations overall. Each iteration removes $\deg(X'_{I\setminus\{i\}})$, which is less than ℓ^μ, elements from A'_I, for a total of at most $4\ell^{\gamma+\mu-\lambda}|\pi_I(A)| \leq |\pi_I(A)|/2$ elements removed. The last inequality holds because of assumption (B.5a). Thus, at least $|\pi_I(A)|/2$ elements remain. □

Our Lemma B.4.5 is slightly stronger than the projection lemmas in [RM99, GPW15] because it needs to handle both outcomes of the query to z_i, so we care not only about each of $\rho_{i,V^b(U)}(B)$ separately but about $B^{(0)}(U) \cap B^{(1)}(U)$. Nonetheless, essentially the same proof of [RM99] using the probabilistic method works—but not that of [GPW15], where the probabilities are too small for us. Procedure project, therefore, just asserts the existence of a return value U with the required properties.

The probabilities in the claims below are with respect to U picked uniformly among all subsets of $[\ell]$ of size ℓ^δ.

Claim B.4.7 ([RM99]). *If A is thick, then $\pi_{I\setminus\{i\}}(\rho_{i,U}(A)) = \pi_{I\setminus\{i\}}(A)$ with probability* $1 - o(1)$.

Claim B.4.8. *If $\beta(C_I) \leq 2\ell^\gamma \log \ell$, then $\beta({C_{I\setminus\{i\}}}^{(0)}(U) \cap {C_{I\setminus\{i\}}}^{(1)}(U)) \leq \beta(C_I) + 1$ with probability* $1 - o(1)$.

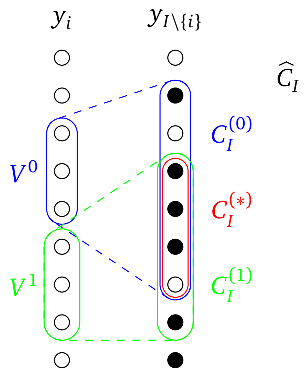


Figure B.7: Sets used in the proof of Claim B.4.8

Proof of Lemma B.4.5. By union bound there exists U that satisfies Claim B.4.7 and Claim B.4.8. Item 1 then follows from Observation B.4.2, and item 2 from Observation B.4.3. □

Proof of Claim B.4.7. We observe that the equality holds if every right vertex of the graph $\mathit{Graph}_i(\pi_I(A))$ with positive degree has an edge into U. Since $\mathit{MinDeg}_i(\pi_I(A)) \geq \ell^\mu$, the probability that U is contained within the non-neighbours of any right vertex $x_{I\setminus\{i\}}$ is

$$\frac{\binom{\ell-|N_i(x_{I\setminus\{i\}})|}{\ell^\delta}}{\binom{\ell}{\ell^\delta}} \leq \frac{\binom{\ell-\ell^\mu}{\ell^\delta}}{\binom{\ell}{\ell^\delta}} \leq (1-\ell^{\mu-1})^{\ell^\delta} \leq e^{-\ell^{\mu+\delta-1}} \quad . \tag{B.12}$$

By a union bound over all right vertices, the probability of the equality not holding is at most

$$\ell^{|I|-1}e^{-\ell^{\mu+\delta-1}} < e^{|I|\log\ell-\ell^{\mu+\delta-1}} \leq e^{\ell^\gamma\log\ell-\ell^{\mu+\delta-1}} = o(1) \ . \tag{B.13}$$

The last equality holds because of assumption (B.5b). □

The proof of Claim B.4.8 follows [RM99] except that our version of Claim B.4.10 is stronger and we apply it twice.

Proof of Claim B.4.8. We begin by observing that $C_{I\setminus\{i\}}{}^{(b)}(U)$, the set of right vertices that can be completed to b over U, is equal to $N_j(V^b(U))$ (see Figure B.7). We want to prove that the set $C^{(*)}_{I\setminus\{i\}}(U) = C_{I\setminus\{i\}}{}^{(0)}(U) \cap C_{I\setminus\{i\}}{}^{(1)}(U)$ of right vertices that can be completed to any colour over U is large, namely that $|C^{(*)}_{I\setminus\{i\}}|/2^{\ell(|I|-1)} \geq \psi/2$ where $\psi = |C_I|/2^{\ell|I|} = 2^{-\beta(C_I)}$.

Right vertices of degree larger than $2^\ell\psi/4$ can be completed to any colour with high probability. Indeed, $|N_i(y_{I\setminus\{i\}})| \geq 2^\ell \cdot \psi/4 = 2^\ell \cdot 2^{-\beta(C)}/4 \geq 2^\ell \cdot 2^{-2\ell^\gamma\log^2\ell}/4 \geq 2^\ell \cdot 2^{-3\ell^\gamma\log^2\ell}$, so for each $b \in \{0,1\}$ we can apply Claim B.4.10 with $\phi = 3\log^2\ell$ to show that $N_i(y_{I\setminus\{i\}}) \cap V^b(U) \neq \emptyset$ with probability $1 - o(1)$. Taking a union bound, we can assume that $N_i(y_{I\setminus\{i\}}) \cap V^b(U) \neq \emptyset$ holds for both $b \in \{0,1\}$ except with probability $o(1)$. In other words, $y_{I\setminus\{i\}} \in C^{(b)}_{I\setminus\{i\}}$ for $b \in \{0,1\}$, so $y_{I\setminus\{i\}} \in C^{(*)}_{I\setminus\{i\}}$.

We have shown that for every $y_{I\setminus\{i\}} \in \widehat{C}_{I\setminus\{i\}} = \{y_{I\setminus\{i\}} : \deg(y_{I\setminus\{i\}}) \geq 2^\ell \psi/4\}$ it holds that $y_{I\setminus\{i\}} \in C^{(*)}_{I\setminus\{i\}}$ with probability $1 - o(1)$. By Observation B.4.9, with probability $1 - o(1)$,

$$|C^{(*)}_{I\setminus\{i\}}| \geq 2/3 \cdot |\widehat{C}_{I\setminus\{i\}}| \ . \tag{B.14}$$

In fact, 2/3 can be chosen to be any arbitrary number strictly smaller than 1, and the statement would still hold.

Therefore it is enough to prove that the set $\widehat{C}_{I\setminus\{i\}}$ of right vertices with large degree is large. Indeed, we have

$$\begin{aligned} 2^{\ell|I|}\psi = |C_I| &\leq |\widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell + |\{0,1\}^{\ell(|I|-1)} \setminus \widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell \cdot \psi/4 & \text{(B.15)} \\ &\leq |\widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell + 2^{\ell(|I|-1)} \cdot 2^\ell \cdot \psi/4 = |\widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell + 2^{\ell|I|} \cdot \psi/4 \ , & \text{(B.16)} \end{aligned}$$

from where

$$|\widehat{C}_{I\setminus\{i\}}| \geq 3/4 \cdot \psi \cdot 2^{\ell(|I|-1)} \ . \tag{B.17}$$

and the claim follows by combining equations (B.14) and (B.17). We observe that the 1/4 in the definition of right vertices with large degree can be any arbitrarily small constant. □

Observation B.4.9. *Let T be a set and S a set-valued random variable. If* $\Pr[s \in S] \geq p$ *for every $s \in T$, then* $\Pr[|S| \geq q|T|] \geq (p-q)/(1-q)$.

Proof. Let $x = \Pr[|S| \geq q|T|]$. Then

$$\begin{aligned} p &\leq \Pr[s \in S] & \text{(B.18)} \\ &= \Pr[s \in S \,|\, |S| \geq q|T|]\Pr[|S| \geq q|T|] + \Pr[s \in S \,|\, |S| < q|T|]\Pr[|S| < q|T|] & \text{(B.19)} \\ &\leq 1 \cdot x + q \cdot (1-x) \ , & \text{(B.20)} \end{aligned}$$

from which the observation follows. □

As before, we think of $W \subseteq \{0,1\}^\ell$ as a set of binary colourings of $[\ell]$ and denote by $\pi_U(W)$ the projection of W to a subset $U \subseteq [\ell]$, i.e. $\pi_U(W) = \{w_U \in \{0,1\}^U : w \in W \text{ for some } w_{[\ell]\setminus U} \in \{0,1\}^{[\ell]\setminus I}\}$. Note that this is the same operation as $\pi_I(A)$, except they apply to different domains.

Claim B.4.10. *Let $W \subseteq \{0,1\}^\ell$ be any set of size at least $2^{-\phi\ell^\gamma} \cdot 2^\ell$, where ϕ is any function of ℓ such that* $\log\phi = o(\log\ell)$*. Let U be a uniformly random subset of $[\ell]$ of size ℓ^δ. Then, for any $b \in \{0,1\}$, $\{b\}^{|U|} \in \pi_U(W)$ with probability at least $1 - o(1)$.*

In the original paper [RM99] the constants are set to $\gamma = 2/20$, $\delta = 5/20$. The same probabilistic argument works for any choice of constants such that $\gamma + 3\delta/2 < 1$. Here we present a combinatorial proof that works for any constants such that $\gamma + \delta < 1$, and in particular holds for $\gamma = 1/3 - \xi$ and $\delta = 2/3$, for any $\xi > 0$.

Proof. We will prove the following equivalent statement: if $\Pr_U[\{b\}^{|U|} \notin \pi_U(W)] \geq q$ (for, say, $q = \phi/\log \ell$), then $|W| \leq 2^{\ell - \phi\ell^\gamma}$. We will only use the fact that $\gamma + \delta < 1 + \frac{\log \frac{q}{\phi}}{\log \ell}$ (for $q = \phi/\log \ell$ this says that $\gamma + \delta < 1 - \frac{\log\log \ell}{\log \ell}$). The following statement, which is a corollary of Kruskal–Katona Theorem, will be proved later on.

Claim B.4.11. *Let $\mathcal{U}$ be any family of subsets of $[\ell]$, where every $U \in \mathcal{U}$ is of size u. If $|\mathcal{U}| \geq \sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i}$ and if $W \subseteq \{0,1\}^\ell$ is such that $\{b\}^{|U|} \notin \pi_U(W)$ for every $U \in \mathcal{U}$, then it holds that $|W| \leq 2^\ell - \sum_{j=0}^{\ell-u} \sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i-j}$.*

Let $\mathcal{U}$ be the set of all $U \subset [\ell]$ of size ℓ^δ such that $\{b\}^{|U|} \notin \pi_U(W)$. We have that

$$\frac{|\mathcal{U}|}{\binom{\ell}{\ell^\delta}} = \Pr_U[\{b\}^{|U|} \notin \pi_U(W)] \geq q \ . \tag{B.21}$$

Hence we get

$$|\mathcal{U}| \geq q\binom{\ell}{\ell^\delta} = q\frac{\ell}{\ell^\delta}\binom{\ell-1}{\ell^\delta-1} \geq \sum_{i=0}^{q\frac{\ell}{\ell^\delta}} \binom{\ell-1-i}{\ell^\delta-1} = \sum_{i=0}^{q\frac{\ell}{\ell^\delta}} \binom{\ell-1-i}{\ell-\ell^\delta-i} \ . \tag{B.22}$$

We can therefore apply Claim B.4.11 with $u = \ell^\delta$ and $t = q\frac{\ell}{\ell^\delta}$ to get

$$|W| \leq 2^\ell - \sum_{j=0}^{\ell-\ell^\delta} \sum_{i=0}^{q\frac{\ell}{\ell^\delta}} \binom{\ell-1-i}{\ell-\ell^\delta-i-j} \leq 2^{\ell - q\frac{\ell}{\ell^\delta}+1} + 2^{\ell^\delta \log \ell} \leq 2^{\ell-\phi\ell^\gamma} \ , \tag{B.23}$$

where the second inequality is a straightforward calculation that we prove in Claim B.4.15; and the last inequality follows from $\gamma + \delta < 1 + \frac{\log \frac{q}{\phi}}{\log \ell}$. □

To prove Claim B.4.11 we need to introduce some terminology from extremal combinatorics.

We use the following terminology from [Juk11]. If w is a binary colouring of $[\ell]$ (i.e. a binary vector of length ℓ), we say a *neighbour* of a w is a colouring which can be obtained from w by flipping one of its 1-entries to 0. A *shadow* of a set $A \subseteq \{0,1\}^\ell$ of binary colourings is the set $\partial(A)$ of all its neighbours. A set A is *k-regular* if every colouring in A colours exactly k elements 1. Note that in this case $\partial(A)$ is $(k-1)$-regular. The best possible lower bounds for the size of $\partial(A)$ were obtained independently by Kruskal [Kru63] and Katona [Kat68].

Theorem B.4.12 (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

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

Let $\mathcal{P}(S)$ denote the power set of S. In [Fra84] it is proven that there exists an explicit compression function $C : \mathcal{P}(\{0,1\}^\ell) \to \mathcal{P}(\{0,1\}^\ell)$ such that, given a k-regular set $A \subseteq \{0,1\}^\ell$, $C(A)$ is k-regular, $|C(A)| = |A|$ and $|\partial(C(A))|$ matches the lower bound in Theorem B.4.12, and, furthermore, the following proposition holds.

Proposition B.4.13. $\partial(C(A)) \subseteq C(\partial(A))$.

Although this follows directly from the proposition in Section 2 of [Fra84], the formulation above is from [And87].

We define the *iterated shadow* of a k-regular set $A \subseteq \{0,1\}^\ell$ to be $\partial^{\leq k}(A) = \cup_{j=0}^{k} A_j$, where $A_0 = A$ and $A_j = \partial(A_{j-1})$ for $0 < j \leq k$. The following corollary follows immediately from Theorem B.4.12 and Proposition B.4.13.

Corollary B.4.14. *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

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

Proof of Claim B.4.11. Given $\mathcal{U}$, define $W_{\mathcal{U}}$ to be the largest set of colourings $\{0,1\}^\ell$ such that $\{b\}^{|U|} \notin \pi_U(W)$ for all $U \in \mathcal{U}$. For simplicity, we will consider $b = 0$, i.e. $W_{\mathcal{U}}$ is the set that contains all the colourings of $[\ell]$ that do not colour any $U \in \mathcal{U}$ completely 0.

Let $\mathcal{U}' \subseteq \mathcal{U}$ be a set of size exactly $\sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i}$. Obviously, $W_{\mathcal{U}'}$ is at least as large as $W_{\mathcal{U}}$.

Let $\mathbf{1}_{U^c} \in \{0,1\}^\ell$ be the indicator functions for the complement of a set U, i.e. $\mathbf{1}_{U^c} = 1 - \mathbf{1}_U$. Let A be the set of $\mathbf{1}_{U^c} \in \{0,1\}^\ell$ for $U \in \mathcal{U}'$. Note that A is $(\ell-u)$-regular and that the iterated shadow of A is exactly the set of colourings that are not in $W_{\mathcal{U}'}$. Applying Corollary B.4.14 to A, we get

$$|\partial^{\leq k}(A)| \geq \sum_{j=0}^{k} \sum_{i=0}^{t} \binom{\ell-1-i}{k-i-j} = \sum_{j=0}^{\ell-u} \sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i-j}.$$

Therefore, $|W_{\mathcal{U}}| \leq |W_{\mathcal{U}'}| = 2^\ell - |\partial^{\leq k}(A)| \leq 2^\ell - \sum_{j=0}^{\ell-u} \sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i-j}$. □

For completeness we include the calculations needed in Claim B.4.10.

Claim B.4.15. *It holds that*

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

Proof. The claim follows from the following sequence of elementary calculations

$$\begin{aligned}
\sum_{j=0}^{\ell-u}\sum_{i=0}^{t}\binom{\ell-1-i}{\ell-u-i-j} &= \sum_{j=0}^{\ell-u}\sum_{i=0}^{t}\binom{\ell-1-i}{\ell-1-j} && (*)\\
&= \sum_{j=0}^{\ell-u}\left(\sum_{i=0}^{\ell-1}\binom{i}{\ell-1-j}-\sum_{i=0}^{\ell-t}\binom{i}{\ell-1-j}\right)\\
&= \sum_{j=0}^{\ell-u}\left(\binom{\ell}{\ell-j}-\binom{\ell-t+1}{\ell-j}\right) && (**)\\
&= \sum_{j=0}^{\ell-u}\binom{\ell}{j}-\sum_{j=0}^{\ell-u-t+1}\binom{\ell-t+1}{j} && (*)\\
&= 2^{\ell}-\sum_{j=\ell-u+1}^{\ell}\binom{\ell}{j}-2^{\ell-t+1}+\sum_{j=\ell-u-t+2}^{\ell-t+1}\binom{\ell-t+1}{j}\\
&\geq 2^{\ell}-2^{\ell-t+1}+\sum_{j=0}^{u-1}\binom{\ell}{j}\\
&\geq 2^{\ell}-2^{\ell-t+1}+\ell^{u}=2^{\ell}-2^{\ell-t+1}+2^{u\log\ell}\quad ,
\end{aligned}$$

where the equalities in $(*)$ follow from renaming of variables and the fact that $\binom{n}{k}=\binom{n}{n-k}$; the equality in $(**)$ follows from $\sum_{i=0}^{n-1}\binom{i}{k-1}=\binom{n}{k}$. □

B.4.2 Simulation of Real Communication Protocols by Decision Trees

In this section we show how to adapt the simulation theorem to real communication.

Theorem B.4.16. *If there is a real communication protocol computing Lift(S) using communication c and r rounds, then there is a parallel decision tree computing S that uses $O(c/\log\ell)$ queries and depth r.*

The proof follows the same strategy as in the deterministic case, this is we are going to construct a decision tree by simulating a real communication protocol and only querying the coordinates where the communication protocol would have too much information on x_i.

The major difference in analyzing real communication protocols as opposed to deterministic ones is that the set of compatible inputs is not a rectangle, but a monotone set as defined next.

Definition B.4.17. A Boolean matrix M is *monotone* if $M_{i_1j_1}\leq M_{i_2j_2}$, for all pairs of entries such that $i_1\leq i_2$ and $j_1\leq j_2$.

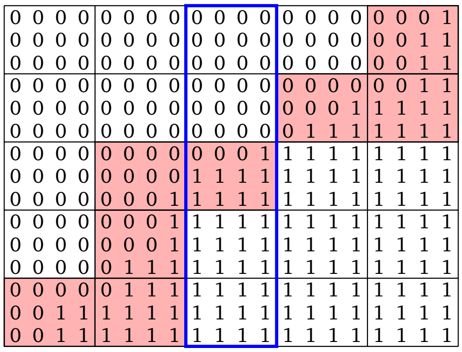


Figure B.8: Monotone matrix partitioned in 5 × 5 blocks; the 3rd block-column has 4 monochromatic blocks.

Recall that each communication step is a comparison $\phi(x) \le \psi(y)$ and that we restrict our attention to inputs in a set $A \times B$. We lay out the results of the comparison in the matrix $(\llbracket \phi(x) \le \psi(y) \rrbracket)_{x,y}$ indexed by $x \in A$, $y \in B$, with rows sorted decreasingly according to ϕ and columns increasingly according to ψ. Note that we use the Iverson bracket notation

$$\llbracket Z \rrbracket = \begin{cases} 1 & \text{if the Boolean expression } Z \text{ is true;} \\ 0 & \text{otherwise.} \end{cases} \tag{B.24}$$

The communication matrix is monotone: if $\phi(x_1) \ge \phi(x_2)$ and $\psi(x_1) \le \psi(x_2)$ then $\phi(x_2) \le \phi(x_1) \le \psi(y_1) \le \psi(y_2)$.

The fact that the set of compatible inputs is not a rectangle can be circumvented, since as observed in [Joh98] in every monotone matrix there exists one quadrant—thus a large rectangle—that is monochromatic. It is therefore possible to restrict the set of compatible inputs to a quadrant when we want to choose the outcome of a comparison, as done in [BEGJ00].

However, this is not enough for us. Since we want to query variables only at the end of each round of k comparisons, and after using procedure `project` we no longer know what B is, we need to restrict the inputs to rectangles beforehand. This means we have to avoid shrinking A too much, and definitely less than the 2^k factor we would get by picking quadrants.

Our solution is to partition the matrix into $(k+1) \times (k+1)$ blocks of size $|A|/(k+1) \times |B|/(k+1)$ and then restrict Bob's input to one of the $(k+1)$ block-columns, so that Alice's input forms a rectangle in k out of the $k+1$ block-rows (see Figure B.8). Formally, we have the following lemma.

Lemma B.4.18. *Let M be a monotone matrix partitioned into $(k+1) \times (k+1)$ blocks. There is a block-column such that k of its blocks are monochromatic.*

Proof. Consider a non-monochromatic block. Since its bottom-right corner has value 1, all blocks below and to the right are 1-blocks. This is, if we consider non-monochromatic blocks left-to-right in a sequence, then a block cannot be below its predecessor, which means that there are at most $2k + 1$ non-monochromatic blocks overall. By the pigeonhole principle, at least one of the column-blocks contains at most one non-monochromatic block. □

We take advantage of this lemma with the following construction. Given two sets $A \subseteq [\ell]^m$ and $B \subseteq \{0,1\}^{\ell m}$, we define the b-monochromatic part of A with respect to B as $A[b,B]_{\phi,\psi} = \{x \in A : \forall y \in B\ [\![\phi(x) \le \psi(y)]\!] = b\}$.

For technical reasons we want each element in $\pi_I(A)$ to have a unique completion to A. Therefore we define a new operator $\sigma_I(A) = \{x \in A : \forall x' \in A \text{ if } x'_I = x_I \text{ then } x < x'\}$, where the order is, say, the lexicographic order. In other words, each element of $\sigma_I(A)$ is the minimum among all elements of A that share the same I coordinates. Observe that $\pi_I(A) = \pi_I(\sigma_I(A))$. We define $\sigma_I(B)$ analogously.

We have all the ingredients to explain the simulation procedure `eval`. Note that the comparisons at lines 4 and 9 are the same.

Lemma B.4.19 (Main Lemma). *If Π is a real protocol that computes Lift(S) using communication $c < \frac{m}{2}(1-\lambda)\log\ell$ and r rounds, then* `eval` *computes S using $5c/(1-\lambda)\log\ell$ queries and depth r.*

Proof. The proof is very similar to the proof of Lemma B.4.6. Let R^v be the set (not necessarily a rectangle) of inputs compatible with node v, let c_v be the amount of communication up to node v, and let r_v be the number of rounds up to node v. Let χ be the number of queries so far, i.e., $\chi = m - |I|$. We show that the following invariants hold throughout the algorithm:

1. $\pi_I(A)$ is thick;

2. $A \times B \subseteq R^v$;

3. $\chi \le (2c_v + 3r_v)/(1-\lambda)\log\ell$;

4. $\beta(C_I) \le (c_v + k_v)\log(c_v + 1) + \chi$;

and the following invariants hold at the beginning of each round:

5. $\beta(\pi_I(B)) \le c_v \log(c_v + 1) + \chi$;

6. $\mathsf{select}(x_i, y_i) = z_i$ for all $(x,y) \in A \times B$ and $i \notin I$.

All five invariants are true at the beginning of the algorithm.

The main difference with the proof of Lemma B.4.6 is proving invariant 1, because we modify A not only at lines 12 and 15, but also at line 7. At each point A is modified, the corresponding procedure ensures that $\pi_I(A)$ is thick. We need to argue, though,

```
1  let $A = [\ell]^m$, $B = \{0,1\}^{\ell m}$, $I = [m]$, $v$ be the root of $\Pi$
2  while $v$ is not a leaf do
3      let $A' = \sigma_I(A)$, $B = \sigma_I(B)$
4      foreach comparison $\phi$ vs $\psi$ do
5          let $B = \operatorname{argmax}_{|B'|=|B|/(k_v+1)} |A'[0,B']_{\phi,\psi} \cup A'[1,B']_{\phi,\psi}|$
6          let $A' = A'[0,B]_{\phi,\psi} \cup A'[1,B]_{\phi,\psi}$
7      let $A$ = prune $(A', I)$
8      let $Q = \emptyset$, $C_I = \pi_I(B)$
9      foreach comparison $\phi$ vs $\psi$ do
10         while $\exists i \in I$ such that $AvgDeg_i(\pi_I(A)) < \ell^\lambda$ do
11             let $U_i$ = project $(A, C_I, I, i)$
12             let $A = \rho_{i,U_i}(A)$, $C_{I\setminus\{i\}} = C_{I\setminus\{i\}}{}^{(0)}(U_i) \cap C_{I\setminus\{i\}}{}^{(1)}(U_i)$
13             let $I = I \setminus \{i\}$, $Q = Q \cup \{i\}$
14         let $b_j = \operatorname{argmax} |\pi_I(A[b,B]_{\phi,\psi})|$
15         let $A$ = prune$(A[b_j,B]_{\phi,\psi}, I)$
16     query coordinates $Q$ to get string $z_Q$
17     for $i \in Q$ do
18         let $B = \rho_{i,V}(B)$, where $V = V^{z_i}(U_i)$
19     let $v = v_{b_1,\dots,b_k}$
20 return the answer at $v$
```

Figure B.9: Procedure eval(Π,z)

that the assumptions of the corresponding lemmas hold, and therefore it is correct to apply them. The argument for applying prune in line 15 is the same as in the proof of Lemma B.4.6. For line 12, we note that, by invariants 4 and 3, $\beta(C_I) \le (c_v + k_v) \cdot \log(c_v+1) + \chi \le \frac{m}{2}(\log \ell) \cdot 2(\log m) + 5c_v \le \ell^\gamma \log^2 \ell + 5\ell^\gamma \log \ell \le 2\ell^\gamma \log^2 \ell$. Hence we only need to prove that we can apply Lemma B.4.4 in line 7.

We begin by observing that at line 3, $AvgDeg_i(\pi_I(A')) \ge \ell^\lambda$, since $\pi_I(A') = \pi_I(A)$. Furthermore, at line 6, the size of A' decreases by at most a $1-1/(k_v+1)$ fraction. Indeed, if we divide the comparison matrix $(\llbracket \phi(x) \le \psi(y) \rrbracket)_{x,y}$ into $(k_v+1) \times (k_v+1)$ blocks of size $|A'|/(k_v+1) \times |B|/(k_v+1)$, by Lemma B.4.18 at least one of the column-blocks contains k_v monochromatic blocks, i.e., a $1-1/(k_v+1)$ fraction is monochromatic.

Since we have at most k_v comparisons, the size of A' at line 7 is at least a $(1-1/(k_v+1))^{k_v} \ge 1/4$ fraction of the original. Also, after line 3 there is a bijection between A' and $\pi_I(A')$, so the size of $\pi_I(A')$ is also at least a $1/4$ fraction and Lemma B.4.4 applies.

For invariant 2, note that A and B never increase and that the set of compatible inputs R^v only changes when v is modified at line 19. However, A was restricted at line 15 so that $\llbracket \phi(x) \le \psi(y) \rrbracket = b$ for every $x \in A$ and $y \in B$; in other words $A \times B \subseteq R^{v_b}$, therefore invariant 2 holds. Note that we can only restrict A in this manner because we had already restricted B in line 5.

To see that we make at most $(2c_v + 3r_v)/(1-\lambda)\log\ell$ queries, we observe that each query comes from a call to `project` in line 11, which decreases α by at least $(1-\lambda)\log\ell$ because $AvgDeg_i(A_I) < \ell^{\lambda}$. However, α only increases (by at most 2) at line 15, i.e., after one bit of communication, and (by at most 3) at line 7, i.e., once per round. Since $\alpha \geq 0$ at all times by definition, the upper bound in invariant 3 follows.

To prove invariants 4 and 5 we observe that B shrinks at two points. One is at line 5, where β increases by $\log(k_v + 1) \leq \log c$ with every bit of communication. Therefore, when we set $C_I = \pi_I(B)$ at line 8 invariant 4 holds. In line 12, Lemma B.4.5 guarantees that $\beta(C_I)$ increases by at most 1 with every query, therefore $\beta(C_I) \leq (c_v + k_v)\log(c + 1) + \chi$ holds at all times. Finally, we note that $\beta(\pi_I(B)) \leq \beta(C_I)$, by the same argument as in the proof of invariant 5 of Lemma B.4.6, and that c_v is updated to $c_v + k_v$ before the next round.

For invariant 6, recall that A and B never increase. Moreover, each time I is modified at line 13, we add the coordinate i for which invariant 6 breaks to Q. Then we restore the invariant before the next iteration by restricting B at line 18. Indeed, if $(x, y) \in A\times B$, then $x_i \in U$ and $y_i \in V^{z_i}(U)$, so by definition of V it holds that $y_{ix_i} = z_i$.

It is clear that the decision tree has depth r and the total number of queries is at most $5c/(1 - \lambda)\log\ell$ by invariant 3. The proof of correctness is identical to that of Lemma B.4.6. □

B.5 From Parallel Decision Trees to Dymond–Tompa Games

In this section we prove that the adversary argument on a parallel decision tree for the falsified search problem of a pebbling contradiction gives a Pebbler strategy for the Dymond–Tompa game.

It is more convenient to work with the Dymond–Tompa game when there is a challenged pebble at all times. Therefore in this and the following section we use an alternative but equivalent definition. Initially the unique sink has a pebble and it is challenged, and then the game starts without a special first round. The number of rounds is the number of actual rounds, not counting the setup, and the cost is the total number of pebbles, including the initial pebble on the sink.

Lemma B.2.6 (Restated). *If there is a parallel decision tree for $Search(Peb_G)$ in depth r using at most c queries, then Pebbler has a winning strategy in the r-round Dymond–Tompa game on G in cost at most $c + 1$.*

We prove that, in fact, the parallel decision tree complexity of the falsified clause search problem of a pebbling contradiction is equivalent to the Dymond–Tompa game on the graph with an extra sink on top. Formally, we define $\widehat{G}$ as a graph with vertices $V(G) \cup \{t\}$ and edges $E(G) \cup \{(z, t)\}$, where z is the unique sink of G. Clearly the game on $\widehat{G}$ needs as many pebbles as G, and one more pebble is enough, so Lemma B.2.6 follows from Lemma B.5.1.

Lemma B.5.1. *There is a parallel decision tree for $Search\big(Peb_G\big)$ in depth r using c queries if and only if Pebbler has a winning strategy in the r-round Dymond–Tompa game on $\widehat{G}$ in cost $c+1$.*

Proof. Assume there is a parallel decision tree for $Search\big(Peb_G\big)$ in depth r using c queries. We construct a strategy for Pebbler in r rounds and $c+1$ pebbles. We say that a vertex s reaches a vertex t if there is a (possibly empty) path from s to t where all intermediate vertices are not queried. We keep these invariants.

1. The challenged pebble in the Dymond–Tompa game is false.
2. A false vertex is reachable from another false vertex if and only if it is not challenged in the Dymond–Tompa game.
3. In the subtree of the challenged pebble a vertex has a pebble if and only if it has been queried.

When it is Pebbler's turn, Pebbler looks at the decision tree and places pebbles in those vertices being queried that can reach the challenged pebble. After Challenger's turn, Pebbler follows the branch in the decision tree in which the challenged pebble is false and other vertices are false if they are reachable from a false vertex or true otherwise.

Dymond–Tompa moves are valid and the invariants are kept. When we reach a leaf in the decision tree we made at most c queries in r rounds by assumption, therefore Pebbler also used at most c pebbles on vertices of G plus one pebble on the extra sink and r rounds. It remains to show that the Dymond–Tompa game also ended. The decision tree points to a falsified clause, which is not the sink axiom because the sink is always false. Therefore we have a false vertex whose predecessors are true. By item 2, that false pebble is challenged, and by item 3 all of its predecessors have pebbles, therefore the Dymond–Tompa game also ended.

Assume there is a Pebbler strategy in r rounds and $c+1$ pebbles. We construct a parallel decision tree for $Search\big(Peb_G\big)$ in depth r using c queries.

We look at the strategy for Pebbler and add a node to the decision tree that queries the variables corresponding to vertices being pebbled that can reach the challenged pebble. For each branch, we simulate a Challenger move. We consider the set of new vertices coloured false and that are not reachable by any false vertex. If this set is empty, then Challenger stays. Otherwise Challenger jumps to any of these vertices.

Dymond–Tompa moves are valid and the invariants are kept. When the Dymond–Tompa game ends, Pebbler has used at most $c+1$ pebbles in r rounds by assumption, one of which outside G, therefore the decision tree also made at most c queries in r rounds. It remains to show that we can label the leaves of the decision tree in such a way that the assignment induced by the decision tree falsifies a clause. At the end of the Dymond–Tompa game, all of the predecessors of the challenged pebble have pebbles. By item 1 it is false, and by item 3 its predecessors are queried. By item 2, its predecessors are true, therefore we can label the leaf with the clause claiming that if the predecessors of the challenged vertex are true then the challenged vertex is true. □

B.6 Dymond–Tompa Trade-offs

In this section we prove upper and lower bounds for the Dymond–Tompa game on graphs of a given family. The lower bounds are the final missing piece in order to get length-space trade-offs for cutting plane proofs, and the upper bounds will be used to obtain space-efficient proofs, as explained in Section B.7.

Our goal is to prove the following lemma.

Lemma B.6.1. *For any $n, d \in \mathbb{N}^+$ such that n is a power of 2, there exists an explicitly constructible DAG $G(n,d)$ of depth d with $\mathrm{O}(dn)$ vertices and indegree at most 2 such that:*

1. *for any $r \leq d$, the cost of an r-round DT game is at most $\min\{r2(2^{\lceil d/r \rceil}-1), rn(2^{\lceil \lceil \log d \rceil / r \rceil}-1)\}$;*
2. *for any $r \leq d$, the cost of an r-round DT game is at least $\min\{\frac{r2^{d/r}}{8}, \frac{n}{8}\}$.*

We first define a family of graphs for which we will prove the lemma.

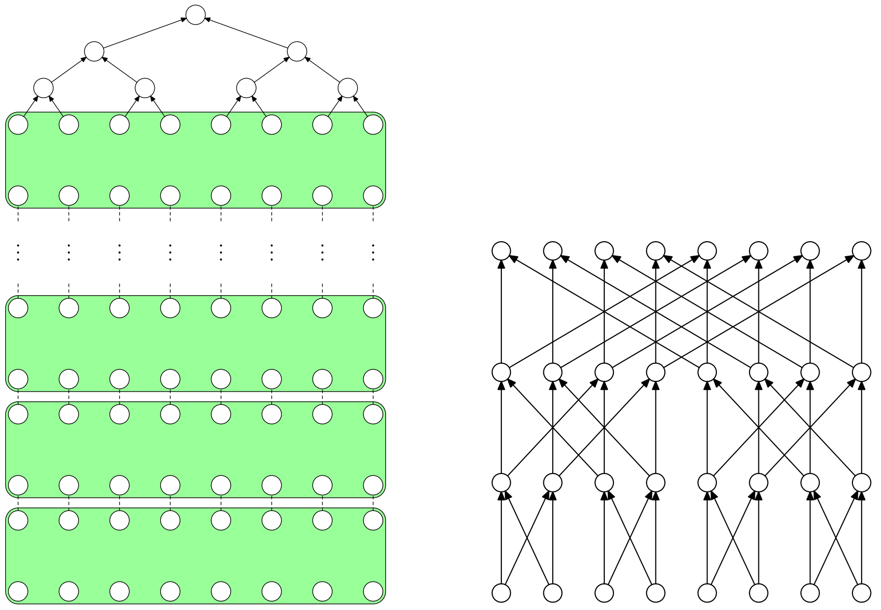
Definition B.6.2 (Butterfly graph). A *k-dimensional butterfly graph G* is a DAG with vertices labelled by pairs (w,i) for $0 \leq w \leq 2^k - 1$ and $0 \leq i \leq k$, and with edges from vertex (w,i) to $(w', i+1)$ if the binary representations of w and w' are equal except for possibly in the $(i+1)$st most significant bit. Note that G has $(2^k+1)k$ vertices, has 2^k sources and 2^k sinks, and that all vertices that are not sources have indegree two.

Moreover, if H is a graph with n sinks and n sources, we say a graph is a stack of s Hs, if it consists of s copies of H such that sources on level i are identified with sinks on level $i+1$ for $i \in \{1, \dots, s-1\}$.

For any $n, d \in \mathbb{N}$ such that n is a power of 2, the graph $G(n,d)$ we will consider for the Dymond–Tompa game consists of a (possibly fractional) stack of butterfly graphs of dimension $\log n$, with an attached binary tree on top such that the depth of this graph is exactly d (see Figure B.10a). Note that if d is a multiple of $\log n$, then this graph has exactly $d/\log n$ blocks (the 1st block is a binary tree). Moreover, if $d \leq \log n$, then $G(n,d,)$ is just a binary tree of depth d. Observe that, if $d \geq \log n$, $G(n,d)$ has $(d - \log n)n + 2n - 1$ vertices.

B.6.1 Upper Bounds for the Cost of the Dymond–Tompa Game on Butterfly Graphs

Given a graph G, we say T is a Pebbler strategy for G if the following holds.


(a) Stacks of graphs with binary tree on top (dashed lines represent vertex identification)

(b) 3-dimensional butterfly graphs

Figure B.10: Stack of butterflies

1. Each node $x \in V(T)$ is labelled with a set of vertices $S(x) \subset V(G)$ (corresponding to a valid set of vertices where Pebbler can place pebbles at the current stage of the game). We note that in order for $S(x)$ to be a valid move for Pebbler at node x, it must be the case that if P is the path from the root of T to x, then $S(x) \cap (\bigcup_{y \in V(P)} S(y)) = \emptyset$ (Pebbler cannot repebble a vertex) and $S(x) \neq \emptyset$ if any vertex in $\bigcup_{y \in V(P)} S(y)$ has an immediate predecessor that is unpebbled (if the game has not ended, Pebbler must place at least one pebble).

2. Each edge leaving a node $x \in V(T)$ is labelled with a set of vertices $S_{xy} \subseteq S(x) \cup S_{p(x)x}$ of possible Challenger moves (corresponding to pebbles that Challenger challenges and that lead to the same Pebbler strategy), where $p(x)$ is the parent of x in T. In the case where x is the root of the tree, define $S_{p(x)x} = \emptyset$. In order for T to be a complete strategy, i.e., for T to describe how to deal with all possible Challenger moves, it must be the case that at every node x either $\bigcup_{y:xy \in E(T)} S_{xy} = S(x) \cup S_{p(x)x}$ or all immediate predecessors of $S(x) \cup S_{p(x)x}$ are pebbled, and in this latter case x is a leaf.

Proposition B.6.3. *If there is a winning Pebbler strategy tree T with max degree a, depth*

d and such that the label of each node is of size at most b, then the cost of a r-round DT game, for any $r \leq d$, is at most $rb(a^{\lceil d/r \rceil} - 1)$.

Proof. Pebbler's strategy will be as follows: at round i Pebbler will be playing according to the strategy tree T_i. At the first rounds, let $T_1 = T$. Let $r \leq d$. For every $i \leq r$, let T_i' be a subtree of T_i consisting of all nodes at distance less than $\lceil d/r \rceil$ of the root. Note that there are at most $a^{\lceil d/r \rceil} - 1$ nodes in T_i'. At rounds i, Pebbler places pebbles on all the vertices that are in some label of nodes in $V(T_i')$. Since each node has at most b labels, Pebbler places at most $b(a^{\lceil d/r \rceil} - 1)$ pebbles at each rounds. If Challenger challenges a vertex v that does not have all its immediate predecessors pebbled, then v must be in the label of some edge xy where x is a leaf of T_i'. Let T_{i+1} be the subtree of T_i having y as root.

This is a valid strategy for Pebbler and since for every i, the depth of T_{i+1} is $\lceil d/r \rceil$ smaller than the depth of T_i and $r\lceil d/r \rceil \geq d$, the game will end in at most r rounds. Thus the total number of pebbles placed is at most $rb(a^{\lceil d/r \rceil} - 1)$. □

We state an immediate corollary of Proposition B.6.3, which is to weak to imply the upper bounds we stated, but has the advantage that it doesn't depend on strategy trees and might be useful for other purposes.

Corollary B.6.4. *If Pebbler has a winning strategy in d rounds and x pebbles per round, then the cost of a r-round DT game, for any $r \leq d$, is at most $rx((x+1)^{\lceil d/r \rceil} - 1) \leq r(x+1)^{\lceil d/r \rceil + 1}$.*

All that is left to prove the upper bounds is to show that there exists Pebbler strategy trees with certain properties. We prove two propositions below which together with Proposition B.6.3 implies the upper bounds in Lemma B.6.1

Proposition B.6.5. *There is a Pebbler strategy tree T for the graph $G(n,d)$ with max degree 2, depth d and such that the label of each vertex is of size at most 2.*

The proof follows from the following straightforward claim.

Claim B.6.6. *For any graph with depth d and indegree at most δ, there is a winning Pebbler strategy in d rounds and using at most δ pebbles per round. Moreover, this strategy is such that if Challenger stays the game immediately ends.*

Proof. The Pebbler strategy is simply to, at every round, pebble all in-neighbours of the challenged vertex. □

Proposition B.6.7. *There is a winning Pebbler strategy tree T for the graph $G(n,d)$ with max degree 2, depth $\lceil \log d \rceil$ and such that the label of each vertex is of size at most n.*

Proof. The winning Pebbler strategy is to do a binary search in the rows of $G(n,d)$. The strategy only depends on whether Challenger stays or moves, but does not depend on what particular pebble Challenger chooses to pebble. Thus the Pebbler strategy tree T has max degree 2. The proposition then follows from the fact that $G(n,d)$ has depth d and has at most n vertices per row. □

B.6.2 Lower Bounds for the Cost of the Dymond–Tompa Game on Butterfly Graphs

Now we would like to show that the strategies described in the previous subsection are essentially the best Pebbler can do. As a warm up, and to give some intuition on the strategy, we prove a special case of Lemma B.6.1. In order to keep the proof simple, we use the alternative definition of the Dymond–Tompa game and consider a stack of butterflies with an extra vertex on top, $\widehat{G}$, as defined in Section B.5.

Lemma B.6.8. *For any $n, r \in \mathbb{N}^+$ such that n is a power of 2, there exists an explicitly constructible DAG $\widehat{G}(n, r\log n)$ of depth $r\log n + 1$ with $\mathrm{O}(nr\log n)$ vertices and indegree at most 2 such that for any $r \leq d$, the cost of an r-round DT game is at least $\frac{n}{4}$.*

This lemma holds not only for stacks of butterfly graphs, but also for stacks of other kinds of graphs as long as they have a strong version of the grate property [Val77].

Definition B.6.9. A graph with n sources and sinks is a *α-uniform grate* if after removing α vertices, there still are at least $n/2+1$ sources, each of which can reach $n/2+1$ sinks.

Throughout the Dymond–Tompa game, we say a vertex t is reachable from s if there is a path from s to t with no pebbles neither on internal vertices of the path nor on the vertex s (but t may be pebbled). We say a sink at level ℓ is *good* if it is unpebbled and is reachable by at least $n/2+1$ sources at level ℓ. Furthermore, we say a source s is disconnected from a sink t if there is no completely (including end points) unpebbled path from s to t, and we consider the number of source-sink disconnections in a graph as the number of pairs (t,s) such that s is disconnected from t.

Observation B.6.10. *Butterfly graphs are $(n/4-1)$-uniform grates.*

Proof. If there are less than $n/2+1$ good sink-vertices, then the number of source-sink disconnections is at least $n/2 \cdot n/2$ (at least $n/2$ non-good sinks are not reached by at least $n/2$ sources). Note that any vertex in a butterfly graph is in exactly n distinct source-sink paths. So if α is the number of vertices removed, then there are at most αn source-sink disconnections. This implies that $\alpha \geq n/4$. □

We can now proceed to the proof of the warm-up lemma.

Proof of Lemma B.6.8. We give a strategy for Challenger in the Dymond–Tompa game over the graph $\widehat{G}(n, r\log n)$ defined above so that, assuming Pebbler has at most $n/4-1$ pebbles, the game will not end within r rounds.

At a high level, Challenger's strategy will be to keep in mind, before every round, a good sink that can reach the challenged vertex; more precisely, before round $\ell+1$ Challenger will have a good sink at level ℓ in mind, say t_ℓ. After the Pebbler places pebbles on the graph, Challenger chooses a good sink at level $\ell+1$ that is reachable from t_ℓ and decides to have that in mind. He will then check if there are any new pebbles that are causing the challenged vertex to be unreachable from t_ℓ, and if so, challenges one that is reachable from t_ℓ.

The proof goes as follows. We maintain the invariant that before round ℓ, Challenger's chosen vertex t_ℓ is a good sink at level ℓ and reaches the challenged vertex. Before the first rounds, the challenged vertex is the sink of $\widehat{G}$ (the vertex that is not in G) and Challenger's chosen vertex is the original sink of G, t_1, which clearly is a good sink at level 1 (it is unpebbled and reachable from all sources at level 1) and reaches the challenged vertex.

Suppose that the invariant was true until round ℓ, i.e. suppose that before Pebbler's $(\ell-1)$st move, Challenger's chosen vertex $t_{\ell-1}$ is a good sink at level $\ell-1$ and reaches the challenged vertex. At round $\ell-1$, Pebbler places some pebbles. Since Pebbler has at most $n/4-1$ pebbles in total, we can conclude by the uniform grate property that there are at least $n/2+1$ good sinks at level ℓ. Since $t_{\ell-1}$ was a good sink before round $\ell-1$, there must be a good sink at level ℓ, say t_ℓ, which was reachable from $t_{\ell-1}$ before round $\ell-1$. Challenger decides t_ℓ will be the next chosen vertex. Since before round $\ell-1$, t_ℓ reached $t_{\ell-1}$ and $t_{\ell-1}$ reached the challenged vertex, the only possible pebbles that are disconnecting t_ℓ from the challenged vertex are the newly put pebbles. If there are no such blocking pebbles, i.e., if t_ℓ reaches the challenged vertex, Challenger stays. If there are newly put pebbles that block all paths from t_ℓ to the challenged vertex, Challenger challenges one that is reachable from t_ℓ. Thus, before round ℓ, Challenger's chosen vertex t_ℓ is a good sink at level ℓ and reaches the challenged vertex, and the invariant is maintained.

We conclude that before round $(r+1)$, Challenger's chosen vertex t_{r+1} is an unpebbled vertex global source (which would have been a good sink at level $r+1$, if such a level had existed) that reaches the challenged vertex, and hence the game has not ended. □

Now to prove the lower bound in Lemma B.6.1 in its full generality, we must allow any number of rounds (at most the depth) and still get a good bound on the cost of the game. We again describe a strategy for Challenger; the difference is that Challenger cannot afford to jump $\log n$ rows every round. Intuitively, we do not think of the graph as a stack of blocks, but as a continuous block such that any consecutive $\log n$ rows is isomorphic to a butterfly graph.

Note that given any vertex $v \in V(G(n,d))$ at distance d' from the set of sources, the subgraph induced by all vertices that reach v is isomorphic to $G(n,d')$. We therefore refer to the top binary tree of the subgraph $G(n,d')$ as the tree induced by the vertices that reach v and are at distance at most $\log n$ from v.

We give a more general definition of a good vertex and define a partially good vertex. Let T be a complete directed binary tree rooted at v. We say v (or T) is good if v can be reached by strictly more than half of the leaves. If T has n leaves this is equivalent to requiring that, for any $h' \leq \log n$, v can be reached by strictly more than $2^{h'}/2$ vertices at distance h' from v. Given a vertex $u \in V(T)$ at distance h from the leaves, we say u (or the subtree of T rooted at u) is T-partially good if, for any $h' \leq h$, u can be reached by strictly more than $2^{h'}/2$ vertices at distance h' from u. When T is clear from the context, we just say u is partially good.

We are now ready to prove the lower bound.

Proof of Lemma B.6.1, item 2. We actually prove something stronger: we allow Pebbler to place some pebbles before the game begins, provided that the top binary tree remains good. We charge only for the pebbles placed outside the binary tree. Challenger is not allowed to challenge any pebble that was placed in this initial stage. We denote this game DT*.

Formally, we prove the following claim. Given a graph G and a challenged vertex on this graph, if there is a vertex v in G that reaches the challenged vertex and that is the sink of a graph $G(n,d)$, then cost of the r-round DT* game on G is at least $\min\{\frac{\gamma 2^{d/\gamma}}{8}, \frac{n}{8}\}$, where $\gamma = \min\{d, r\}$.

We prove this claim by induction on γ. For $\gamma = 1$, either $d > r = 1$ or $d = 1$. If $d > r = 1$, $G(n,d)$ consists of at least a binary tree of depth $d' = \min\{d, \log n\}$ with $2^{d'+1} - 1$ vertices and such that the sink reaches the challenged vertex. Since after Pebbler places the initial pebbles the binary tree must be a good tree, at least half of the tree reaches the challenged vertex (actually, strictly more than half of the pebbles in every row must reach the challenged vertex, which makes a total of at least $2^{d'} + d'$ vertices that reach the challenged vertex). Clearly Pebbler must pebble all the vertices that reach the challenged vertex in order to finish the game in one round, therefore the cost is more than $\min\{\frac{2^d}{8}, \frac{n}{8}\}$. If $d = 1$, then clearly at least 1 pebble is needed and $1 \geq 1/4 = \gamma 2^{d/\gamma}/8$, so the base case holds.

Now suppose $\gamma \geq 2$ and that Pebbler has placed some initial pebbles on the graph, but maintaining the top binary tree good. Pebbler then starts the first round by placing some pebbles. Let $x \geq 1$ be the number of pebbles Pebbler placed in the top binary tree in the first round (note we are not counting the initial pebbles placed before the game began). If $d \leq \lceil \log 4x \rceil$ (i.e., if the graph is shallow or if Pebbler placed too many pebbles), the claim holds since this implies $x \geq \frac{2^d}{8}$ and clearly $\frac{2^d}{8} \geq \frac{2^{d/\gamma + \log \gamma}}{8} = \frac{\gamma 2^{d/\gamma}}{8}$, for any γ and d that satisfy $2 \leq \gamma \leq d$. We thus assume $d > \lceil \log 4x \rceil$.

Note that the row that is at distance $\lceil \log 4x \rceil$ from the root of the top binary tree has exactly $y = 2^{\lceil \log 4x \rceil} \geq 4x$ vertices. Before the first round, at least $y/2$ of these vertices were partially good (with respect to the top binary tree). Since $x \leq y/4$ pebbles were placed, at least $y/4$ of these partially good trees were untouched at this round. We will show that, provided that there are less than $n/8$ pebbles in the graph, then at least one of these partially good trees is totally good.

Fix a set of $y/4$ partially good trees that were untouched at this round. If $d \leq \log n$, then the partially good trees are all totally good. If $d > \log n$, we consider the set S of all the (pebbled or unpebbled) vertices at distance $\log n$ from the root of the top binary tree that are in one of these $y/4$ partially good trees. Since these trees are disjoint there are at least $y/4 \cdot (n/y) = n/4$ such vertices. Consider the block consisting of vertices at distance at most $\log y$ from S. Note that the number of source-sinks paths in this block is at least $n/4 \cdot y$ and any vertex in this block is in at most y such paths. Therefore, if there are less than $n/8$ pebbles, then there are more than $ny/8$ unpebbled source-sink paths in this block. This means that at least one of the $y/4$ partially good tree has more than $n/2$ unpebbled source-sink paths in this block, which implies that it is a totally good tree.

Let v be the root of this totally good tree. We know that v reached the challenged vertex before this rounds. This implies that if v no longer reaches the challenged vertex then there are newly placed pebbles blocking a path between v and the challenged vertex. If this is the case, Challenger challenges a newly placed pebble that is in such a path and that is closest to v (i.e., is reachable from v). The graph induced by all vertices that reach v satisfies the invariants and has depth $d - \log y \geq d - \log 8x$. We observe that, the x pebbles we account for at this round were placed on the binary tree, and therefore are not counted again when applying the induction hypothesis.

If $r - 1 > d - \log y$ (i.e., the number of rounds left is larger than the depth of the remaining subgraph), we argue that claim follows. To show this, we consider two cases: $r < d/2$ and $d/2 \leq r$. In the first case, we show that the number of pebbles placed has to be large since in one round the depth of the remaining subgraph was reduced by a lot. Note that $r < d/2$ this implies that $d - \log y < d/2 - 1$ and thus $8x \geq y > 2^{d/2+1}$. Moreover, $\gamma = \min\{r, d\} = r < d/2$ and since $d/\gamma + \log \gamma$ is a monotone decreasing function for $\gamma \in [2, d/2]$, we have that $8x > 2^{d/2+1} \geq 2^{d/\gamma + \log \gamma}$ and the claim follows. In the second case, the intuition is that the claim we want to prove is not so strong. Indeed, note that $d/2 \leq r$ implies that $d/2 \leq \gamma = \min\{r, d\} \leq d$, and thus $2^{d/\gamma + \log \gamma} \leq 2d$, so it is enough to show that at least $2d/8$ pebbles are needed. Applying the induction hypothesis on the subgraph induced by all vertices that reach v we have that the cost of this subgraph is at least $\min\{\frac{2(d - \log y)}{8}, \frac{n}{8}\}$. The claim follows by noting that $8x + 2(d - \log y) \geq 2d + 8x - 2\log 8x \geq 2d$, where the last inequality follows since $x \geq 1$.

We thus assume $r - 1 \leq d - \log y$ (which implies $r \leq d$ and $\gamma = r$), and apply the induction hypothesis on the subgraph induced by all vertices that reach v to get that the cost of this subgraph with one less round is at least $\min\{\frac{(r-1)2^{(d - \log 8x)/(r-1)}}{8}, \frac{n}{8}\}$.

It suffices to show that

$$(r-1)2^{(d-\log 8x)/(r-1)} + 8x \geq r2^{d/r}.$$

Let $a = \frac{d}{r} - \frac{d-\log 8x}{r-1}$, so that $8x = 2^{d/r+a(r-1)}$. Rewriting the equation above we have

$$(r-1)2^{(d-\log 8x)/(r-1)} + 8x = (r-1)2^{d/r-a} + 2^{d/r+a(r-1)} = r2^{d/r}\left(\frac{(r-1)2^{-a}}{r} + \frac{2^{a(r-1)}}{r}\right).$$

Note that, for any $r \geq 1$, $(r-1)2^{-a} + 2^{a(r-1)} \geq r$. Indeed, for any fixed $r \geq 1$, a straightforward calculation shows that the real function $f(a) = (r-1)2^{-a} - r + 2^{a(r-1)}$ is minimized at $a = 0$ and $f(0) = 0$. This concludes the proof. □

To conclude this section, we define a more general class of previously studied graphs and prove that Lemma B.6.8 also applies for stacks of any graph in this class.

Definition B.6.11 (Grate). An (α,β)-*grate* is a DAG such that after removing any α vertices, at least β pairs of a source and a sink are connected.

It is straightforward to verify that an $(\alpha, n^2-(n/2+1)^2)$-grate is an α-uniform grate—in fact, this is what we noted in the proof of Observation B.6.10—hence Lemma B.6.8 holds for stacks of grates. In the converse direction, an α-uniform grate is an $(\alpha, (n/2+1)^2)$-grate. By Proposition 6.2 in [Val77], any $(\Omega(n), \Omega(n^{1+\epsilon}))$-grate of logarithmic depth needs to have superlinear size, so butterfly graphs are close to optimal.

Other than butterfly graphs, an example of $\Omega(n)$-uniform grates are the so-called *supergrates* [KS90]. Supergrates are of linear size, but they are too deep for the strategy of Proposition B.6.5 to give meaningful upper bounds. Another example are connector graphs, as we proceed to show. Connector graphs can be shallow but require $\Omega(n \log n)$ edges [PV76].

Definition B.6.12 (Connector). An *n-connector* is a DAG with n sources S and n sinks T, and that satisfies the following property: for any subsets $S' \subseteq S$ of sources and $T' \subseteq T$ of sinks of size $|S'| = |T'|$ and for any specification M of which source in S' should be connected to which sink in T' (one-to-one correspondence), it holds that there are $|S'|$ vertex-disjoint paths between S' and T' satisfying M.

Proposition B.6.13. *An n-connector is an* $(\alpha, n^2 - \alpha n)$*-grate.*

Proof. We prove that an n-connector satisfies the following two properties:

1. any source can reach any sink;

2. the removal of any set of α vertices causes at most αn source-sink disconnections, i.e., the sum over all sinks v of the number of sources that cannot reach v is at most αn.

Let G be a n-connector. Obviously G satisfies property 1. Let A be any set of vertices in G. Let $\alpha = |A|$. We will show that the removal of A causes at most αn source-sink disconnections, thus concluding that G also satisfies property 2. Let G' be the graph that results from G after the removal of A.

Let $H = ((S, T), E)$ be a bipartite graph, where S correspond to the sources in G and T to the sinks, and there is an edge (s, t) if source s doesn't reach sink t in G'. Let q' be the size of a maximum matching in H. This implies that H has a vertex cover of size q' (Kőnig's theorem). Since every vertex in H has degree at most n, we conclude that H has at most $q'n$ edges, which means that A caused at most $q'n$ source-sink disconnections.

Suppose $q' > \alpha$, and let $M = \{(s_1, t_1), (s_2, t_2), \dots, (s_{\alpha+1}, t_{\alpha+1})\}$ be a matching of size $\alpha + 1$ in H. Given the set $S' = \{s_1, s_2, \dots, s_{\alpha+1}\}$ of sources, the set $T' = \{t_1, t_2, \dots, t_{\alpha+1}\}$ of sinks and M as the specification of which source should be connected to which sink, we have that in G there are $\alpha + 1$ disjoint paths connecting S' to T' according to M. But this is a contradiction, since all paths must contain a vertex in A.

Therefore, we conclude that $q' \leq \alpha$ and A caused at most $rn \leq \alpha n$ source-sink disconnections. □

Interestingly, butterfly graphs and connectors also relate in that connecting two k-dimensional butterfly graphs in a certain back-to-back fashion gives a 2^k-connector. A description of this construction and a proof of this fact can be found, e.g., in [Nor19].

B.7 Upper Bounds for Size and Space

We prove the upper bounds in terms of the weaker *resolution* proof system. A resolution configuration $\mathbb{C}$ is a set of clauses. A resolution refutation of a CNF formula F is a sequence of configurations $\mathbb{C}_0, \dots, \mathbb{C}_\tau$ such that $\mathbb{C}_0 = \emptyset$, the empty clause $\bot \in \mathbb{C}_\tau$, and for $t \in [\tau]$ we obtain $\mathbb{C}_t$ from $\mathbb{C}_{t-1}$ by one of the following steps:

Axiom download $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C\}$, for $C \in F$.

Inference $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C \vee D\}$, where $C \vee D$ is inferred by the resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}.$$

Erasure $\mathbb{C}_t = \mathbb{C}_{t-1} \setminus \{C\}$, for some $C \in \mathbb{C}_{t-1}$.

The length of a refutation is the number of axiom downloads and inferences. The line space of a configuration is the number of clauses, and the total space is the number of literals. The (line/total) space of a refutation is the maximum over all configurations.

It is easy to see that cutting planes can simulate the resolution rule using at most w additions and one division, where w is the width of the shortest clause, and therefore a resolution refutation in length L, width w and space s gives a cutting planes refutation in size $O(w^2L)$ and space $s + 1$ where the largest coefficient is 2. The refutation that we construct in Lemma B.7.3 is of constant width, so cutting planes can simulate it with constant overhead, and in Lemma B.7.7 it is not but we can ignore polynomial factors.

The search depth of a formula F is the minimum number of queries of a decision tree for the search problem of F. As observed in [LNNW95, BIW04], a search tree for the falsified clause search problem is equivalent to a tree-like resolution refutation. We can construct a refutation essentially by replacing each internal node labelled with a variable x in the search tree with the result of resolving its two children over the variable x. It is straightforward to check that this is indeed a valid resolution refutation.

Lemma B.7.1 ([ET01]). *If a CNF formula has search depth d, then it has a refutation in length 2^d, width d, and space d simultaneously.*

Proof. Consider the refutation tree T equivalent to a minimal depth search tree. Traversing the refutation tree in depth-first order it is straightforward to reconstruct a tree-like refutation of length $|T| \leq 2^d$, width d, and space d, where $|T|$ is the order of T. □

To show length upper bounds we simulate a black pebbling in resolution and then lift that refutation, as done in for instance [BN11].

The *black pebble game* is played by a single player on a DAG. The allowed moves are to place a pebble on a vertex if its predecessors have pebbles and to remove a pebble from any vertex. A pebbling is a sequence of moves that begin with the empty graph and end with a pebble on the sink. The number of moves of a pebbling is called the time, and the maximum number of pebbles on the graph at the same time the space. An excellent survey of pebbling up to ca 1980 is [Pip80], and some more recent developments are covered in the upcoming survey [Nor19].

Lemma B.7.2. *If there is a black pebbling for an indegree 2 graph G in space s and time τ, then there is a resolution refutation of Peb_G in length $O(\tau)$, width 3, and total space $O(s)$.*

Proof. We build a refutation π of Peb_G by keeping in memory the unit clause v for every vertex v that has a pebble. This is trivial for sources because these clauses are already axioms. For a vertex v with predecessors u_1 and u_2, when we place a pebble over v its predecessors have pebbles, therefore the clauses u_1 and u_2 are in memory. We download the axiom $\overline{u_1} \lor \overline{u_2} \lor v$, resolve it with u_1 and u_2 to obtain the clause v, and then delete intermediate clauses. □

We can use a generic procedure to transform any refutation into a refutation for the corresponding lifted formula (see Lemma 4.3 in the ECCC version of [BN11]). However, we obtain better upper bounds if we take the structure of the refutation into account.

Lemma B.7.3. *Let G be a graph of indegree 2 with a black pebbling in space s and time τ. Then there is a refutation of $Lift_\ell(Peb_G)$ in size $O(\tau \cdot \ell^3)$ and total space $O(s \cdot \ell)$.*

$$\dfrac{\dfrac{\dfrac{\overline{x}_{1,u_1} \vee \overline{y}_{1,u_1} \vee B \qquad \overline{x}_{1,u_1} \vee y_{1,u_1}}{\overline{x}_{1,u_1} \vee B} \qquad x_{1,u_1} \vee s_{2,u_1}}{s_{2,u_1} \vee B} \qquad \dfrac{\dfrac{\overline{x}_{2,u_1} \vee \overline{y}_{2,u_1} \vee B \qquad \overline{x}_{2,u_1} \vee y_{2,u_1}}{\overline{x}_{2,u_1} \vee B} \qquad \overline{s}_{2,u_1} \vee x_{2,u_1} \vee s_{3,u_1}}{\overline{s}_{2,u_1} \vee s_{3,u_1} \vee B}}{s_{3,u_1} \vee B}$$

$$\dfrac{\begin{matrix} \vdots \\ s_{\ell,u_1} \vee B \end{matrix} \qquad \begin{matrix} \vdots \\ \overline{s}_{\ell,u_1} \vee B \end{matrix}}{B}$$

Figure B.11: Simulation of a pebbling step

Proof. Let π be the refutation of Peb_G given by Lemma B.7.2. We build a refutation π' of $Lift(F)$ by deriving, for each unit clause v, the ℓ clauses $Lift(v)$. This is trivial in the axiom download and erasure cases, and we are left with inference. The only inference steps we need to deal with are of the form

$$\dfrac{\dfrac{\overline{u_1} \vee \overline{u_2} \vee v \qquad u_1}{\overline{u_2} \vee v} \qquad u_2}{v} \tag{B.25}$$

and we handle both inference steps at once.

Recall that for a lifted formula to have constant width we have to split the wide auxiliary clauses (B.3a), introducing extension variables, but we were not explicit about how to do that. We split the clause $\bigvee_{a=1}^{\ell} x_{a,u}$ into a clause $x_{1,u} \vee s_{2,u}$, $\ell - 2$ clauses of the form $\overline{s}_{a,u} \vee x_{a+1,u} \vee s_{a,u}$, and a clause $\overline{s}_{\ell,u} \vee x_{\ell,u}$.

First we fix a clause $C \in Lift(v)$ that we want to derive. Then we fix a clause $B \in Lift(\overline{u_2} \vee v)$ that contains C as a subclause. We can derive B by resolving the clauses $\overline{x}_{a,u_1} \vee \overline{y}_{a,u_1} \vee B$, which are actual axioms of $Lift(Peb_G)$, first with $\overline{x}_{a,u_1} \vee y_{a,u_1}$, which are in memory by hypothesis, and then with the axioms $\overline{s}_{a,u_1} \vee x_{a,u_1} \vee s_{a+1,u_1}$ that result of breaking $\bigvee_{a=1}^{\ell} x_{a,u_1}$ into clauses of constant width. See Figure B.11 for details. Such a derivation requires $O(\ell)$ steps and constant space.

We repeat this procedure for all of the ℓ clauses in $B \in Lift(\overline{u_2} \vee v)$ that contain C as a subclause, using at most $O(\ell^2)$ steps and space $\ell + O(1)$. Now we have all the clauses required to derive C by repeating the above procedure with the clauses $Lift(\overline{u_2}) \vee C$ that we just derived, the clauses $\overline{x}_{a,u_1} \vee y_{a,u_1}$, which are also in memory by hypothesis, and the axioms $\overline{s}_{a,u_1} \vee x_{a,u_1} \vee s_{a+1,u_1}$. Such a derivation requires an additional $O(\ell)$ steps and constant additional space, for a total of $O(\ell^2)$ steps and space $\ell + O(1)$. Finally we repeat the whole procedure ℓ times, once for each clause $C \in Lift(v)$, for a total of $O(\ell^3)$ steps and space $2\ell + O(1)$.

Observe that all clauses are of constant width, so the size and total space are also $O(\ell^3)$ and $O(\ell)$, and furthermore we can simulate the resolution proof in cutting planes with constant overhead. □

If we only care about optimizing size, then a strategy that places pebbles in topological order and never removes a pebble is a valid pebbling of any graph of order m in time m and space m, which gives a short refutation in size $O(m\ell^3)$ and space $O(m\ell)$.

Lemma B.7.4. *Let G be a graph of order m and indegree 2. For any $\ell \geq m^3$ there is a refutation of $Lift_\ell(Peb_G)$ in size $O(N)$ and total space $O(N^{2/5})$, where $N = \Theta(m\ell)$ is the size of $Lift_\ell\big(Peb_G\big)$.*

In terms of space, even the most space-efficient pebbling strategy would give a refutation in space $O(\ell)$, which is too weak. Therefore to obtain good space upper bounds we go through the Dymond–Tompa game and search depth instead of black pebbling. The following Lemma follows from Lemma B.5.1 and was first proved in [Cha13].

Lemma B.7.5 ([Cha13]). *If there is a Dymond–Tompa pebbling strategy for a graph G in space s, then the formula Peb_G has search depth s.*

If we lay out the extension variables so that their indices form an ordered binary tree and attach two nodes labelled $x_{a,u}$ and $x_{a+1,u}$ to each leaf $s_{a,u}$ we get a search tree that finds a selector variable set to true by any assignment that respects auxiliary clauses. We can use this tree to build search trees for a lifted formula.

Lemma B.7.6. *Given a CNF formula F of search depth d, the lifted formula $Lift_\ell(F)$ has search depth $d \log \ell$.*

Proof. Given a decision tree T_1 for the falsified clause search problem on F of depth d and a decision tree T_2 that finds a selector variable set to true of depth $\log \ell$, we build a decision tree T_3 for the falsified clause search problem on $Lift_\ell(F)$ of depth $d \log \ell$ by composing the trees as follows.

First we modify T_2. We reinterpret the leaves as queries to selector variables $x_{a,u}$, and we attach two new nodes to every selector variable query. We label the 0-leaf of $x_{a,u}$ with the falsified clause $\overline{s}_{a,u} \vee x_{a,u} \vee s_{a+1,u}$, and we label the 1-node with the main variable $y_{a,u}$. We add two unlabelled leaves to the $y_{a,u}$ node.

Then, starting at the root of T_1, we apply the following recursive procedure. If the root is an inner vertex labelled with a variable u, then we add a copy of T_2 that queries variables corresponding to u. To each 0-leaf we attach the result of this procedure on the 0-subtree of T_1, and to each 1-leaf we attach the result of this procedure on the 1-subtree.

Finally, for each leaf of T_3 that we did not label yet, there is a corresponding leaf in T_1 labelled with a clause C. C is falsified by the assignment α induced by the branch leading to C. By construction, the assignment β induced by the branch in T_3 respects auxiliary clauses and, for every variable $u \in Vars(C)$ it sets $x_{a,u} = 1$ and $y_{a,u} = \alpha(u)$ for some $a \in [\ell]$. Therefore we can label the leaf of T_3 with the main clause $\bigvee_{u \in Vars(C)} \overline{x}_{a,u} \vee y_{a,u}^{1-\alpha(u)}$. □

Lemma B.7.7. *Let G be a graph of order m and indegree 2 with Dymond–Tompa price s. For any $\ell \geq m^3$ there is a refutation of $Lift_\ell(Peb_G)$ in size $2^{O(s\log N)}$ and space $O(s\log N)$, where $N = \Theta(m\ell^3)$ is the size of $Lift_\ell(Peb_G)$.*

Proof. This follows immediately from Lemmas B.7.5, B.7.6 and B.7.1. □

B.8 Putting the Pieces Together

By the technical result proved in Section B.2, Theorem B.2.8, we know that if G is a graph over m vertices such that the r-round Dymond–Tompa game on G costs $\Omega(c)$, then for $\ell = m^{3+\epsilon}$, $Lift_\ell(Peb_G)$ is a 6-CNF formula over $\Theta(m^{4+\epsilon})$ variables and $N = \Theta(m^{10+3\epsilon})$ clauses such that for any CP refutation of $Lift_\ell(Peb_G)$ even with coefficients of unbounded size in formula space less than $\frac{c}{r}\log N$ requires length greater than $2^{\Omega(r)}$. This fact together with the lower and upper bounds proven in Sections B.6 and B.7 yield the following theorem.

Theorem B.8.1. *There is an explicitly constructible two-parameter family of unsatisfiable 6-CNF formulas $F(n,d)$, for $n,d \in \mathbb{N}^+$, of size $N = \Theta((dn)^{10+\epsilon})$ such that:*

1. *$F(n,d)$ can be refuted by CP with small coefficients in size $O(N)$ and total space $O(N^{2/5})$.*
2. *$F(n,d)$ can be refuted by CP with small coefficients in total space $O(d\log N)$ and size $2^{O(d\log N)}$.*
3. *For any $r \leq d$, any CP refutation even with coefficients of unbounded size of $F(n,d)$ in formula space less than $\min\{\frac{2^{d/r}\log N}{8}, \frac{n\log N}{8r}\}$ requires length greater than $2^{\Omega(r)}$.*

Proof. Let G be a stack of depth d of butterfly graphs of dimension $\log n$ which has a total of $\Theta(dn)$ vertices. Let $F(n,d) = Lift_\ell(Peb_G)$.

Item 1 follows directly from Lemma B.7.4. Item 2 follows from setting $r = d$ in the upper bound stated in part 1 of Lemma B.6.1 and combining it with Lemma B.7.7.

By Lemma B.6.1 we get that for any $r \leq d$, the r-round Dymond–Tompa game played on G has cost at least at least $\min\{\frac{r2^{d/r}}{8}, \frac{n}{8}\}$. Thus, by Theorem B.2.8, we get item 3. □

Choosing the right values for d and r in Theorem B.8.1, we get the following to corollaries. These are generalizations of Theorems B.1.1 and B.1.2.

Corollary B.8.2. *For any positive constant K, there exists a family of 6-CNF formulas $\{F_N\}_{N=1}^{\infty}$ of size $\Theta(N)$ such that:*

1. *F_N can be refuted by CP with small coefficients in size $O(N)$ and total space $O(N^{2/5})$.*

2. *F_N can be refuted by CP with small coefficients in total space $\mathrm{O}(\log^{K+2} N)$ and size $2^{\mathrm{O}(\log^{K+2} N)}$.*

3. *Any CP refutation even with coefficients of unbounded size of F_N in formula space less than $N^{1/10-\epsilon}$ requires length greater than $2^{\Omega(\log^K N)}$, for any constant $\epsilon > 0$.*

Proof. The proof follows from setting $d = \log^{K+1} n$ and $r = d/\log n$ in Theorem B.8.1 and for every N, choosing n to be a power of 2 such that N is at most a factor off from $(nd)^{10+\epsilon}$.

We note that $\log N = \Theta(\log n)$, so 2 holds. Moreover, $N = \mathrm{o}((nd)^{10+2\epsilon})$, thus $N^{1/10-\epsilon} = \mathrm{o}((nd)^{(10+2\epsilon)(1/10-\epsilon)})$ and

$$\begin{aligned}(nd)^{(10+2\epsilon)(1/10-\epsilon)} &= (n\log^{K+1} n)^{(10+2\epsilon)(1/10-\epsilon)} \\ &\le (n\cdot n^{\epsilon})^{(1-9\epsilon)} < n^{(1-8\epsilon)} \\ &< \frac{n}{8\log^K n} < \frac{n}{8r}\log N,\end{aligned}$$

and therefore 3 also holds. □

Corollary B.8.3. *For any positive constant K, there exists a family of 6-CNF formulas $\{F_N\}_{N=1}^{\infty}$ of size $\Theta(N)$ such that:*

1. *F_N can be refuted by CP with small coefficients in size $\mathrm{O}(N)$ and total space $\mathrm{O}(N^{2/5})$.*

2. *F_N can be refuted by CP with small coefficients in total space $\mathrm{O}\big(N^{\frac{1}{10(K+1)}}\big)$ and size $2^{\mathrm{O}\big(N^{\frac{1}{10(K+1)}}\big)}$.*

3. *Any CP refutation even with coefficients of unbounded size of F_N in formula space less than $N^{\frac{K-1}{10(K+1)}-\epsilon}$ requires length greater than $2^{\Omega\big(N^{\frac{1}{10(K+1)}}\big)}$, for any constant $\epsilon > 0$.*

Proof. The proof follows from setting $d = n^{1/K}\log n$ and $r = d/\log n$ in Theorem B.8.1 and for every N, choosing n to be a power of 2 such that N is at most a factor off from $(nd)^{10+\epsilon}$.

We note that $N^{\frac{1}{10(K+1)}} = \Theta((nd)^{\frac{10+\epsilon}{10(K+1)}})$ and $(nd)^{\frac{10+\epsilon}{10(K+1)}} > d\log N$, hence 2 holds. Moreover, $N = \mathrm{o}((nd)^{10+2\epsilon})$, thus $N^{\frac{K-1}{10(K+1)}-\epsilon} = \mathrm{o}((nd)^{(10+2\epsilon)(\frac{K-1}{10(K+1)}-\epsilon)})$ and

$$\begin{aligned}(nd)^{(10+2\epsilon)(\frac{K-1}{10(K+1)}-\epsilon)} &= (n^{(K+1)/K}\log n)^{(10+2\epsilon)(\frac{K-1}{10(K+1)}-\epsilon)} \\ &< n^{(10+2\epsilon)(\frac{K-1}{10K}-\epsilon)}\cdot n^{\epsilon} < \frac{n^{(K-1)/K}}{8} \\ &< \frac{n^{(K-1)/K}}{8}\log N = \frac{n}{8r}\log N,\end{aligned}$$

and, therefore 3 also holds. □

B.9 Exponential Separation of the Monotone AC Hierarchy

Unsurprisingly, we follow the same approach as [RM99]. Our function is a restriction of the GEN function, except that instead of restricting the valid instances to pyramid graphs, which are unconditionally hard, we restrict the valid instances to the graphs from Section B.6 that exhibit round-space trade-offs. We then use our round-aware simulation theorem to lift the trade-off to communication complexity and the Karchmer–Wigderson game to translate it to a trade-off for monotone circuits.

Definition B.9.1. The *Karchmer–Wigderson game* [KW90] is the following communication problem: given a monotone function f, Alice gets an input x such that $f(x) = 1$ and Bob gets an input y such that $f(y) = 0$. Their task is to compute a coordinate i such that $x_i = 1$ and $y_i = 0$

Theorem B.9.2 ([KW90]). *If there is a monotone circuit for f of fan-in 2^c and depth r, then there is a protocol for the Karchmer–Wigderson game of communication rc and r rounds.*

Proof. The proof is a simple induction on the depth of the circuit. If the circuit has no gates, the players just return the index of the output variable. Otherwise, assume the output gate is an OR-gate, i.e., $f = \bigvee g_i$. Then $g_i(y) = 0$ for all i and there exists i such that $g_i(x) = 1$. Alice sends i with cost at most c and we apply the induction hypothesis on a circuit of one less level. If the output gate is an AND-gate, Bob acts analogously. □

Informally, the G-GEN function computes whether a subset of the lifted pebbling formula on a graph G is unsatisfiable given the indicator vector of such subset. It can be further generalized to CSP-SAT as in [GP18], but we give a definition specialized to pebbling formulas that already suggests a circuit to compute it.

Definition B.9.3. Given a graph G of indegree 2 and $\ell \in \mathbb{N}$, the G-GEN Boolean function is defined as follows. There is a variable (v, a) for every source $v \in G$ and index $a \in [\ell]$. There is a variable $(v \vee \overline{u_1} \vee \overline{u_2}, a, b, c)$ for every non-source vertex $v \in G$ with predecessors u_1 and u_2 and triple $(a, b, c) \in [\ell]^3$. There is a variable $(\overline{z}, a)$ for every index $a \in [\ell]$. A pair (v, a) is reachable if v is a source and (v, a) is 1, or there exist (b, c) such that $(v \vee \overline{u_1} \vee \overline{u_2}, a, b, c)$ is 1, (u_1, b) is reachable, and (u_2, c) is reachable. The value of G-GEN is 1 if there exists some index $a \in [\ell]$ such that (z, a) is reachable and $(\overline{z}, a)$ is 1.

Lemma B.9.4. *There is a monotone circuit that computes G-GEN in depth $2d$, fan-in ℓ^2, and size $O(m\ell^3)$, where d is the depth of G.*

Proof. The circuit computes whether each of the pairs (v, a) is reachable. For v a source we just have a variable. For a non-source v, we have, for each pair $(b, c) \in [\ell]^2$, an AND-gate of fan-in 3 and inputs the gate that computes (u_1, b), the gate that computes

(u_2, c), and the variable $(v \vee \overline{u_1} \vee \overline{u_2}, a, b, c)$, and one OR-gate of fan-in ℓ^2 with inputs all of these gates. Finally, we have ℓ AND-gates with inputs the gate that computes (z, a) and the variable $(\overline{z}, a)$, and an OR-gate of fan-in ℓ. □

It remains to give a reduction from the Karchmer–Wigderson game on G-GEN to the communication game for which we proved lower bounds, and this follows straightforwardly from the corresponding reduction in [RM99].

Lemma B.9.5. *If there is a deterministic communication protocol for the Karchmer–Wigderson game on G-GEN of communication c and r rounds, then there is a deterministic communication protocol for* $\mathit{Lift}(\mathit{Search}(\mathit{Peb}_G))$ *of communication c and r rounds.*

Proof. Let (x, y) be an input to assignment to $\mathit{Lift}(\mathit{Search}(\mathit{Peb}_G))$, this is a vector of indices and a vector of binary strings such that $\mathsf{select}(x_v, y_v)$ is an assignment to a variable v of Peb_G.

Alice builds a 1-input for the Karchmer–Wigderson game on G-GEN as follows. For every source $v \in G$, Alice sets the variable (v, x_v) to 1, and for every non-source vertex $v \in G$, Alice sets the variable $(v \vee \overline{u_1} \vee \overline{u_2}, x_v, x_{u_1}, x_{u_2})$ to 1. Alice sets $(\overline{z}, x_z)$ to 1. The remaining variables are 0.

Bob builds an input as follows. For every source $v \in G$, Bob sets the variable (v, a) to $(y_v)_a$, and for every non-source vertex $v \in G$, Bob sets the variable $(v \vee \overline{u_1} \vee \overline{u_2}, a, b, c)$ to 0 if $(y_v)_a = 0$, $(y_{u_1})_b = 1$, and $(y_{u_2})_c = 1$, and to 1 otherwise. Bob sets $(\overline{z}, a)$ to $1 - (y_z)_a$. Observe that, in Bob's input, if a pair (v, a) is reachable, then $(y_v)_a = 1$. For the sink, this means that if (z, a) is reachable then $(\overline{z}, a) = 1 - (y_z)_a = 0$, so the input evaluates to 0.

Both players then simulate the protocol for the Karchmer–Wigderson game and they get a variable that Alice set to 1 and Bob set to 0. If it is (v, x_v), then $(y_v)_{x_v} = 0$, so axiom v is falsified. If it is $(v \vee \overline{u_1} \vee \overline{u_2}, x_v, x_{u_1}, x_{u_2})$, then $(y_v)_{x_v} = 0$, $(y_{u_1})_{x_{u_1}} = 1$, and $(y_{u_1})_{x_{u_1}} = 1$, so axiom $v \vee \overline{u_1} \vee \overline{u_2}$ is falsified. If it is (t, x_v), then $(y_z)_{x_v} = 1$, so axiom $\overline{z}$ is falsified. □

We have all the ingredients to prove the two main theorems of this section.

Theorem B.1.3 (Restated). *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth* $\log^i n$*, fan-in* 2*, and size* $\mathrm{O}(n)$*, but for which every monotone circuit of depth* $\mathrm{O}(\log^{i-1} n)$ *requires superpolynomial size.*

Proof. Let G be a stack of $\log^i n / 80 i^2 \log\log^2 n$ butterflies with $w = \log^{2i} n$ sources and sinks. This is a graph of depth $d = \log^i n / 40 i \log\log n$ and size $m < \log^{3i} n$, so we can set $\ell = m^{3+\epsilon} < \log^{10i} n$. By Lemma B.9.4 there is a monotone circuit of depth $2d$, fan-in ℓ^2, and size $\mathrm{O}(m\ell^3)$ that computes G-GEN, which we can expand into a circuit of depth $4d \log \ell < \log^i n$, fan-in 2, and size $\mathrm{O}(m\ell^4) = \mathrm{O}(\log^{43i} n)$. By Theorem B.9.2, Lemma B.9.5, Theorem B.4.1, Lemma B.2.6, and Lemma B.6.1, any circuit of depth at most $q \log^{i-1} n$ that computes G-GEN requires size

$$2^{\Omega((w \log \ell)/(q \log^{i-1} n))} = 2^{\Omega(\log^{i+1} n)} \ , \tag{B.26}$$

that is superpolynomial. □

The function above only depends on a polylogarithmic number of variables, with the remaining being padding, so that we can use a small enough gadget to keep the function in NC^i. This implies that we can only obtain a superpolynomial lower bound, but if we are willing to forfeit the function belonging in NC^i then we can achieve an exponential lower bound.

Theorem B.1.4 (Restated). *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth $\log^i n$, fan-in $n^{4/5}$, and size $O(n)$, but for which every monotone circuit of depth $q \log^{i-1} n$ requires size $2^{\Omega(n^{1/(10+4\epsilon)q})}$.*

Proof. Let G be a stack of $\log^{i-1} n$ butterflies with $w = (n^{1/(10+3\epsilon)})/(\log^{i-1} n)$ sources and sinks. This is a graph of depth $d \le \log^i n/(10 + 3\epsilon)$ and size $m \le n^{1/(10+3\epsilon)}$, so we can set $\ell = m^{3+\epsilon} \le n^{(3+\epsilon)/(10+3\epsilon)}$, and the number of variables of G-GEN is indeed at most $m\ell^3 \le n$. By Lemma B.9.4 there is a monotone circuit of depth $2d \le \log^i n$, fan-in $\ell^2 \le n^{4/5}$, and size $O(n)$ that computes G-GEN. By Theorem B.9.2, Lemma B.9.5, Theorem B.4.1, Lemma B.2.6, and Lemma B.6.1, any circuit of depth at most $q \log^{i-1} n$ that computes G-GEN requires size

$$2^{(w^{1/q} \log \ell)/(4r)} = 2^{\Omega(n^{1/(10+4\epsilon)q})} \tag{B.27}$$

as we wanted to show. □

A simulation theorem with a smaller gadget would allow us to obtain a stronger separation between $\mathsf{monotone\text{-}AC}^{i-1}$ and $\mathsf{monotone\text{-}NC}^i$, but we leave that as an open problem.

B.10 Concluding Remarks

In this paper we report the first true size-space trade-offs for cutting planes, exhibiting CNF formulas which have small-size and small-space proofs with constant-size coefficients but for which any short proofs must use a lot of memory, even when using exponentially large coefficients and even when we measure just the number of lines (i.e., inequalities) rather than total size. Furthermore, these results also hold for resolution and polynomial calculus, and are thus the first trade-offs to uniformly capture the proof systems underlying the currently best SAT solvers.

The main technical component in our proof is a reduction to communication complexity as in [HN12, GP18], but with the crucial difference that we reduce to round-efficient protocols in the real communication model of [Kra98]. Extending the techniques in [RM99, GPW15, BEGJ00] to this more general setting, and combining them with new trade-off results for Dymond–Tompa pebbling [DT85], yields our results. Using the same approach we are also able to obtain an exponential separation between $\mathsf{monotone\text{-}AC}^{i-1}$ and $\mathsf{monotone\text{-}AC}^{i}$, improving on the superpolynomial separation in [RM99].

An interesting challenge would be to extend our reduction to stronger communication models such as two-party randomized or multi-party real communication, which would yield trade-offs for stronger proof systems. A recent result in this direction is [GLM$^+$16], but unfortunately it seems hard to incorporate round-efficiency in this framework.

Another question concerns the size of the lifting gadgets we need to construct formulas exhibiting trade-offs. Our gadgets have large polynomial size, which incurs a substantial loss in the results. It would be nice to construct constant-size gadgets, which could lead to tighter trade-off results.

Many proof complexity trade-offs have been obtained by reducing to the *black-white pebble game* [CS76], but in this paper we use the Dymond–Tompa game. It would be desirable to obtain a better understanding of the role of these games and what kind of trade-offs can be obtained from them.

Finally, from a proof complexity perspective we have very few examples of formula families that exhibit size-space trade-offs. Apart from the pebbling formulas studied in this work, the only natural examples[4] are the Tseitin contradictions over long, narrow grids in [BBI16, BNT13]. It would be interesting to prove size-space trade-offs for the latter formulas also in cutting planes, or to find other formulas with size-space trade-offs for this or other proof systems.

Acknowledgements

The authors wish to acknowledge Mladen Mikša, who participated in the initial stages of this work and has kept contributing helpful remarks throughout the project, and Massimo Lauria, with whom we have had many fruitful discussions on time-space trade-offs and other topics in proof complexity. We want to thank Siu Man Chan for introducing us to the wonderful world of Dymond–Tompa pebbling. Different subsets of the authors are grateful for detailed and very helpful discussions on communication complexity with Joshua Brody, Arkadev Chattopadhyay, Prahladh Harsha, Johan Håstad, Troy Lee, Jaikumar Radakrishnan, and Anup Rao. Finally, we are thankful to Dieter van Melkebeek and Ryan Williams for help with references for general SAT time-space trade-offs.

[4]Ignoring trade-offs obtained in [Nor09b] by gluing together disjoint copies of unrelated formulas.

The authors were funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement number 279611 as well as by the Swedish Research Council grant 621-2012-5645.

C

Paper C

Lifting with Simple Gadgets and Applications to Circuit and Proof Complexity

Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals

Abstract

We significantly strengthen and generalize the theorem lifting Nullstellensatz degree to monotone span program size by Pitassi and Robere (2018) so that it works for any gadget with high enough rank, in particular, for useful gadgets such as *equality* and *greater-than*. We apply our generalized theorem to solve two open problems:

- We present the first result that demonstrates a separation in proof power for cutting planes with unbounded versus polynomially bounded coefficients. Specifically, we exhibit CNF formulas that can be refuted in quadratic length and constant line space in cutting planes with unbounded coefficients, but for which there are no refutations in subexponential length and subpolynomial line space if coefficients are restricted to be of polynomial magnitude.
- We give the first explicit separation between monotone Boolean formulas and monotone real formulas. Specifically, we give an explicit family of functions that can be computed with monotone real formulas of nearly linear size but require monotone Boolean formulas of exponential size. Previously only a non-explicit separation was known.

An important technical ingredient, which may be of independent interest, is that we show that the Nullstellensatz degree of refuting the pebbling formula over a DAG G over any field coincides exactly with the reversible pebbling price of G. In particular, this implies that the standard decision tree complexity and the parity decision tree complexity of the corresponding falsified clause search problem are equal.

C.1 Introduction

Lifting theorems in complexity theory are a method of transferring lower bounds in a weak computational model into lower bounds for a more powerful computational model, via function composition. There has been an explosion of lifting theorems in the last ten years, essentially reducing communication lower bounds to query complexity lower bounds.

Early papers that establish lifting theorems include Raz and McKenzie's separation of the monotone NC hierarchy [RM99] (by lifting decision tree complexity to deterministic communication complexity), and Sherstov's pattern matrix method [She11] which lifts (approximate) polynomial degree to (approximate) matrix rank. Recent work has established query-to-communication lifting theorems in a variety of models, leading to the resolution of many longstanding open problems in many areas of computer science. Some examples include the resolution of open questions in communication complexity [GPW15, GLM+16, GKPW17, GJPW17, GPW18], monotone complexity [RPRC16, PR17, PR18], proof complexity [HN12, GP18, dRNV16, GGKS18], extension complexity of linear and semidefinite programs [KMR17, GJW18, LRS15], data structures [CKLM18] and finite model theory [BN16].

Lifting theorems have the following form: given functions $f : \{0,1\}^n \to \{0,1\}$ (the "outer function") and $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ (the "gadget"), a lower bound for f in a weak computational model implies a lower bound on $f \circ g^n$ in a stronger computational model. The most desirable lifting theorems are the most general ones. First, it should hold for *any* outer function, and ideally f should be allowed to be a partial function or a relation (i.e., a search problem). Indeed, nearly all of the applications mentioned above require lifting where the outer function is a relation or a partial function. Secondly, it is often desirable that the gadget is as small as possible. The most general lifting theorems established so far, for example lifting theorems for deterministic and randomized communication complexity, require at least logarithmically-sized gadgets; if these theorems could be improved generically to hold for constant-sized gadgets then many of the current theorems would be vastly improved. Some notable examples where constant-sized gadgets are possible include Sherstov's degree-to-rank lifting [She11], critical block-sensitivity lifting [GP18, HN12], and lifting for monotone span programs [PR17, PR18, Rob18].

C.1.1 A New Lifting Theorem

In this paper, we generalize a lifting theorem of Pitassi and Robere [PR18] to use any gadget that has nontrivial rank. This theorem takes a search problem associated with an unsatisfiable CNF, and lifts a lower bound on the Nullstellensatz degree of the CNF to a lower bound on a related communication problem.

More specifically, let $\mathcal{C}$ be an unsatisfiable k-CNF formula. The search problem associated with $\mathcal{C}$, $\mathsf{Search}(\mathcal{C})$, takes as input an assignment to the underlying variables, and outputs a clause that is falsified by the assignments. [PR18] prove that for any unsatisfiable $\mathcal{C}$, and for a sufficiently rich gadget g, deterministic communication complexity lower bounds for the composed search problem $\mathsf{Search}(\mathcal{C}) \circ g^n$ follow from Nullstellensatz degree lower bounds for $\mathcal{C}$.[1] We significantly improve this lifting theorem so that it holds for *any* gadget of large enough rank.

Theorem C.1.1. *Let $\mathcal{C}$ be a CNF over n variables, let $\mathbb{F}$ be any field, and let g be any gadget of rank at least r. Then the deterministic communication complexity of* $\mathsf{Search}(\mathcal{C} \circ g^n)$ *is at least* $\mathsf{NS}_{\mathbb{F}}(\mathcal{C})$*, the Nullstellensatz degree of $\mathcal{C}$, as long as $r \geq cn/\mathsf{NS}_{\mathbb{F}}(\mathcal{C})$ for some large enough constant c.*

An important special case of our generalized theorem is when the gadget g is the equality function. In this work, we apply our theorem to resolve two open problems in proof complexity and circuit complexity. Both solutions depend crucially on the ability to use the equality gadget.

We note that lifting with the equality gadget has recently been the focus of another paper. Loff and Mukhopadhyay [LM19] observed that a lifting theorem for *total functions* with the equality gadget can be proven using a rank argument. Surprisingly, they also observed that it is *not* possible to lift query complexity to communication complexity for arbitrary relations! Concretely, [LM19] give an example of a relation with linear query complexity but whose composition with equality has only polylogarithmic communication complexity. Nonetheless, they are able to prove a lifting theorem for general relations using the equality gadget by replacing standard query complexity with a stronger complexity measure (namely, the 0-query complexity of the relation).

Unfortunately, we cannot use either of the lifting theorems of [LM19] for our applications. Specifically, in our applications we lift a search problem (and therefore cannot use their result for total functions), and this search problem has small 0-query complexity (and therefore we cannot use their lifting theorem for general relations). Indeed, this shows that our lifting theorem is incomparable to the results of [LM19], even when specialized to the equality gadget. We note that our theorem, too, bypasses the impossibility result of [LM19] by using a stronger complexity measure, which in our case is the Nullstellensatz degree.

C.1.2 A Separation in Proof Complexity

The main application of our lifting theorem is the first separation in proof complexity between cutting planes proofs with high-weight versus low-weight coefficients. The cutting planes proof-system is a proof system that can be used to refute an unsatisfiable CNF by translating it into a system of integer inequalities and showing that this system has no integer solution. The latter is achieved by a sequence of steps that derive new integer inequalities from old ones, until we derive the inequality $0 \geq 1$ (which clearly has no solution). The efficiency of such a refutation is measured by its length (i.e., the

[1]In fact the result is quite a bit stronger—it applies to Razborov's rank measure [Raz90], which is a strict strengthening of deterministic communication complexity.

number of steps) and its space (i.e., the maximal number of inequalities that have to be stored simultaneously during the derivation).

The standard variant of the cutting planes proof system, commonly denoted by CP, allows the inequalities to use coefficients of arbitrary size. However, it is also interesting to consider the variant in which the coefficients are polynomially bounded, which is commonly denoted by CP*. This gives rise to the natural question of the relative power of CP vs. CP*: are they polynomially equivalent or is there a super-polynomial length separation? This question appeared in [BC96] and remains stubbornly open to date. In this work we finally make progress by exhibiting a setting in which unbounded coefficients afford an exponential increase in proof power.

Theorem C.1.2. *There is a family of CNF formulas of size N that have cutting planes refutations of length $\tilde{O}(N^2)$ and space $O(1)$, but for which any refutation in length L and space s with polynomially bounded coefficients must satisfy $s \log L = \tilde{\Omega}(N)$.*

Our result is the first result in proof complexity demonstrating any situation where high-weight coefficients are more powerful than low-weight coefficients. In comparison, for computing Boolean functions, the relative power of high-weight and low-weight linear threshold functions has been understood for a long time. The greater-than function can be computed by high-weight threshold functions, but not by low-weight threshold functions, and weights of length polynomial in n suffice [Mur71] for Boolean functions. For higher depth threshold formulas, it is known that depth-d threshold formulas of high-weight can efficiently be computed by depth-$(d+1)$ threshold formulas of low-weight [GHR92].

In contrast to our near-complete knowledge of high versus low weights for functions, almost nothing is known about the relative power of high versus low weights in the context of proof complexity. Buss and Clote [BC96], building on work by Cook, Coullard, and Turán [CCT87], proved an analog of Muroga's result for cutting planes, showing that weights of length polynomial in the length of the proof suffice. Quite remarkably, this result is not known to hold for other linear threshold proof systems: there is *no* nontrivial upper bound on the weights for more general linear threshold propositional proof systems (such as *stabbing planes* [BFI$^+$18], and Krajíek's threshold logic proof system [Kra95b] where one can additionally branch on linear threshold formulas). Prior to our result, there was no separation between high and low weights, for any linear threshold proof system.

C.1.3 A Separation in Circuit Complexity

A second application of our lifting theorem relates to monotone real circuits, which were introduced by Pudlák [Pud97]. A monotone real circuit is a generalization of monotone Boolean circuits where each gate is allowed to compute any non-decreasing real function of its inputs, but the inputs and output of the circuit are Boolean. A formula is a tree-like circuit, that is, every gate has fan-out one. The first (exponential) lower bound for monotone real circuits was proven already in [Pud97] by extending the lower bound for computing the clique-colouring function with monotone Boolean circuits [Raz85, AB87]. This lower bound, together with a generalization of the interpolation technique [Kra97] which applied only to CP*, was used by Pudlák to obtain the first exponential lower bounds for CP.

Shortly after monotone real circuits were introduced, there was an interest in understanding the power of monotone real computation in comparison to monotone Boolean computation. By extending techniques in [RM99], Bonet et al. prove that there are functions with polynomial size monotone Boolean circuits that require monotone real formulas of exponential size [Joh98, BEGJ00]. This illustrates the power of DAG-like computations in comparison to tree-like. In the other direction, we would like to know whether monotone real circuits are exponentially stronger than monotone Boolean circuits. Rosenbloom [Ros97] presented an elegant, simple proof that monotone real formulas are exponentially stronger than (even non-monotone) Boolean circuits, since slice functions can be computed by linear-size monotone real formulas, whereas by a counting argument we know that most slice functions require exponential size Boolean circuits.

The question of finding explicit functions that demonstrate that monotone real circuits are stronger than general Boolean circuits is much more challenging since it involves proving explicit lower bounds for Boolean circuits—a task that seems currently completely out of reach. A more tractable problem is that of finding explicit functions showing that monotone real circuits or formulas are stronger than *monotone* Boolean circuits or formulas, but prior to this work, no such separation was known either. We provide an explicit separation for *monotone formulas*, that is, we provide a family of explicit functions that can be computed with monotone real formulas of near-linear size but require exponential monotone Boolean formulas. This is the first explicit example that illustrates the strength of monotone real computation.

Theorem C.1.3. *There is an explicit family of functions f_n over $\mathrm{O}(n \operatorname{polylog} n)$ variables that can be computed by monotone real formulas of size $\mathrm{O}(n \operatorname{polylog} n)$ but for which every monotone Boolean formula requires size $2^{\Omega(n/\log n)}$.*

Another motivation for studying lifting theorems with simple gadgets, and in particular the equality gadget, are connections with proving *non-monotone* formula size lower bounds. As noted earlier, lifting theorems have been extremely successful in proving monotone circuit lower bounds, and it has also been shown to be useful in some computational settings that are only “partially” monotone; notably monotone span programs [RPRC16, PR17, PR18] and extended formulations [GJW18, KMR17].

This raises the question of to what extent lifting techniques can help prove *non-monotone* lower bounds. The beautiful work by Karchmer, Raz and Wigderson [KRW95] initiated such an approach for separating P from NC^1—this opened up a line of research popularly known as the *KRW conjecture*. Intriguingly, steps towards resolving the KRW conjecture are closely connected to proving lifting theorems for the equality gadget. The first major progress was made in [EIRS01] where lower bounds for the universal relation game are proven, which is an important special case of the KRW conjecture. Their result was recently improved in several papers [GMWW17, HW93, KM18], and Dinur and Meir [DM18] gave a new top-down proof of the state-of-the-art $\Omega(n^3)$ formula-size lower bounds via the KRW approach.

The connection to lifting using the equality gadget is obtained by observing that the KRW conjecture involves communication problems in which Alice and Bob are looking for a bit on which they differ—this is exactly an *equality* problem. Close examination of the results in [EIRS01, HW93] show that they are equivalent to proving lower bounds for the search problem associated with the pebbling formula when lifted with a 1-bit equality gadget on a *particular* graph [Pit16]. Our proof of Theorem C.4.1 actually establishes near-optimal lower bounds on the communication complexity of the pebbling formula lifted with equality for *any* graph, but where the size of the equality is not 1. Thus if our main theorem could be improved with one-bit equality gadgets this would imply the results of [EIRS01, HW93] as a direct corollary and with significantly better parameters.

C.1.4 Overview of Techniques

We conclude this section by giving a brief overview of our techniques, also trying to convey some of the simplicity of the proofs which we believe is an extra virtue of these results.

Lifting theorem In order to prove their lifting theorem, Pitassi and Robere [PR18] defined a notion of a "good" gadget. They then showed that if we compose a polynomial p with a good gadget g, the rank of the resulting matrix $p \circ g^n$ is determined *exactly* by the non-zero coefficients of p and the rank of g. Their lifting theorem follows by using this correspondence to obtain bounds on the ranks of certain matrices, which in turn yield the required communication complexity lower bound.

In this work, we observe that every gadget g can be turned into a good gadget using a simple transformation. This observation allows us to get an approximate bound on the rank of $p \circ g^n$ for any g with nontrivial rank. While the correspondence we get in this way is only an approximation and not an exact correspondence as in [PR18], it turns out that this approximation is sufficient to prove the required lower bounds. We thus get a lifting theorem that works for every gadget g with sufficiently large rank.

Cutting planes separation The crux of our separation between CP and CP* is the following observation: CP can encode a conjunction of linear equalities with a single equality, by using exponentially large coefficients. This allows CP refutations to obtain a significant saving in space when working with *linear equalities*. This saving is not available to CP*, and this difference between the proof systems allows the separation.

In order to exploit this observation, one of our main innovations is to concoct the separating formula. To do this, we must come up with a candidate formula that can only be refuted by reasoning about a large conjunction of linear equalities, to show that cutting planes (CP) can efficiently refute it, and to show that low-weight cutting planes (CP^*) cannot.

To find such a candidate formula family we resort to *pebbling formulas* which have played a major role in many proof complexity trade-off results. Interestingly, pebbling formulas have short resolution proofs that reason in terms of large conjunctions of literals. When we lift such formulas with the equality gadget this proof can be simulated in cutting planes by using the large coefficients to encode many equalities with a single equality. This yields cutting planes refutation of any pebbling formula in quadratic length and constant space.

On the other hand we prove our time-space lower bound showing that any CP^* refutation requires large length or large space for the same formulas. To prove this lower bound, the first step is to instantiate the connection in [HN12] linking time/space bounds for many proof systems to communication complexity lower bounds for lifted search problems. This connection means that we can obtain the desired CP^*-lower bounds for our formulas $\text{Peb}_G \circ \text{EQ}^n$ by proving communication complexity lower bounds for the corresponding lifted search problem $\mathsf{Search}(\text{Peb}_G) \circ \text{EQ}^n$.

In order to prove the latter communication lower bounds, we prove lower bounds on the Nullstellensatz degree of $\mathsf{Search}(\text{Peb}_G)$, and then invoke our new lifting theorem to translate them into communication lower bounds for $\mathsf{Search}(\text{Peb}_G) \circ \text{EQ}^n$. To show the Nullstellensatz lower bounds, we prove the following lemma, which establishes an equivalence between Nullstellsatz degree and the *reversible* pebbling number, and may be interesting in its own right. (We remark that connections between Nullstellensatz degree and pebbling were previously shown in [BCIP02]; however their result was not tight.)

Lemma C.1.4. *For any field $\mathbb{F}$ and any directed acyclic graph G the Nullstellensatz degree of Peb_G is equal to the reversible pebbling number of G.*

Using this equivalence we obtain near-linear Nullstellensatz degree refutations for a family of graphs with maximal pebbling number, which completes our time/space lower bound for CP^*.

We remark that we require a very specific gadget and lifting theorem in order to separate CP from CP^*. Specifically, the gadget should be strong enough, so that lifting holds for deterministic communication complexity (which can efficiently simulate small time/space CP^* proofs), but on the other hand also weak enough, so that lifting does not hold for stronger communication models (randomized, real) that can efficiently compute high-weight inequalities. The reason that we are focusing on the equality gadget is that it hits this sweet spot—it requires large deterministic communication complexity, yet has short randomized protocols, and furthermore equalities can be represented with a single pair of inequalities.

Separation for monotone formulas As was the case for the separation between CP and CP*, to obtain a separation between monotone Boolean formulas and monotone real formulas we must find a function that has just the right level of hardness.

To obtain a size lower bound for monotone Boolean formulas we invoke the characterization of formula depth by communication complexity of the Karchmer–Wigderson game [KW90]. By choosing a function that has the same Karchmer–Wigderson game as the search problem of a lifted pebbling formula, we get a depth lower bound for monotone Boolean formulas from the communication lower bound of the search problem. Note that since monotone Boolean formulas can be balanced, a depth lower bound implies a size lower bound.

In the other direction, we would like to show that these functions are easy for real computation. Analogously to the Karchmer–Wigderson relation, it was shown in [HP18] that there is a correspondence between real DAG-like communication protocols (as defined in [Kra98]) and monotone real circuits. Using this relation, a small monotone real *circuit* can be extracted from a short CP proof of the lifted pebbling formula. However, we would like to establish a monotone real *formula* upper bound. One way to achieve this is by finding small tree-like CP refutations of lifted pebbling formulas. The problem is that for many gadgets lifted pebbling formulas require exponentially long tree-like proofs. Nevertheless, for pebbling formulas lifted with the equality gadget we are able to exhibit a short *semantic* tree-like CP refutation, which via real communication yields a small monotone real formula.

C.1.5 Organization of This Paper

Section C.2 contains formal definitions of concepts discussed above and some useful facts. Our main lifting theorem is proven in Section C.3. Section C.4 is devoted to proving our separation between high-weight and low-weight cutting planes. In Section C.5 we prove the separation between monotone real and Boolean formulas. We conclude in Section C.6 with some open problems.

C.2 Preliminaries

In this section we review some background material from communication complexity and proof complexity.

C.2.1 Communication Complexity and Lifted Search Problems

Given a function $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{I}$, we denote by $g^n : \mathcal{X}^n \times \mathcal{Y}^n \to \mathcal{I}^n$ the function that takes as input n independent instances of g and applies g to each of them separately. A *total search problem* is a relation $\mathcal{S} \subseteq \mathcal{I} \times \mathcal{O}$ such that for all $z \in \mathcal{I}$ there is an $o \in \mathcal{O}$ such that $(z, o) \in \mathcal{S}$. Intuitively, S represents the computational task in which we are given an input $z \in \mathcal{I}$ and would like to find an output $o \in \mathcal{O}$ that satisfies $(z, o) \in \mathcal{S}$.

An important example of a search problem, which has proved to be very useful for proof complexity results, comes from unsatisfiable k-CNF formulas. Given a k-CNF formula $\mathcal{C}$ over variables $z_1, \ldots, z_n$, the search problem $\mathsf{Search}(\mathcal{C}) \subseteq \{0,1\}^n \times \mathcal{C}$ takes as input an assignment $z \in \{0,1\}^n$ and outputs a clause $C \in \mathcal{C}$ that is falsified by z.

Given a search problem $\mathcal{S} \subseteq \mathcal{I}^n \times \mathcal{O}$ with a product input domain and a function $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{I}$, we define the *composition* $\mathcal{S} \circ g^n \subseteq \mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{O}$ in the natural way: $(x, y, o) \in \mathcal{S} \circ g$ if and only if $(g^n(x, y), o) \in \mathcal{S}$. We remark that this composition notation extends naturally to functions: for instance, if $f : \mathcal{I}^n \to \mathbb{F}$ is a function taking values in some field $\mathbb{F}$, for example, then the composition $f \circ g^n$ is a $\mathcal{X}^n \times \mathcal{Y}^n$ matrix over $\mathbb{F}$. Second, we remark that we will sometimes write $S \circ g$ instead of $S \circ g^n$ if n is clear from context.

A *communication search problem* is a search problem with a bipartite input domain $\mathcal{I} = \mathcal{A} \times \mathcal{B}$. A communication protocol for a search problem $\mathcal{S} \subseteq \mathcal{A} \times \mathcal{B} \times \mathcal{O}$ is a strategy for a collaborative game where two players Alice and Bob hold $x \in \mathcal{A}, y \in \mathcal{B}$, respectively, and wish to output an $o \in \mathcal{O}$ such that $((x, y), o) \in \mathcal{S}$ while communicating as few bits as possible. Messages are sent sequentially until one player announces the answer and only depend on the input of one player and past messages. The cost of a protocol is the maximum number of bits sent over all inputs, and the communication complexity of a search problem, which we denote by $\mathsf{P}^{\mathrm{cc}}(\mathcal{S})$, is the minimum cost over all protocols that solve $\mathcal{S}$. For more details on communication complexity, see, for instance, [KN97].

Given a CNF formula $\mathcal{C}$ on n variables $z_1, z_2, \ldots, z_n$ and a Boolean function $g : \{0,1\}^q \times \{0,1\}^q \to \{0,1\}$, we define a *lifted formula* $\mathcal{C} \circ g^n$ as follows. For each variable z_i of $\mathcal{C}$, we have $2q$ new variables $x_{i,1}, \ldots, x_{i,q}, y_{i,1}, \ldots, y_{i,q}$. For each clause $C \in \mathcal{C}$ we replace each literal z_i or $\neg z_i$ in C by a CNF encoding of either $g(x_{i,1}, \ldots, x_{i,q}, y_{i,1}, \ldots, y_{i,q})$ or $\neg g(x_{i,1}, \ldots, x_{i,q}, y_{i,1}, \ldots, y_{i,q})$ according to the sign of the literal. We then expand the resulting expression into a CNF, which we denote by $C \circ g$, using de Morgan's rules. The substituted formula is $\mathcal{C} \circ g = \bigcup_{C \in \mathcal{C}} C \circ g$.

For the sake of an example, consider the clause $u \vee \overline{v}$, and we will substitute with the equality gadget on two bits. Formally, we replace u with $x_{u,1}x_{u,2} = y_{u,1}y_{u,2}$ and v with $x_{v,1}x_{v,2} = y_{v,1}y_{v,2}$. We can encode a two-bit equality as the CNF formula

$$(x_1x_2 = y_1y_2) \equiv (\overline{x}_1 \vee y_1) \wedge (x_1 \vee \overline{y}_1) \wedge (\overline{x}_2 \vee y_2) \wedge (x_2 \vee \overline{y}_2),$$

and a two-bit disequality as the CNF formula

$$(x_1x_2 \neq y_1y_2) \equiv (\overline{x}_1 \vee \overline{x}_2 \vee \overline{y}_1 \vee \overline{y}_2) \wedge (\overline{x}_1 \vee x_2 \vee \overline{y}_1 \vee y_2) \wedge (x_1 \vee \overline{x}_2 \vee y_1 \vee \overline{y}_2) \wedge (x_1 \vee x_2 \vee y_1 \vee y_2).$$

So, in the clause $u \vee \overline{v}$, we would substitute u for the CNF encoding of $x_{u,1}x_{u,2} = y_{u,1}y_{u,2}$ and $\overline{v}$ with the CNF encoding of $x_{v,1}x_{v,2} \neq y_{v,1}y_{v,2}$; finally, we would convert the new formula to a CNF by distributing the top $\vee$ over the $\wedge$s from the new CNF encodings.

While $\mathsf{Search}(\mathcal{C}) \circ g^n$ is not the same problem as $\mathsf{Search}(\mathcal{C} \circ g^n)$, we can reduce the former to the latter. Specifically, suppose we are given a protocol Π for $\mathsf{Search}(\mathcal{C} \circ g^n)$. Consider the following protocol Π' for $\mathsf{Search}(\mathcal{C}) \circ g^n$: Given an input (x, y), the protocol Π' interprets (x, y) as an input to Π. Now, assume that Π' outputs on (x, y) a clause D of $\mathcal{C} \circ g^n$, which was obtained from a clause C of $\mathcal{C}$. Then, the clause C is a valid $\mathsf{Search}(\mathcal{C})$ on (x, y), so Π' outputs it. Let us record this observation.

Observation C.2.1. $\mathsf{P}^{\mathrm{cc}}(\mathsf{Search}(\mathcal{C} \circ g)) \geq \mathsf{P}^{\mathrm{cc}}(\mathsf{Search}(\mathcal{C}) \circ g)$ *for any unsatisfiable CNF $\mathcal{C}$ and any Boolean gadget g.*

C.2.2 Nullstellensatz

As a proof system, Nullstellensatz allows verifying that a set of polynomials does not have a common root, and it can also be used to refute CNF formulas by converting them into polynomials. It plays an important role in our lower bounds.

Let $\mathbb{F}$ be a field, and let $\mathcal{P} = \{p_1 = 0, p_2 = 0, \dots, p_m = 0\}$ be an unsatisfiable system of polynomial equations in $\mathbb{F}[z_1, z_2, \dots, z_n]$. A *Nullstellensatz refutation* of $\mathcal{P}$ is a sequence of polynomials $q_1, q_2, \dots, q_m \in \mathbb{F}[z_1, z_2, \dots, z_n]$ such that $\sum_{i=1}^m p_i q_i = 1$ where the equality is syntactic. The *degree* of the refutation is $\max_i \deg(p_i q_i)$; the *Nullstellensatz degree* of $\mathcal{P}$, denoted $\mathsf{NS}_{\mathbb{F}}(\mathcal{P})$, is the minimum degree of any Nullstellensatz refutation of $\mathcal{P}$.

Let $\mathcal{C} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an unsatisfiable CNF formula over Boolean variables $z_1, z_2, \dots, z_n$. We introduce a standard encoding of each clause C_i as a polynomial equation. If C is a clause then let C^+ denote the set of variables occurring positively in C and C^- denote the set of variables occurring negatively in C; with this notation we can write $C = \bigvee_{z \in C^+} z \vee \bigvee_{z \in C^-} \overline{z}$. From C define the polynomial

$$\mathcal{E}(C) \equiv \prod_{z \in C^+} (1 - z) \prod_{z \in C^-} z,$$

over formal variables $z_1, z_2, \dots, z_n$. Observe that $\mathcal{E}(C) = 0$ is satisfied (over $0/1$ assignments to z_i) if and only if the corresponding assignment satisfies C. We abuse notation and let $\mathcal{E}(\mathcal{C}) = \{\mathcal{E}(C) : C \in \mathcal{C}\} \cup \{z_i^2 - z_i\}_{i \in [m]}$, and note that the second set of polynomial equations restricts the z_i inputs to $\{0, 1\}$ values. The *$\mathbb{F}$-Nullstellensatz degree* of $\mathcal{C}$, denoted $\mathsf{NS}_{\mathbb{F}}(\mathcal{C})$, is the Nullstellensatz degree of refuting $\mathcal{E}(\mathcal{C})$.

How do we know that a Nullstellensatz refutation always exists? One can deduce this from Hilbert's Nullstellensatz, but for our purposes it is enough to use a simpler version proved by Buss et al. (Theorem 5.2 in [BIK+97]): if $\mathcal{P}$ is a system of polynomial equations over $\mathbb{F}[z_1, \dots, z_n]$ with no $\{0, 1\}$ solutions, then there exists a Nullstellensatz refutation of $\mathcal{P} \cup \{z_i^2 - z_i = 0\}_{i \in [n]}$.

C.2.3 Cutting Planes

The *Cutting planes (CP)* proof system was introduced in [CCT87] as a formalization of the integer linear programming algorithm in [Gom63, Chv73]. Cutting planes proofs give a formal method to deduce new linear inequalities from old that are *sound over integer solutions*—that is, if some integral vector x^* satisfies a set of linear inequalities

$\mathcal{I}$, then x^* will also satisfy any inequality $ax \geq b$ deduced from $\mathcal{I}$ by a sequence of cutting planes deductions. The allowed deductions in a cutting planes proof are the following:

$$\text{Linear combination } \frac{\sum_i a_i x_i \geq A \qquad \sum_i b_i x_i \geq B}{\sum_i (ca_i + db_i)x_i \geq cA + dB} \qquad \text{Division} \frac{\sum_i ca_i x_i \geq A}{\sum_i a_i x_i \geq \lceil A/c \rceil}$$

where a_i, b_i, c, d, A, and B are all integers and $c, d \geq 0$.

In order to use cutting planes to refute unsatisfiable CNF formulas, we need to translate clauses to inequalities. It is easy to see how to do this by example: we translate the clause $x \vee y \vee \neg z$ to the inequality $x + y + (1 - z) \geq 1$, or, equivalently, $x + y - z \geq 0$ if we collect all constant terms on the right-hand side. For refuting CNF formulas we equip cutting planes proofs with the following additional rules ensuring all variables take $\{0, 1\}$ values:

$$\text{Variable axioms } \frac{}{x \geq 0} \quad \frac{}{-x \geq -1}$$

The goal, then, is to prove unsatisfiability by deriving the inequality $0 \geq 1$. This is possible if and only if there is no $\{0, 1\}$-assignment satifying all constraints.

As discussed in the introduction, we are interested in several natural parameters of cutting planes proof—length, space, and the sizes of the coefficients. So, we define a cutting planes refutation as a sequence of *configurations* (this is also known as the *blackboard model*). A configuration is a set of linear inequalities with integer coefficients, and a sequence of configurations $\mathbb{C}_0, \ldots, \mathbb{C}_L$ is a cutting planes refutation of a formula C if $\mathbb{C}_0 = \emptyset$, $\mathbb{C}_L$ contains the contradiction $0 \geq 1$, and each configuration $\mathbb{C}_{t+1}$ follows from $\mathbb{C}_t$ either by adding an inequality in C, by adding the result of one of the above inference rules where all the premises are in $\mathbb{C}_t$, or by removing an inequality present in $\mathbb{C}_t$. The length of a refutation is then defined to be the number of configurations L; the space[2] is $\max_{t \in [L]} |\mathbb{C}_t|$, the maximum number of inequalities in a configuration; and the coefficient bit size is the maximum size in bits of a coefficient that appears in the refutation.

For any proof system, it is natural to ask what is the minimal amount of space needed to prove tautologies. Indeed, there has been much work in the literature studying this, and for proof systems such as resolution (e.g. [ET01, ABRW02, BG03, BN08]) and polynomial calculus (e.g. [ABRW02, FLN$^+$15, BG15, BBG$^+$17]) it is known that there are unsatisfiable CNF formulas which unconditionally require large space to refute. In contrast (and quite surprisingly!) it was shown in [GPT15] that for cutting planes proofs, constant line space is always enough. The proof presented in [GPT15] does use coefficients of exponential magnitude, but the authors are not able to show that this is necessary—only that coefficients of at most constant magnitude are not sufficient.

Similarly, one can ask whether cutting planes refutations require large coefficients to realize the full power of the proof system. Towards this, define CP* to be cutting

[2]Formally, this is known as the *line space*.

planes proofs with *polynomially-bounded coefficients* or, in other words, a cutting planes refutation Π of a formula $\mathcal{C}$ with n variables is a CP* refutation if the largest coefficient in Π has magnitude $\mathsf{poly}(n, L)$.

The question of how CP* relates to unrestricted cutting planes has been raised in several papers, e.g., [BPR97, BEGJ00]. This question was studied already in [BC96], where it was proven that any cutting planes refutation in length L can be transformed into a refutation with $L^{O(1)}$ lines having coefficents of magnitude $\exp(O(L))$ (here the asymptotic notation hides a mild dependence on the size of the coefficients in the input). The authors write, however, that their original goal had been to show that coefficients of only polynomial magnitude would be enough, i.e., that CP* would be as powerful as cutting planes except possibly for a polynomial loss, but that they had to leave this as an open problem. To the best of our knowledge, there has not been a single example of *any unsatisfiable formula* where CP* could potentially perform much worse than general (high-weight) cutting planes.

Finally, as observed in [BPS07, HN12], we can use an efficient cutting planes refutation of a formula $\mathcal{C}$ to solve $\mathsf{Search}(\mathcal{C})$ by an efficient communication protocol. Since the first configuration $\mathbb{C}_0$ is always true and the last configuration $\mathbb{C}_L$ is always false, the players can simulate a binary search by evaluating the truth value of a configuration according to their joint assignment and find a true configuration followed by a false configuration. It is not hard to see that the inequality being added corresponds to a clause in $\mathcal{C}$ and it is a valid answer to $\mathsf{Search}(\mathcal{C})$.

Lemma C.2.2 ([HN12]). *If there is a cutting planes refutation of $\mathcal{C}$ in length L, line space s, and coefficient bit size c, then there is a deterministic communication protocol for $\mathsf{Search}(\mathcal{C})$ of cost $O(s(c + \log n) \log L)$.*

C.3 Rank Lifting from Any Gadget

In this section we discuss our new lifting theorem, restated next.[3]

Theorem C.3.1. *Let $\mathcal{C}$ be any unsatisfiable k-CNF on n variables and let $\mathbb{F}$ be any field. For any Boolean valued gadget g with $\mathrm{rank}(g) \geq 12enk/\mathsf{NS}_{\mathbb{F}}(\mathcal{C})$ we have*

$$\mathsf{P}^{\mathsf{cc}}(\mathsf{Search}(\mathcal{C}) \circ g) \geq \mathsf{NS}_{\mathbb{F}}(\mathcal{C}).$$

This generalizes a recent lifting theorem from [PR18], which only allowed certain "good" gadgets. The main technical step of that proof showed that "good" gadgets can be used to lift the degree of multilinear polynomials to the rank of matrices. In this section, we improve this, showing that *any* gadget with non-trivial rank can be used to lift polynomial degree to rank. Given this result, Theorem C.3.1 is proved by

[3] In fact, we prove a somewhat more general theorem (see Theorem C.3.8 in Section C.3.3 for details). We also remark that this theorem in fact holds for a stronger communication measure (Razborov's *rank measure* [Raz90]), and so implies lower bounds for other models—see Section C.3.3 for details.

reproducing the proof of [PR18] with a tighter analysis. With this in mind, in this section we will prove our new lifting argument for degree to rank, and then relegate the rest of the proof of Theorem C.3.1 to Section C.3.3.

Let us now make these arguments formal. We start by recalling the definition of a "good" gadget of [PR18].

Definition C.3.2 (Definition 3.1 in [PR18]). Let $\mathbb{F}$ be a field. A gadget $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{F}$ is *good* if for any matrices A, B of the same size we have

$$\mathrm{rank}(\mathbb{1}_{\mathcal{X},\mathcal{Y}} \otimes A + g \otimes B) = \mathrm{rank}(A) + \mathrm{rank}(g)\mathrm{rank}(B)$$

where $\mathbb{1}_{\mathcal{X},\mathcal{Y}}$ denotes the $\mathcal{X} \times \mathcal{Y}$ all-1s matrix.

In [PR18] it is shown that good gadgets are useful because they lift *degree* to *rank* when composed with multilinear polynomials.

Theorem C.3.3 (Theorem 1.2 in [PR18]). *Let $\mathbb{F}$ be any field, and let $p \in \mathbb{F}[z_1, z_2, \dots, z_n]$ be a multilinear polynomial over $\mathbb{F}$. For any good gadget $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{F}$ we have*

$$\mathrm{rank}(p \circ g^n) = \sum_{S : \hat{p}(S) \neq 0} \mathrm{rank}(g)^{|S|}.$$

In the present work, we show that a gadget being good is *not* strictly necessary to obtain the above lifting from degree to rank. In fact, composing with *any* gadget lifts degree to rank!

Theorem C.3.4. *Let $p \in \mathbb{F}[z_1, z_2, \dots, z_n]$ be any multilinear polynomial and let $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{F}$ be any non-zero gadget with* $\mathrm{rank}(g) \geq 3$. *Then*

$$\sum_{S : \hat{p}(S) \neq 0} (\mathrm{rank}(g) - 3)^{|S|} \leq \mathrm{rank}(p \circ g^n) \leq \sum_{S : \hat{p}(S) \neq 0} \mathrm{rank}(g)^{|S|}.$$

We remark that the lower bound in the theorem can be sharpened to $\mathrm{rank}(g) - 2$ if the gadget g is not full rank. While the previous theorem does not require the gadget g to be good, the notion of a good gadget will still play a key role in the proof. The general idea is that every gadget with non-trivial rank can be transformed into a good gadget with a slight modification. With this in mind, en-route to proving Theorem C.3.4 we give the following characterization of good gadgets which may be of independent interest.

Lemma C.3.5. *A gadget g is good if and only if the all-*1*s vector is not in the row or column space of g.*

In the remainder of the section we prove Theorem C.3.4 and Lemma C.3.5.

C.3.1 Proof of Lemma C.3.5

We begin by proving Lemma C.3.5, which is by a simple linear-algebraic argument. Given a matrix M over a field, let $row(M)$ denote the row-space of M and let $col(M)$ denote the column-space of M. The following characterization of when rank is additive will be crucial.

Theorem C.3.6 ([MS72]). *For any matrices A, B of the same size over any field,* $\operatorname{rank}(A+B) = \operatorname{rank}(A) + \operatorname{rank}(B)$ *if and only if* $row(A) \cap row(B) = col(A) \cap col(B) = \{\mathbf{0}\}$.

The previous theorem formalizes the intuition that rank should be additive if and only if the corresponding linear operators act on disjoint parts of the vector space. Lemma C.3.5 follows almost immediately from the previous theorem.

Proof of Lemma C.3.5. Let $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{F}$ and let $a = |\mathcal{X}|$ and $b = |\mathcal{Y}|$. Let $\mathbf{1}_n$ denote the all-1s vector of dimension n and we will write $\mathbf{1}$ when n is clear from context. By Theorem C.3.6, the gadget g is good if and only if for any matrices A, B of the same size we have

$$row(\mathbb{1} \otimes A) \cap row(g \otimes B) = col(\mathbb{1} \otimes A) \cap col(g \otimes B) = \{\mathbf{0}\}.$$

We prove that there exist matrices A, B such that $col(\mathbb{1} \otimes A) \cap col(g \otimes B) \neq \{\mathbf{0}\}$ if and only if $\mathbf{1} \in col(g)$. A similar claim holds for the row spaces, and together they imply that g is not good if and only if either $\mathbf{1} \in row(g)$ or $\mathbf{1} \in col(g)$.

We start by proving that if $\mathbf{1} \in col(g)$ then there exist matrices A, B such that $col(\mathbb{1} \otimes A) \cap col(g \otimes B) \neq \{\mathbf{0}\}$. Assume that $\mathbf{1} \in col(g)$. Let $A = B = (1)$ be the size-1 identity matrix, so $\mathbb{1} = \mathbb{1} \otimes A$ and $g = g \otimes B$. Since $\mathbf{1} \in col(\mathbb{1})$ and $\mathbf{1} \in col(g)$ it follows that $\mathbf{1} \in col(\mathbb{1} \otimes A) \cap col(\mathbb{1} \otimes g)$, and it follows that g cannot be good; the same argument applies to the row spaces.

Conversely, suppose there exists matrices A, B and a non-zero vector u such that $u \in col(\mathbb{1} \otimes A) \cap col(g \otimes B)$. It is easy to see that the column space of $\mathbb{1} \otimes A$ is

$$col(\mathbb{1} \otimes A) = \left\{ \begin{pmatrix} Ax \\ Ax \\ \vdots \\ Ax \end{pmatrix} : x \in \mathbb{F}^n \right\}$$

On the other hand, the column space of $g \otimes B$ is

$$col(g \otimes B) = \left\{ \begin{pmatrix} \sum_{j=1}^{b} g(1,j) B \vec{x}_j \\ \sum_{j=1}^{b} g(2,j) B \vec{x}_j \\ \vdots \\ \sum_{j=1}^{b} g(m,j) B \vec{x}_j \end{pmatrix} : \vec{x}_1, \vec{x}_2, \dots, \vec{x}_b \in \mathbb{F}^n \right\}$$

It follows that a non-zero vector u is in $col(\mathbb{1} \otimes A) \cap col(g \otimes B)$ if and only if there are vectors $x, \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_b$ such that $u = Ax$ and for each row index $i \in [m]$

$$Ax = \sum_{j=1}^{b} g(i,j) B \vec{x}_j.$$

By the above system of equations, we can choose $x, \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_b$ such that $Ax = u$ and

$$\sum_{j=1}^{b} g(i,j) B \vec{x}_j = u$$

for all row indices i. Let A_1 denote the top row-vector of A and B_1 denote the top row-vector of B; we therefore have

$$\sum_{j=1}^{b} g(i,j) B_1 \vec{x}_j = u_1 = A_1 x$$

for all row indices i. Let β be the vector whose jth component is $\beta_j = B_1 \vec{x}_j$; it follows from the above equations that

$$\sum_{j=1}^{b} g(i,j) B_1 \vec{x}_j = \sum_{j=1}^{b} g(i,j) \beta_j = u_1$$

for all $i \in \mathcal{X}$; or, in other words,

$$g\beta = \begin{pmatrix} u_1 \\ u_1 \\ \vdots \\ u_1 \end{pmatrix}.$$

Setting $\alpha = \beta / u_1$ we have $g\alpha = \mathbf{1}$, finishing the proof of the lemma. □

C.3.2 Proof of Theorem C.3.4

In this section we prove Theorem C.3.4 using Lemma C.3.5. The theorem follows by induction using the following lemma, and the proof mimics the proof from [PR18, Rob18].

Lemma C.3.7. *Let $\mathbb{F}$ be any field, and let $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{F}$ be any gadget with* $\operatorname{rank}(g) \geq 3$. *For any matrices A, B of the same size we have*

$$\operatorname{rank}(\mathbb{1}_{\mathcal{X},\mathcal{Y}} \otimes A + g \otimes B) \geq \operatorname{rank}(A) + (\operatorname{rank}(g) - 3)\operatorname{rank}(B)$$

where $\mathbb{1}_{\mathcal{X},\mathcal{Y}}$ is the $\mathcal{X} \times \mathcal{Y}$ all-1s matrix.

Proof. Assume without loss of generality that $|\mathcal{X}| \geq |\mathcal{Y}|$ and let $\mathbb{1} = \mathbb{1}_{\mathcal{X},\mathcal{Y}}$. Thinking of g as a matrix, let u be any column vector of g. If we zero the entries of u in g, then the remaining matrix cannot have full rank, implying that some row-vector v of the remaining matrix will become linearly dependent. Let g_1 be the $\mathcal{X} \times \mathcal{Y}$ matrix consists of the u column and v row of g, and let g_2 be the $\mathcal{X} \times \mathcal{Y}$ matrix obtained by zeroing out u and v in g. Observe $g = g_1 + g_2$, and also since that g_2 contains an all-0 row and an all-0 column it is good by Lemma C.3.5 (as any linear combination of rows/columns of g must contain a zero coordinate).

Now, observe that

$$\begin{aligned}\operatorname{rank}(\mathbb{1} \otimes A + g \otimes B) &= \operatorname{rank}(\mathbb{1} \otimes A + g_1 \otimes B + g_2 \otimes B)\\ &\geq \operatorname{rank}(\mathbb{1} \otimes A + g_2 \otimes B) - \operatorname{rank}(g_1 \otimes B)\\ &= \operatorname{rank}(\mathbb{1} \otimes A + g_2 \otimes B) - \operatorname{rank}(g_1)\operatorname{rank}(B)\end{aligned}$$

where the inequality follows since adding a rank-R matrix can decrease the rank by at most R. Since g_1 consists of a single non-zero row and column we have $\operatorname{rank}(g_1) \leq 2$; by the construction of g_2 we have $\operatorname{rank}(g_2) = \operatorname{rank}(g) - 1$. Using these facts and the fact that g_2 is good, we have

$$\begin{aligned}\operatorname{rank}(\mathbb{1} \otimes A + g_2 \otimes B) - \operatorname{rank}(g_1)\operatorname{rank}(B) &\geq \operatorname{rank}(A) + \operatorname{rank}(g_2)\operatorname{rank}(B) - 2\operatorname{rank}(B)\\ &= \operatorname{rank}(A) + (\operatorname{rank}(g) - 3)\operatorname{rank}(B). \qquad \square\end{aligned}$$

With the lemma in hand we can prove Theorem C.3.4.

Proof of Theorem C.3.4. We prove

$$\operatorname{rank}(p \circ g^n) \geq \sum_{S:\hat{p}(S)\neq 0} (\operatorname{rank}(g) - 3)^{|S|}$$

by induction on n, the number of variables.

Observe that the inequality is trivially true if $n = 0$. Assume $n > 0$, and let $\mathbb{1} = \mathbb{1}_{\mathcal{X},\mathcal{Y}}$. Write $p = q + z_1 r$ for multilinear polynomials $q, r \in \mathbb{F}[z_2, z_3, \ldots, z_n]$. Note that it clearly holds that $p \circ g^n = \mathbb{1} \otimes (q \circ g^{n-1}) + g \otimes (r \circ g^{n-1})$. From the claim we have by induction that

$$\begin{aligned}\operatorname{rank}(p \circ g^n) &= \operatorname{rank}(\mathbb{1} \otimes (q \circ g^{n-1}) + g \otimes (r \circ g^{n-1}))\\ &\geq \operatorname{rank}(q \circ g^{n-1}) + (\operatorname{rank}(g) - 3)\operatorname{rank}(r \circ g^{n-1})\\ &= \sum_{S:\hat{q}(S)\neq 0} (\operatorname{rank}(g) - 3)^{|S|} + (\operatorname{rank}(g) - 3) \sum_{T:\hat{r}(T)\neq 0} (\operatorname{rank}(g) - 3)^{|T|}\\ &= \sum_{\substack{S:\hat{p}(S)\neq 0\\ z_1 \notin S}} (\operatorname{rank}(g) - 3)^{|S|} + (\operatorname{rank}(g) - 3) \sum_{\substack{T:\hat{p}(T)\neq 0\\ z_1 \in T}} (\operatorname{rank}(g) - 3)^{|T|-1}\\ &= \sum_{S:\hat{p}(S)\neq 0} (\operatorname{rank}(g) - 3)^{|S|}.\end{aligned}$$

For the upper bound, by subadditivity of rank we have

$$\begin{aligned}\operatorname{rank}(\mathbb{1} \otimes A + g \otimes B) &\leq \operatorname{rank}(\mathbb{1} \otimes A) + \operatorname{rank}(g \otimes B) \\ &= \operatorname{rank}(\mathbb{1})\operatorname{rank}(A) + \operatorname{rank}(g)\operatorname{rank}(B) \\ &= \operatorname{rank}(A) + \operatorname{rank}(g)\operatorname{rank}(B).\end{aligned}$$

Apply the above induction argument using this inequality *mutatis mutandis*. □

C.3.3 Lifting Nullstellensatz Degree for All Gadgets

In this section we prove Theorem C.3.1. In fact, we prove the following stronger result, which implies Theorem C.3.1 as a corollary.

Theorem C.3.8. *Let $\mathcal{C}$ be an unsatisfiable k-CNF on n variables and let $\mathbb{F}$ be any field. Let g be any Boolean-valued gadget with* $\operatorname{rank}(g) \geq 4$. *Then*

$$\mathsf{P}^{\mathsf{cc}}(\mathsf{Search}(\mathcal{C}) \circ g^n) \geq \mathsf{NS}_{\mathbb{F}}(\mathcal{C}) \log\left(\frac{\mathsf{NS}_{\mathbb{F}}(\mathcal{C})\operatorname{rank}(g)}{en}\right) - \frac{6n \log e}{\operatorname{rank}(g)} - \log k.$$

The proof of the theorem follows the proof of a similar lifting theorem from [PR18]. As such, we will need some notation from that paper. Let us begin by introducing a key notion: Razborov's *rank measure*. Given sets $\mathcal{U}, \mathcal{V}$, a *rectangle cover* $\mathcal{R}$ of $\mathcal{U} \times \mathcal{V}$ is a covering of $\mathcal{U} \times \mathcal{V}$ by combinatorial rectangles.

Definition C.3.9. Let $\mathcal{U}, \mathcal{V}$ be sets and let $\mathcal{R}$ be any rectangle cover of $\mathcal{U} \times \mathcal{V}$. Let A be any $\mathcal{U} \times \mathcal{V}$ matrix over a field $\mathbb{F}$. The *rank measure* of $\mathcal{R}$ at A is the quantity

$$\mu_{\mathbb{F}}(\mathcal{R}, A) = \frac{\operatorname{rank}(A)}{\max\limits_{R \in \mathcal{R}} \operatorname{rank}(A \upharpoonright R)}.$$

Using the rank measure we can lower bound the deterministic communication complexity of composed CNF search problems as follows. The key observation is that any deterministic communication protocol outputs a rectangles that lie in a "structured" rectangle cover in the following sense. We note below that if A is a collection of tuples from some product set $\mathcal{I}^n$ then we write A_i to mean the projection of A to the ith coordinate and A_I for $I \subseteq [n]$ to mean the projection onto the coordinates in I.

Definition C.3.10. Let $\mathcal{C}$ be an unsatisfiable k-CNF on n variables and let $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be a gadget. For a clause $C \in \mathcal{C}$, a combinatorial rectangle $R \subseteq \mathcal{X}^n \times \mathcal{Y}^n$ is C-*structured* if $g^n(x, y)$ falsifies C for all $(x, y) \in R$ and for all $i \notin \mathit{Vars}(C)$ we have $R_i = \mathcal{X} \times \mathcal{Y}$. A rectangle cover $\mathcal{R}$ of $\mathcal{X}^n \times \mathcal{Y}^n$ is $\mathcal{C}$-*structured* if every $R \in \mathcal{R}$ is C-structured for some $C \in \mathcal{C}$.

Lemma C.3.11. *Let $\mathcal{C}$ be an unsatisfiable k-CNF on n variables and let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a gadget. Let $\mathbb{F}$ be any field and let A be any $\mathcal{X}^n \times \mathcal{Y}^n$ matrix over $\mathbb{F}$. Then*

$$\mathsf{P}^{\mathrm{cc}}(\mathsf{Search}(\mathcal{C}) \circ g) \geq \min_{\mathcal{R}} \log \mu_{\mathbb{F}}(\mathcal{R}, A)$$

where the minimum is taken over $\mathcal{C}$-structured rectangle covers of $\mathcal{X}^n \times \mathcal{Y}^n$.

Proof. Let Π be any communication protocol solving $\mathsf{Search}(\mathcal{C}) \circ g$, and let $\mathcal{T}$ be the monochromatic rectangle partition corresponding to Π. Since Π solves the search problem, for every rectangle $R \in \mathcal{T}$ there is a clause C such that for all $(x, y) \in R$, $g^n(x, y)$ falsifies C. We can write $R = A \times B$ for some sets $A \subseteq \mathcal{X}^{Vars(C)} \times \mathcal{X}^{[n]\setminus Vars(C)}$ and $B \subseteq \mathcal{Y}^{Vars(C)} \times \mathcal{Y}^{[n]\setminus Vars(C)}$. Consider $R' = A' \times B'$ where $A' = A_{Vars(C)} \times \mathcal{X}^{[n]\setminus Vars(C)}$ and $B' = B_{Vars(C)} \times \mathcal{X}^{[n]\setminus Vars(C)}$. Since C only depends on indices in $Vars(C)$ we have that $g^n(x, y)$ falsifies C for all $(x, y) \in R'$ and, moreover, $R'_i = \mathcal{X} \times \mathcal{Y}$ for all $i \notin Vars\big(C\big)$. It follows that R' is C-structured. Let $\mathcal{R}$ be the $\mathcal{C}$-structured rectangle covering obtained from $\mathcal{T}$ by relaxing all rectangles of $\mathcal{T}$ in this way.

We now have an $\mathcal{C}$-structured rectangle cover $\mathcal{R}$ such that every $T \in \mathcal{T}$ is contained in some rectangle of $\mathcal{R}$. Razborov [Raz90] proved that if $\mathcal{T}$ is a rectangle partition and $\mathcal{R}$ is a rectangle cover such that for each $T \in \mathcal{T}$ there is an $R \in \mathcal{R}$ such that $T \subseteq R$ it holds that

$$|\mathcal{T}| \geq \mu_{\mathbb{F}}(\mathcal{R}, A)$$

for any matrix A. Since $\log |\mathcal{T}| \leq |\Pi| = \mathsf{P}^{\mathrm{cc}}(\mathsf{Search}(\mathcal{C}) \circ g)$ the lemma follows. □

We now introduce the notion of a *certificate* of an unsatisfiable CNF formula.

Definition C.3.12. Let $\mathcal{C}$ be an unsatisfiable Boolean formula on n variables in conjunctive normal form, and let C be a clause in $\mathcal{C}$. The *certificate* of C, denoted $\mathrm{Cert}(C)$, is the partial assignment $\pi : [n] \to \{0, 1, *\}$ which falsifies C and sets the maximal number of variables to $*$s. Let $\mathrm{Cert}(\mathcal{C})$ denote the set of certificates of clauses of $\mathcal{C}$.

We say that an assignment $z \in \{0,1\}^n$ *agrees* with a certificate $\pi \in \mathrm{Cert}(\mathcal{C})$ if $\pi(i) = z_i$ for each i assigned to a $\{0,1\}$ value by π. Since the CNF formula $\mathcal{C}$ is unsatisfiable, it follows that every assignment in $z \in \{0,1\}^n$ agrees with some $\{0,1\}$-certificate of $\mathcal{C}$. Next we introduce an alternative definition of Nullstellensatz degree called the *algebraic gap complexity*.

Definition C.3.13. Let $\mathbb{F}$ be a field. Let $\mathcal{C}$ be an unsatisfiable CNF on n variables. The $\mathbb{F}$-*algebraic gap complexity* of $\mathcal{C}$ is the maximum positive integer $\mathrm{gap}_{\mathbb{F}}(\mathcal{C}) \in \mathbb{N}$ for which there exists a multilinear polynomial $p \in \mathbb{F}[z_1, z_2, \ldots, z_n]$ such that

$$\deg(p) = n \quad \text{ and } \quad \forall \pi \in \mathrm{Cert}(\mathcal{C}) : \deg(p \restriction \pi) \leq n - \mathrm{gap}_{\mathbb{F}}(\mathcal{C}) \ .$$

When the field is clear from context we will write $\mathrm{gap}(\mathcal{C})$.

In [PR18, Rob18] it was shown that the algebraic gap complexity is equal to Nullstellensatz degree.

Theorem C.3.14. *For any unsatisfiable CNF formula $\mathcal{C}$ on n variables and any field $\mathbb{F}$,* $\mathrm{gap}_{\mathbb{F}}(\mathcal{C}) = \mathsf{NS}_{\mathbb{F}}(\mathcal{C})$.

We now prove a lifting theorem from Nullstellensatz degree to the rank measure, from which Theorem C.3.8 follows by applying Lemma C.3.11.

Theorem C.3.15. *Let $\mathcal{C}$ be an unsatisfiable k-CNF on n variables and let $\mathbb{F}$ be any field. Let g be any Boolean-valued gadget with* $\mathrm{rank}(g) \geq 4$. *There is a matrix A such that for any $\mathcal{C}$-structured rectangle cover $\mathcal{R}$ we have*

$$\mu_{\mathbb{F}}(\mathcal{R}, A) \geq \frac{1}{k}\left(\frac{\mathsf{NS}_{\mathbb{F}}(\mathcal{C})\mathrm{rank}(g)}{en}\right)^{\mathsf{NS}_{\mathbb{F}}(\mathcal{C})} \exp(-6n/\mathrm{rank}(g)) \ .$$

Proof. Let $p \in \mathbb{F}[z_1, z_2, \ldots, z_n]$ be the polynomial witnessing the algebraic gap complexity $\mathrm{gap}(\mathcal{C})$, and let $A = p \circ g^n$ be the pattern matrix obtained by composing p and g. We need to analyze

$$\mu_{\mathbb{F}}(\mathcal{R}, p \circ g^n) = \frac{\mathrm{rank}_{\mathbb{F}}(p \circ g^n)}{\max_{R \in \mathcal{R}} \mathrm{rank}_{\mathbb{F}}(p \circ g^n \restriction R)} \ .$$

Let us first analyze the denominator. Let R be an arbitrary rectangle from the cover $\mathcal{R}$, and suppose that R is C-structured for the clause $C \in \mathcal{C}$. Let $\pi = \mathrm{Cert}(C)$. We want to show that

$$\mathrm{rank}_{\mathbb{F}}(p \circ g^n \restriction R) \leq \sum_{S : \widehat{p \restriction \pi}(S) \neq 0} \mathrm{rank}(g)^{|S|} \ . \tag{C.1}$$

To prove this, we claim that $p \circ g^n \restriction R$ is column-equivalent to the block matrix

$$[(p \restriction \pi) \circ g^{[n] \setminus \mathit{Vars}(C)}, (p \restriction \pi) \circ g^{[n] \setminus \mathit{Vars}(C)}, \ldots, (p \restriction \pi) \circ g^{[n] \setminus \mathit{Vars}(C)}]$$

for some number of copies of the matrix $(p \restriction \pi) \circ g^{[n] \setminus \mathit{Vars}(C)}$. Indeed Equation C.1 immediately follows from this claim as

$$\mathrm{rank}_{\mathbb{F}}(p \circ g^n \restriction R) = \mathrm{rank}_{\mathbb{F}}((p \restriction \pi) \circ g^{[n] \setminus \mathit{Vars}(C)}) \leq \sum_{S : \widehat{p \restriction \pi}(S) \neq 0} \mathrm{rank}(g)^{|S|}$$

by Theorem C.3.4. So, we now prove the claim.

Write $R = A \times B$. Fix assignments $\alpha \in A_{\mathit{Vars}(C)}$ and $\beta \in B_{\mathit{Vars}(C)}$, and note that since R is C-structured we have that $g^{\mathit{Vars}(C)}(\alpha, \beta) = \pi$ and $(\alpha, x') \in A$ and $(\beta, y') \in B$ for all x', y'. Thus, by ranging $x_{[n] \setminus \mathit{Vars}(C)}, y_{[n] \setminus \mathit{Vars}(C)}$ over all values yields the matrix $(p \restriction \pi) \circ g^{[n] \setminus \mathit{Vars}(C)}$. Then, ranging $x_{\mathit{Vars}(C)}$ and $y_{\mathit{Vars}(C)}$ over all α, β such that $g^{\mathit{Vars}(C)}(\alpha, \beta) = \pi$ yields the claim and Equation C.1.

Now, consider the rank measure $\mu_{\mathbb{F}}(\mathcal{R})$, which by Theorem C.3.4 and Equation C.1 satisfies

$$\mu_{\mathbb{F}}(\mathcal{R}) \geq \frac{\operatorname{rank}_{\mathbb{F}}(p \circ g^n)}{\max_{R \in \mathcal{R}} \operatorname{rank}_{\mathbb{F}}(p \circ g^n \restriction R)} = \frac{\sum_{S : \hat{p}(S) \neq 0} (\operatorname{rank}(g) - 3)^{|S|}}{\max_{\pi \in \operatorname{Cert}(\mathcal{C})} \sum_{S : \widehat{p \restriction \pi}(S) \neq 0} \operatorname{rank}(g)^{|S|}}$$

By definition of $\operatorname{gap}(\mathcal{C})$ we have $\deg p = n$ and thus the numerator is at least $(\operatorname{rank}(g) - 3)^n$. For the denominator, since p witnesses the algebraic gap of $\mathcal{C}$, we have that $\deg p \restriction \pi \leq n - \operatorname{gap}(\mathcal{C})$ for all $\pi \in \operatorname{Cert}(\mathcal{C})$. We may assume that $\hat{p}(S) = 0$ when $|S| < n - \operatorname{gap}(\mathcal{C})$ as the definition of algebraic gaps depends only on the coefficients of monomials of p with degree larger than $n - \operatorname{gap}(\mathcal{C})$. So, for any restriction π:

$$\begin{aligned} \sum_{S : \widehat{p \restriction \pi}(S) \neq 0} \operatorname{rank}(g)^{|S|} &\leq \sum_{i=0}^{k} \binom{n}{\operatorname{gap}(\mathcal{C}) - i} \operatorname{rank}(g)^{n - \operatorname{gap}(\mathcal{C}) - i} \\ &\leq k \left(\frac{en}{\operatorname{gap}(\mathcal{C})} \right)^{\operatorname{gap}(\mathcal{C})} \operatorname{rank}(g)^{n - \operatorname{gap}(\mathcal{C})} \end{aligned}$$

Putting it all together, and using the fact that $\operatorname{rank}(g) \geq 6en/\operatorname{gap}(\mathcal{C})$, we have

$$\begin{aligned} \mu_{\mathbb{F}}(\mathcal{R}, p \circ g^n) &\geq \frac{(\operatorname{rank}(g) - 3)^n}{k(en/\operatorname{gap}(\mathcal{C}))^{\operatorname{gap}(\mathcal{C})} \operatorname{rank}(g)^{n - \operatorname{gap}(\mathcal{C})}} \\ &= \frac{1}{k} \left(\frac{\operatorname{gap}(\mathcal{C}) \operatorname{rank}(g)}{en} \right)^{\operatorname{gap}(\mathcal{C})} \cdot \left(1 - \frac{3}{\operatorname{rank}(g)} \right)^n \\ &\geq \frac{1}{k} \left(\frac{\operatorname{gap}(\mathcal{C}) \operatorname{rank}(g)}{en} \right)^{\operatorname{gap}(\mathcal{C})} \exp(-6n/\operatorname{rank}(g)) \ . \end{aligned}$$

Since $\operatorname{gap}(\mathcal{C}) = \mathsf{NS}(\mathcal{C})$ the theorem is proved. □

Theorem C.3.8 follows immediately from Theorem C.3.15 and Lemma C.3.11.

C.4 Application: Separating Cutting Planes Systems

In this section we prove a new separation between high-weight and low-weight cutting planes proofs in the bounded-space regime.

Theorem C.4.1. *There is a family of* $O(\log \log n)$*-CNF formulas over* $O(n \log \log n)$ *variables and* $\tilde{O}(n)$ *clauses that have CP refutations in length* $\tilde{O}(n^2)$ *and line space* $O(1)$*, but for which any CP* refutation in length L and line space s must satisfy* $s \log L = \Omega(n/\log^2 n)$*.*

By the results of [GPT15], *any* unsatisfiable CNF formula has a cutting planes refutation in constant line space, albeit with exponential length and exponentially large coefficients. In Theorem C.4.1 we show that the length of such a refutation can be reduced to polynomial for certain formulas, described next.

At a high level, we prove Theorem C.4.1 using the reversible pebble game. Given any DAG G with a unique sink node t, the *reversible pebble game* [Ben89] is a single-player game that is played with a set of pebbles on G. Initially the graph is empty, and at each step the player can either place or remove a pebble on a vertex whose predecessors already have pebbles (in particular the player can always place or remove a pebble on a source). The goal of the game is to place a pebble on the sink while using as few pebbles as possible. The reversible pebbling price of a graph, denoted $\mathrm{rpeb}(G)$, is the minimum number of pebbles required to place a pebble on the sink.

The family of formulas witnessing Theorem C.4.1 are *pebbling formulas* composed with the equality gadget. Intuitively, the pebbling formula [BW01] Peb_G associated with G is a formula that claims that it is impossible to place a pebble on the sink (using any number of pebbles). Since it is always possible to place a pebble by using some amount of pebbles, this formula is clearly a contradiction.

Formally, the pebbling formula Peb_G is the following CNF formula. For each vertex $u \in V$ there is a variable z_u (intuitively, z_u should take the value "true" if and only if it is possible to place a pebble on u using any number of pebbles). The variables are constrained by the following clauses.

- a clause z_s for each source vertex s (i.e., we can always place a pebble on any source),
- a clause $\bigvee_{u \in \mathrm{pred}(v)} \neg z_u \vee z_v$ for each non-source vertex v with predecessors $\mathrm{pred}(v)$ (i.e., if we can place a pebble on the predecessors of v, then we can place a pebble on v), and
- a clause $\neg z_t$ for the sink t (i.e., it is impossible to place a pebble on t).

Proving Theorem C.4.1 factors into two tasks: a lower bound and an upper bound. By applying our lifting theorem from the previous section, the lower bound will follow immediately from a good lower bound on the Nullstellensatz degree of pebbling formulas. In order to prove lower bounds on the Nullstellensatz degree, we show in Section C.4.1 that over *every* field, the Nullstellensatz degree required to refute Peb_G is *exactly* the reversible pebbling price of G. We then use it together with our lifting theorem to prove the time-space tradeoff for bounded-coefficient cutting planes refutations of $\mathrm{Peb}_G \circ g$ in Section C.4.2 for *any* high-rank gadget g. Finally, in Section C.4.3 we prove the upper bound by presenting a short and constant-space refutation of $\mathrm{Peb}_G \circ \mathrm{EQ}$ in cutting planes with unbounded coefficients.

C.4.1 Nullstellensatz Degree of Pebbling Formulas

In this section we prove that the Nullstellensatz degree of the pebbling formula of a graph G equals the reversible pebbling number of G.

Lemma C.4.2. *For any field $\mathbb{F}$ and any graph G, $\mathsf{NS}_{\mathbb{F}}(\mathrm{Peb}_G) = \mathrm{rpeb}(G)$.*

We crucially use the following dual characterization of Nullstellensatz degree by designs [Bus98].

Definition C.4.3. Let $\mathbb{F}$ be a field, let d be a positive integer, and let $\mathcal{P}$ be an unsatisfiable system of polynomial equations over $\mathbb{F}[z_1, z_2, \dots, z_n]$. A *$d$-design* for $\mathcal{P}$ is a linear functional D on the space of polynomials satisfying the following axioms:

1. $D(1) = 1$.
2. For all $p \in \mathcal{P}$ and all polynomials q such that $\deg(pq) \le d$, we have $D(pq) = 0$.

Clearly, if we have a candidate degree-d Nullstellensatz refutation $1 = \sum p_i q_i$, then applying a d-design to both sides of the refutation yields $1 = 0$, a contradiction. Thus, if a d-design exists for a system of polynomials then there cannot be a Nullstellensatz refutation of degree d. Remarkably, a converse holds for systems of polynomials over $\{0,1\}^n$.

Theorem C.4.4 (Theorems 3, 4 in [Bus98]). *Let $\mathbb{F}$ be a field and let $\mathcal{P}$ be a system of polynomial equations over $\mathbb{F}[z_1, z_2, \dots, z_n]$ containing the Boolean equations $z_i^2 - z_i = 0$ for all $i \in [n]$. Then $\mathcal{P}$ does not have a degree-d Nullstellensatz refutation if and only if it has a d-design.*

With this characterization in hand we prove Lemma C.4.2.

Proof of Lemma C.4.2. Let G be a DAG, and consider the pebbling formula G. Following the standard translation of CNF formulas into unsatisfiable systems of polynomial equations, we express Peb_G with the following equations:

Source Equations. The equation $(1 - z_s) = 0$ for each source vertex s.

Sink Equations. The equation $z_t = 0$ for the sink vertex t.

Neighbour Equations. The equation $(1 - z_v)\prod_{u \in \mathrm{pred}(v)} z_u = 0$ for each internal vertex v.

Boolean Equations. The equation $z_v^2 - z_v = 0$ for each vertex v.

We prove that a d-design for the above system exists if and only if $d < \mathrm{rpeb}(G)$, and this implies the lemma. Let D be a d-design for the system. First, note that since the Boolean axioms are satisfied and since D is linear, it follows that D is completely specified by its value on multilinear monomials $z_T := \prod_{i \in T} z_i$ (with this notation note that $z_\emptyset := 1$). Moreover, D must satisfy the following properties:

Empty Set Axiom. $D(z_\emptyset) = 1$.

Source Axioms. $D(z_T) = D(z_T z_s)$ for every source s and every $T \subseteq [n]$ with $|T \cup \{s\}| \le d$.

Neighbour Axioms. $D(z_T z_{\mathrm{pred}(v)}) = D(z_T z_{\mathrm{pred}(v)} z_v)$ for every non-source vertex v and every $T \subseteq [n]$ with $|T \cup \mathrm{pred}(v) \cup \{v\}| \leq d$.

Sink Axiom. $D(z_T z_t) = 0$ for the sink t and every $T \subseteq [n]$ with $|T \cup \{t\}| \leq d$.

We may assume without loss of generality that $D(z_T) = 0$ for any set T with $|T| > d$.

Given a set S of vertices of G, we think of S a the reversible pebbling configuration in which there are pebbles on the vertices in S and there are no pebbles on any other vertex. We say that a configuration T is *reachable* from a configuration S if there is a sequence of legal reversible pebbling moves that changes S to T while using at most d pebbles at any given point.

Now, we claim that the only way to satisfy the first three axioms is to set $D(x_T) = 1$ for every configuration T that is reachable from $\emptyset$. To see why, observe that those axioms are satisfiable if and only if the empty configuration is assigned the value 1, any configuration containing the sink is labelled 0, and $D(z_S) = D(z_T)$ for any two configurations S, T with at most d pebbles that are mutually reachable via a single reversible pebbling move. Hence, this setting of D is the only one we need to consider.

Finally, observe that this specification of a design D satisfies the sink axiom if and only if $d < \mathrm{rpeb}(G)$, since the sink is reachable from $\emptyset$ using $\mathrm{rpeb}(G)$ pebbles but not with less (by the definition of $\mathrm{rpeb}(G)$). Therefore, a d-design for Peb_G exists if and only if $d < \mathrm{rpeb}(G)$, as required. □

C.4.2 Time-Space Lower Bounds for Low-Weight Refutations

In this section we prove the lower bound part of the time-space trade-off for CP*.

Lemma C.4.5. *There is a family of graphs $\{G_n\}$ with n vertices and constant degree, such that every CP* refutation of* $\mathrm{Peb}_{G_n} \circ \mathrm{EQ}$ *in length L and line space s must have* $s \log L = \Omega(n/\log^2 n)$.

Our plan is to lift a pebbling formula that is hard with respect to Nullstellensatz degree, and as we just proved it is enough to find a family of graphs whose reversible pebbling price is large. Paul et al. [PTC77] provide such a family (and in fact prove their hardness in the stronger standard pebbling model).

Theorem C.4.6. *There is a family of graphs $\{G_n\}$ with n vertices, constant degree, and for which* $\mathrm{rpeb}(G_n) = \Omega(n/\log n)$.

Since we collected our last ingredient, let us begin the proof.

Proof of Lemma C.4.5. Let Peb_G be the pebbling formula of a graph $G = G_n$ from the family given by Theorem C.4.6. By Lemma C.4.2 the Nullstellensatz degree of the formula is

$$\mathsf{NS}_{\mathbb{F}}(\mathrm{Peb}_G) = \mathrm{rpeb}(G) = \Omega(n/\log n) \ . \tag{C.2}$$

This allows us to use our lifting theorem, Theorem C.3.1, with an equality gadget of arity $q = \mathrm{O}(\log(n/\mathsf{NS}_{\mathbb{F}}(\mathrm{Peb}_G)) = \mathrm{O}(\log\log n)$, and obtain that the lifted search problem $\mathsf{Search}(\mathrm{Peb}_G) \circ \mathrm{EQ}$ requires deterministic communication

$$\mathsf{P}^{\mathsf{cc}}(\mathsf{Search}(\mathrm{Peb}_G) \circ \mathrm{EQ}) \geq NS_{\mathbb{F}}(\mathrm{Peb}_G) = \Omega(n/\log n) \ . \tag{C.3}$$

As we noted in Observation C.2.1, this implies that the search problem of the lifted formula also requires deterministic communication

$$\mathsf{P}^{\mathsf{cc}}(\mathsf{Search}(\mathrm{Peb}_G \circ \mathrm{EQ})) \geq \mathsf{P}^{\mathsf{cc}}(\mathsf{Search}(\mathrm{Peb}_G) \circ \mathrm{EQ}) = \Omega(n/\log n) \ . \tag{C.4}$$

Finally, using Lemma C.2.2 we have that every cutting planes refutation of the lifted formula in length L, line space s, and coefficient length c must satisfy

$$s(c + \log n)\log L = \Omega(\mathsf{P}^{\mathsf{cc}}(\mathsf{Search}(\mathrm{Peb}_G \circ \mathrm{EQ}))) = \Omega(n/\log n) \ . \tag{C.5}$$

Since the size of the lifted formula $\mathrm{Peb}_G \circ \mathrm{EQ}$ is $\tilde{\mathrm{O}}(n)$, the coefficients of a CP* refutation are bounded by a polynomial of n in magnitude, and hence by $\mathrm{O}(\log n)$ in length. Substituting the value of $c = \mathrm{O}(\log n)$ in (C.5) we obtain that

$$s \log L = \Omega(n/\log^2 n) \tag{C.6}$$

as we wanted to show. □

C.4.3 Time-Space Upper Bounds for High Weight Refutations

We now prove Theorem C.4.7, showing that cutting planes proofs with large coefficients can efficiently refute pebbling formulas composed with equality gadgets in constant line space. Let EQ_q denote the equality gadget on q bits.

Theorem C.4.7. *Let* Peb_G *be any constant-width pebbling formula. There is a cutting planes refutation of* $\mathrm{Peb}_G \circ \mathrm{EQ}_{\log\log n}$ *in length* $\tilde{O}(n^2)$ *and space* $\mathrm{O}(1)$.

We also use the following lemma, which is a "derivational" analogue of the recent result of [GPT15] showing that any set of unsatisfiable integer linear inequalities has a cutting planes refutation in constant space. As the techniques are essentially the same we postpone the proof to Section C.4.4.

Lemma C.4.8 (Space Lemma). *Let $\mathcal{C}$ be a set of integer linear inequalities over n variables that implies a clause C. Then there is a cutting planes derivation of C from $\mathcal{C}$ in length* $\mathrm{O}(n^2 2^n)$ *and space* $\mathrm{O}(1)$.

Let us begin by outlining the high level proof idea. We would like to refute the lifted formula $\mathrm{Peb}_G \circ \mathrm{EQ}_q$ using constant space. Consider first the unlifted formula Peb_G. The natural way to refute it is the following: Let $v_1, \ldots, v_n$ be a topological ordering of the vertices of G. The refutation will go over the vertices in this order, each time deriving the equation that says that the variable z_{v_i} must take the value "true" by using the equations that were derived earlier for the predecessors of v_i. Eventually, the refutation will derive the equation that says that the sink must take the value "true", which contradicts the axiom that says that the sink must be false.

Going back to the lifted formula $\mathrm{Peb}_G \circ \mathrm{EQ}_q$, we construct a refutation using the same strategy, except that now the equation of z_{v_i} is replaced with the equations

$$x_{v_i,1} = y_{v_i,1}, \ldots x_{v_i,q} = y_{v_i,q}.$$

The main obstacle is that if we implement this refutation in the naive way, we will have to store all the equations simultaneously, yielding a refutation of space $\mathrm{O}(q \cdot n)$. The key idea of our proof is that CP can encode the conjunction of many equations using a *single* equation. We can therefore use this encoding in our refutation to store at any given point all the equations that were derived so far in a single equation. The implementation yields a refutation of constant space, as required.

To see how we can encode multiple equations using a single equation, consider the following example. Suppose we wish to encode the equations

$$x_1 = y_1, x_2 = y_2, x_3 = y_3,$$

where all the variables take values in $\{0, 1\}$. Then, it is easy to see that those equations are equivalent to the equation

$$4 \cdot x_1 + 2 \cdot x_2 + x_3 = 4 \cdot y_1 + 2 \cdot y_2 + y_3.$$

This idea generalizes in the straightforward to deal with more equations, as well as with arbitrary linear gadgets, to be discussed below.

The rest of this section is devoted to the proof of Theorem C.4.7. The following notion is central to the proof. Say a gadget $g(x, y) : \{0, 1\}^q \times \{0, 1\}^q \to \{0, 1\}$ is *linear* if there exists a linear expression with integer coefficients

$$L(x, y) = c + \sum_{i=1}^{q} a_i x_i + b_i y_i$$

such that $g(x, y) = 1$ if and only if $L(x, y) = 0$. Note that the equality gadget is linear, as it corresponds to the linear expression $\sum_{i=1}^{q} 2^{i-1}(x_i - y_i)$.

Let g be any linear gadget with corresponding linear expression L. Let $K = 1 + \max_{x,y} |L(x, y)|$, and let G be the underlying DAG of the composed pebbling formula $\mathrm{Peb}_G \circ g^n$. Note that for each vertex u of G the composed formula has corresponding variables $x_u, y_u \in \{0, 1\}^q$. Once and for all, fix an ordering of the vertices of G and assume that all subsets are ordered accordingly. For a subset of vertices $U \subseteq V$ define

$$L(U) := \sum_{u_i \in U} K^i L(x_{u_i}, y_{u_i}).$$

The following claim states that $L(U)$ encodes the truth value of the conjunction

$$\bigwedge_{u_i \in U} g(x_{u_i}, y_{u_i}) \ . \tag{C.7}$$

Claim C.4.9. *For a set of vertices U and any $x, y \in \{0,1\}^{qn}$, $L(U) = 0$ if and only if $\bigwedge_{u_i \in U} g(x_{u_i}, y_{u_i}) = 1$.*

Proof. Since g is linear, if $g(x_{u_i}, y_{u_i}) = 1$ for all $u_i \in U$ then it follows that $L(x_{u_i}, y_{u_i}) = 0$ for all $u_i \in U$, which in turn implies $L(U) = 0$. Conversely, suppose $g(x_{u_i}, y_{u_i}) = 0$ for some vertex u_i, and let i be the *largest* such index. It follows that $L(u_i) \neq 0$, and clearly

$$\left|\sum_{j<i} K^j L(x_{u_j}, y_{u_j})\right| \leq \sum_{j<i} K^j |L(x_{u_j}, y_{u_j})| \leq (K-1) \sum_{j<i} K^j < K^i. \tag{C.8}$$

This implies $L(U) \neq 0$, since

$$\begin{aligned} |L(U)| &= \left| K^i \cdot L(u_i) + \sum_{j<i} K^j L(x_{u_j}, y_{u_j}) \right| \\ &\geq \left| K^i \cdot L(u_i) \right| - \left| \sum_{j<i} K^j L(x_{u_j}, y_{u_j}) \right| \\ &\geq K^i - \left| \sum_{j<i} K^j L(x_{u_j}, y_{u_j}) \right| > 0 \end{aligned}$$

□

From here on in the proof, we consider $L(U) = 0$, or $L(U)$ for short, as being syntactically represented in cutting planes as the pair of inequalities $L(U) \geq 0$, $-L(U) \geq 0$. The bulk of the proof lies in the following lemma, which shows how to "encode" and "decode" unit literals in the expressions $L(U)$.

Lemma C.4.10 (Coding Lemma). *Let U be any set of vertices. Then*

1. *For any $u \in U$ there is a cutting planes derivation of $L(u)$ from $L(U)$ in length $O(q|U|)$ and space $O(1)$.*
2. *Let $C = \neg z_{u_1} \vee \neg z_{u_2} \vee \cdots \vee \neg z_{u_{k-1}} \vee z_{u_k}$ be an axiom of Peb_G with $u_1, u_2, \ldots, u_{k-1} \in U$. Let ℓ_g and s_g be such that there exists a derivation of $L(u)$ from a CNF encoding of $g(u)$ in length ℓ_g and space s_g. From $L(u_1), L(u_2), \ldots, L(u_{k-1})$ and $C \circ g^n$ there is a cutting planes derivation of $L(u_k)$ in length $O(2^{kq}\ell_g)$ and space $O(s_g)$.*
3. *For any $u \notin U$ there is a cutting planes derivation of $L(U \cup \{u\})$ from $L(U)$ and $L(u)$ in length $O(1)$ and space $O(1)$.*

Let us first use the Coding Lemma to complete the proof. We show a more general statement from which Theorem C.4.7 follows immediately by setting $k = 3$ and $g = \mathrm{EQ}_q$, with $q = \mathrm{O}(\log\log n)$, and bounding $\ell_{\mathrm{EQ}} = \mathrm{O}(q)$ and $s_{\mathrm{EQ}} = \mathrm{O}(1)$.

Lemma C.4.11. *If* Peb_G *is a width-k pebbling formula on n variables and g is a linear gadget of arity q then there is a cutting planes refutation of* $\mathrm{Peb}_G \circ g^n$ *in length $O(n(kqn + 2^{kq}\ell_g))$ and space* $\mathrm{O}(k + s_g)$.

Proof. We begin with $L(\emptyset)$, which is represented as the pair of inequalities $0 \geq 0, 0 \geq 0$. By combining Parts (2) and (3) of the Coding Lemma we can derive $L(S)$, where S is the set of sources of G. We then follow a unit-propagation proof of Peb_G, deriving $L(u)$ for each vertex of G in topological order. Suppose at some point during the derivation we have derived $L(U)$ for some subset U of vertices. For any axiom C of Peb_G of the form $C = \neg z_{u_1} \vee \neg z_{u_2} \vee \cdots \vee \neg z_{u_{k-1}} \vee z_{u_k}$ with $u_1, u_2, \ldots, u_{k-1} \in U$ we do the following: first apply Part (1) of the Coding Lemma to obtain $L(u_i)$ for each $i \in [k-1]$. Apply Part (2) to derive $L(u_k)$, forget $L(u_i)$ for each $i \in [k-1]$, and then apply Part (3) to $L(U)$ and $L(u_k)$ to derive $L(U \cup \{u_k\})$. Continue in this way until we derive $L(z)$ where z is the sink vertex of G. Since $\{L(z), \neg z \circ g\}$ is an unsatisfiable set of linear inequalities, it follows by the Space Lemma (Lemma C.4.8) that we can deduce a contradiction in length $O(q^2 2^q)$ and space $O(1)$.

In the above proof we need to derive $L(u)$ for each of the n vertices of the graph. Deriving $L(u)$ requires at most $O(k)$ applications of Part (1), one application of Part (2), and one application of Part (3). Thus, in total, we require length $O(n(kqn + 2^{kq}\ell_g))$ and space $O(k + s_g)$. □

It remains to prove the Coding Lemma (Lemma C.4.10).

Proof of Coding Lemma. Let $U = \{u_1, u_2, \ldots, u_t\}$ be an arbitrary subset of vertices of size t. Recall the definition of $L(U) = \sum_{i=1}^{t} K^{i-1} L(u_i)$. For any $u_i \in U$ a *term* of $L(U)$ will be one of the terms $K^{i-1} L(u_i)$, which is a sum of $2q$ variables itself. We begin by defining two auxiliary operations that allow us to trim both the least and the most significant terms from $L(U)$.

To trim the i least significant terms of an inequality we essentially divide by K^i. More formally, for every variable v with a positive coefficient a_j less than K^i we add the inequality $-a_j v \geq -a_j$, and for every variable with a negative coefficient greater than $-K^i$ we add the inequality $a_j v \geq 0$. This takes length $\mathrm{O}(qi)$, since each term contributes $2q$ coefficients, and space $\mathrm{O}(1)$.

At this point all the remaining coefficients on the LHS are divisible by K^i, so we can apply the division rule. By construction the RHS is greater than $-K^i - \sum_{j \geq i} cK^j$, therefore when we divide by K^i the coefficient on the RHS becomes $-\sum_{j \geq i} cK^{j-i}$.

Finally, to restore the values of the coefficients to the values they had before dividing, we multiply by so we multiply by K^i at the end to restore them.

To trim the $m - i$ most significant terms of an inequality with m terms we need to use the opposite inequality, since the remaining part only has a semantic meaning when the most significant part vanishes. Hence we first trim the i least significant terms of the opposite inequality, keeping exactly the negation of the terms that we want to discard.

Then we add both inequalities so that only the i least significant terms remain. This takes length $O(qi)$ and space $O(1)$.

Using the trimming operations we can prove items 1–3 in the lemma.

1. We must show that for any $u \in U$ there is a cutting planes derivation of $L(u)$ from $L(U)$ in length $O(qt)$ and space $O(1)$. This is straightforward: begin by making copies of the pair of inequalities $L(U) \geq 0$ and $-L(U) \geq 0$ encoding $L(U) = 0$. Trim the terms that are strictly more and strictly less significant than $L(u)$ from both of the inequalities, in length $O(qt)$ and space $O(1)$.

2. Recall that we assumed there is a derivation Π of $L(u_k)$ from the CNF formula $z_{u_k} \circ g$ in length ℓ_g and space s_g, so our goal is to produce the set of clauses $z_{u_k} \circ g$. Any such clause D is implied by the set of inequalities $\{L(u_i)\}_{i=1}^{k-1}$ together with the CNF encoding of $C \circ g^n$, therefore it has a derivation Π_D in length $O(2^{kq})$ and space $O(1)$ by the Space Lemma (Lemma C.4.8). Replacing each usage of a clause $D \in z_{u_k} \circ g$ in Π as an axiom by the corresponding derivation Π_D we obtain a sound derivation $L(u_{t+1})$ in length $O(2^{tq}\ell_g)$ and space $O(s_g)$.

3. Simply add $K^{t+1}L(u) \geq 0$ to $L(U) \geq 0$ and $-K^{t+1}L(u) \geq 0$ to $-L(U) \geq 0$; this clearly uses bounded length and space. □

This completes the proof of Theorem C.4.7.

Note that the largest coefficient used in the refutation is bounded by K^n. Indeed, the argument can be generalized to give a continuous trade-off between the size of the largest coefficient and the number of inequalities, simply by adding a new pair of empty inequalities once the coefficient required to add a vertex to an existing pair would be too large. This means that if we allow up to ξ inequalities then we can use coefficients of size bounded by $K^{O(n/\xi)}$.

C.4.4 Proof of the Space Lemma

In this section we prove the Space Lemma (Lemma C.4.8), restated next.

Lemma C.4.12. *Let $\mathcal{C}$ be a set of inequalities over n variables that implies a clause C. Then there is a cutting planes derivation of C from $\mathcal{C}$ in length* $O(n^2 2^n)$ *and space* $O(1)$.

We do so by adapting the proof in [GPT15] that any formula has a cutting planes refutation in constant space in order to show that, in fact, we can derive any clause that follows from a set of inequalities in constant space.

At a bird's eye view, the proof in [GPT15] has two steps. The primary step is building a refutation of the complete tautology, the formula that contains all 2^n clauses with n variables each forbiding one of the possible 2^n assignments, in constant space. The authors come up with an order and a way to encode that the first K clauses are all true in small space for an arbitrary K, and the rest of the primary step consists of showing how to operate with this encoding also in small space, starting with no clause being true and adding clauses one by one until a contradiction arises. The secondary step is to transform the original set of linear inequalities into the complete tautology.

If we do not start with an unsatisfiable set of linear inequalities we obviously cannot reach a contradiction, but given a clause C that follows from $\mathcal{C}$ we can still encode that all the clauses that are a superset of C must be true, and this expression is equivalent to C.

Let us set up some notation. We number the variables from 0 to $n-1$. If α is a total assignment, we denote by C_α the clause over n variables that is falsified exactly by α. We overload notation and also denote by C_α the standard translation of the clause C_α into an inequality. We say that an assignment is less than a natural number B and write $\alpha < B$ if α is lexicographically smaller than the binary representation of B, that is if $\sum_{i=0}^{n-1} 2^i \alpha(x_i) < B$. We write T_B to denote the inequality $\sum_{i=0}^{n-1} 2^i x_i \geq B$ that is falsified exactly by the assignments $\{\alpha \in \{0,1\}^n \mid \alpha < B\}$.

We can reuse the following two intermediate lemmas from [GPT15], corresponding to the primary and the secondary steps.

Lemma C.4.13 ([GPT15]). *There is a cutting planes derivation of T_B from the set of clauses $\{C_\alpha \mid \alpha < B\}$ in length $\mathrm{O}(nB)$ and space $\mathrm{O}(1)$.*

Lemma C.4.14 ([GPT15]). *If a total assignment α falsifies a set of inequalities $\mathcal{C}$, then there is a cutting planes derivation of C_α from $\mathcal{C}$ in length $\mathrm{O}(n)$ and space $\mathrm{O}(1)$.*

Lemma C.4.13, which contains the core of the argument, follows from the proof of Lemma 3.2 in [GPT15]. We repeat Claim 3 in that proof, which shows how to inductively derive $T_{B'+1}$ from $T_{B'}$ and $C_{B'}$, not 2^n but B times.

In turn Lemma C.4.14 follows from the proof of Theorem 3.4 in [GPT15]: since α must falsify some inequality I from $\mathcal{C}$, we only need to reproduce the derivation of C_α from I verbatim.

Proof of Lemma C.4.8. Assume for now that $C = x_{n-1} \vee \cdots \vee x_{n-k}$. Consider the derivation Π of the inequality $T_{2^{n-k}}$ from $\{C_\alpha \mid \alpha < B\}$, which is equivalent to C, given by Lemma C.4.13. We build a new derivation Π' extending Π as follows.

Every time that we add an axiom C_α to a configuration in Π, we replace that step by the derivation of C_α from $\mathcal{C}$ given by Lemma C.4.14. Observe that we only add axioms C_α with $\alpha < 2^{n-k}$, and since any such assignment falsifies $\mathcal{C}$ we meet the conditions to apply Lemma C.4.14.

Finally we obtain C from $T_{2^{n-k}}$ by considering $\{T_{2^{n-k}}\}$ as a set of inequalities over the k variables $x_{n-1} \dots x_{n-k}$ and applying Lemma C.4.14 with $\alpha = 0^k$ being the only assignment over these variables that falsifies $T_{2^{n-k}}$. The result is $C_0 = C$.

To derive a general clause C that contains k' negative literals, say $C = \neg x_{n-1} \vee \cdots \vee \neg x_{n-k'} \vee x_{n-k'-1} \vee \cdots \vee x_{n-k}$, we build a derivation with the same structure as Π', except that we replace every occurrence of x_i by $(1-x_i)$ for $n-k' \leq i < n$. To do so, we replace each derivation of an axiom C_α with a derivation of the axiom $C_{\alpha+(0^{n-k'}1^{k'})}$ and, for $n-k' \le i < n$, replace each use of $x_i \ge 0$ and $-x_i \ge -1$ by $-x_i \ge -1$ and $x_i \ge 0$, respectively. Linear combination and division steps go through unchanged, we only observe that at a division step the coefficient on the right hand side differs by a multiple of the divisor, so rounding is not affected. □

C.5 Application: Separating Monotone Boolean and Real Formulas

In this section we exhibit an explicit function that exponentially separates the size of monotone Boolean formulas and monotone real formulas.

Theorem C.5.1. *There is an explicit family of functions f_n over $\mathrm{O}(n \operatorname{polylog} n)$ variables that can be computed by monotone real formulas of size $\mathrm{O}(n \operatorname{polylog} n)$ but for which every monotone Boolean formula requires size $2^{\Omega(n/\log n)}$.*

To prove the lower bound part of Theorem C.5.1 we use the characterization of formula depth by communication complexity [KW90]. Given a monotone Boolean function f, the monotone Karchmer–Wigderson game of f is a search problem $\mathrm{mKW}(f)\colon \{0,1\}^n \times \{0,1\}^n \to [n]$ defined as $((x,y),i) \in \mathrm{mKW}(f)$ if $f(x)=1$, $f(y)=0$, $x_i=1$, and $y_i=0$. In other words, given a 1-input x and a 0-input y for f, the task is to find an index $i \in [n]$ such that $x_i = 1$, and $y_i = 0$. Such an index always exists because f is monotone.

If we denote by $\mathrm{mD}(f)$ the minimum depth of a monotone Boolean formula required to compute a Boolean function f, then we can write the characterization as

Lemma C.5.2 ([KW90]). *For every function f, it holds that* $\mathrm{mD}(f) = \mathsf{P}^{\mathrm{cc}}(\mathrm{mKW}(f))$.

The analogue of this characterization for real circuits is in terms of DAG-like real protocols [Kra98, Sok17, HP18]. Since we are only interested in formulas rather than circuits we only consider tree-like protocols, which we call *locally real* protocols to distinguish them from the stronger model of real protocols, also known as real games [Kra98].

A locally real communication protocol, then, is a communication protocol where the set of inputs compatible with a node is defined by one half-space, as opposed to a real protocol where the set of compatible inputs is defined by the intersection of all half-spaces in the path leading to that node.

Formally, a locally real protocol for a search problem $\mathsf{Search}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is a tree where every internal node v is labelled with a halfspace $H_v \subseteq X \times Y$, and every leaf is additionally labelled with an element $z \in \mathcal{Z}$. The root is labelled with the full space $X \times Y$, children are consistent in the sense that if a node x has children u and v then $H_x \subseteq H_u \cup H_v$. Given an input (x,y), the protocol produces a nondeterministic output z as follows. We start at the root and at each internal node we nondeterministically move to a child that contains (x,y), which exists by the consistency condition. The output of the protocol is the label of the resulting leaf. A protocol is correct if for any input $(x,y) \in \mathcal{X} \times \mathcal{Y}$ it holds that $z \in \mathcal{Z}$.

It is not hard to turn a real formula into a locally real protocol, and the converse also holds.

Lemma C.5.3 ([HP18]). *Given a locally real protocol for the monotone Karchmer–Wigderson game of a partial function f, there exists a monotone real formula with the same underlying graph that computes f.*

In order to obtain a function whose Karchmer–Wigderson game we can analyse we use the fact that every search problem can be interpreted as the Karchmer–Wigderson game of some function. To state the result we need the notion of a nondeterministic communication protocol, which is a collection N of deterministic protocols such that $((x,y),z) \in S$ if and only if there exists some protocol $\pi \in N$ such that $\pi(x,y) = z$. The cost of a nondeterministic protocol is $\log|N| + \max_{\pi \in N} \operatorname{depth}(\pi)$.

Lemma C.5.4 ([Raz90, Gál01], see also [Rob18]). *Let S be a two-party total search problem with nondeterministic communication complexity k. There exists a partial monotone Boolean function $f : \{0,1\}^{2^k} \to \{0,1,*\}$ such that S is exactly the monotone Karchmer–Wigderson game of f.*

We use as a search problem the falsified clause search problem of a hard pebbling formula composed with equality given by Lemma C.4.5. To exhibit a real formula for the function it induces, we first build a tree-like cutting planes proof of small size of the composed pebbling formula.

Theorem C.5.5. *If C is the pebbling formula of a graph of indegree 2, then there is a tree-like semantic cutting planes refutation of $C \circ \mathrm{EQ}_{\log\log n}$ in length $\mathrm{O}(n \log n \log\log n)$.*

It is not hard to see that we can extract an efficient locally real protocol from a tree-like cutting planes refutation of small size, but let us record this fact formally.

Lemma C.5.6 (Folklore, see [Sok17]). *Given a semantic cutting planes refutation of a formula F, there is a locally real protocol for the monotone Karchmer–Widgerson game of $\mathsf{Search}(F)$ with the same underlying graph.*

Before we move into the proof of Theorem C.5.5, let us complete the proof of Theorem C.5.1.

Proof of Theorem C.5.1. Let $S = \mathsf{Search}(\mathrm{Peb}_G \circ \mathrm{EQ}_{\log\log n})$ be the search problem given by Lemma C.4.5. The non-deterministic communication complexity of f is $\log(|\mathrm{Peb}_G \circ \mathrm{EQ}_{\log\log n}|) + 1$, since given a certificate consisting of a clause falsified by the inputs each party can independently verify that their part is falsified and communicate so to the other party. Therefore by Lemma C.5.4 there is a partial monotone function f^* over $\mathrm{O}(n \operatorname{polylog} n)$ variables whose monotone Karchmer–Wigderson game is equivalent to S. By Theorem C.5.5 there is a semantic cutting planes refutation of the formula $\mathrm{Peb}_G \circ \mathrm{EQ}_{\log\log n}$ of length $\mathrm{O}(n \operatorname{polylog} n)$, which we convert into a locally real protocol for

S of size $\mathrm{O}(n \operatorname{polylog} n)$ using Lemma C.5.6, and then into a monotone real formula for f^* of size $\mathrm{O}(n \operatorname{polylog} n)$ using Lemma C.5.3. Add a threshold gate on top of the formula to ensure that the output is always Boolean and let f be the total function that the formula computes. Since f extends f^*, by Lemma C.5.2 and Lemma C.4.5 f requires monotone Boolean formulas of depth $\Omega(n/\log n)$, and therefore size $2^{\Omega(n/\log n)}$. $\square$

C.5.1 A Short Tree-like Refutation

For simplicity in this section we reinterpret the pebbling formula of a graph G of indegree 2 lifted with equality of q bits as the pebbling formula of a graph G' lifted with equality of 1 bit or XNOR, where G' is the graph where we replace every vertex in G by a blob of q vertices and we replace every edge by a bipartite complete graph between blobs, and with the difference that instead of having axioms asserting that all sinks are false, the axioms assert that some sink is false.

Without further ado, let us prove Theorem C.5.5, which follows by setting $q = \log\log n$ in the following Lemma.

Lemma C.5.7. *If $\mathcal{C}$ is the pebbling formula of a graph of indegree 2, then there is a tree-like semantic cutting planes refutation of $\mathcal{C} \circ \mathrm{EQ}_q$ in length $\mathrm{O}(nq2^q)$.*

As in Section C.4.3 we fix a topological order of G and we build a refutation by keeping two inequalities $L(W) \geq 0$ and $-L(W) \geq 0$. The main difference is that we cannot use the Coding Lemma to isolate the value of a single vertex, since then we would lose the information on the rest of vertices, therefore we have to simulate the inference steps in place as we describe next.

Let us set up some notation. If W is a set of vertices, let $g(W) = \bigwedge_{v\in W} \mathrm{XNOR}(v)$. We represent $\mathrm{XNOR}(v)$ with $L(v) = 0$, where $L(v) = x^v - y^v$, and $g(W)$ with $L(W) = 0$, where $L(W) = \sum_{v_j \in W} 2^j L(v_j)$. We begin with $W = \emptyset$ and with the trivial inequalities $0 = L(\emptyset) = 0$. Let us show how to derive each vertex.

Lemma C.5.8. *There is a tree-like semantic derivation of $L(W \cup \{w\}) \geq 0$ from $L(W) \geq 0$ and the axioms in 2^q steps.*

Proof. If w is a source, then the inequality $L(w) \geq 0$ is already an axiom, hence it is enough to multiply $L(W) \geq 0$ by 2 and add $L(w)$.

The complex case is when w has predecessors $u_1, \dots, u_q$. Let $\ell(v, b) = b + (-1)^b v$ be the literal over variable v and polarity $1 - b$. Consider the 2^q axioms $I_b \geq 0$ indexed by $b \in \{0,1\}^q$ and defined as

$$J_b = \sum_{j=1}^{q} \ell(x^{u_j}, b_j) + \ell(y^{u_j}, b_j) \tag{C.9}$$

$$I_b = L(w) + J_b \ . \tag{C.10}$$

We start with an inequality $L(W) \geq 0$. In order to have enough working space for the axioms we multiply the inequality by 2^q, and using weakening axioms we add a slack term defined as

$$S = \sum_{j=1}^{q} 2^{j-1} x^{u_j} \tag{C.11}$$

to obtain $L_0^+ \geq 0$ with

$$L_0^+ = 2^q L(W) + S \ . \tag{C.12}$$

The coefficients for S are chosen so that if we evaluate S on a string $b \in \{0,1\}^q$, the result is b interpreted as a binary number. We use it to keep track of which axioms we have processed so far, in a similar fashion to how the space-efficient refutation of the complete tautology [GPT15] that we reproduce in Section C.4.4 keeps track of processed truth value assignments.

The next step is to add each axiom to L_0^+, but for this to work we need to represent each intermediate step with one inequality as follows.

Claim C.5.9. *We can represent the Boolean expression*

$$g_B^+ = [\![L(W) \geq 0]\!] \wedge \left(g(W) \rightarrow \bigwedge_{b \leq B} I_b \right) \tag{C.13}$$

with the inequality $L_B^+ \geq 0$ defined as

$$L_B^+ = (B+1)L(w) + L_0^+ \ . \tag{C.14}$$

Proof. Let us begin proving the claim by showing that $g_B^+ \Rightarrow L_B^+ \geq 0$. First consider an assignment α that satisfies $L(W) \geq 0$ but not $g(W)$, that is an assignment where $x^{u_j} = 1$ and $y^{u_j} = 0$ for some predecessor u_j of w. Then $L_0^+{\restriction_\alpha} \geq 2^q$, hence $L_B^+{\restriction_\alpha} \geq -(B+1) + 2^q \geq 0$.

Now consider an assignment α that satisfies $g(W)$, hence $L(W) = 0$. If $\alpha_{u_1,\ldots,u_q} = b \leq B$ then, since α falsifies $J_b \geq 1$, α must satisfy $L(w) \geq 0$, so both $L_0^+ \geq 0$ and $L(w) \geq 0$. Otherwise if $\alpha_{u_1,\ldots,u_q} = b > B$ then $S{\restriction_\alpha} = b \geq B+1$, and we have $L_B^+{\restriction_\alpha} \geq -(B+1) + b \geq 0$.

Let us finish by showing that $g_B^+ \Leftarrow L_B^+ \geq 0$. First consider an assignment α that falsifies $L(W) \geq 0$. Then $L_B^+{\restriction_\alpha} \leq (B+1) - 2^q < 0$.

Now consider an assignment α that satisfies $g(W)$ but not an axiom I_b with $b \leq B$. Then in particular α falsifies $L(w) \geq 0$, hence $L_B^+{\restriction_\alpha} = (B+1)L(w) + S{\restriction_\alpha} = -(B+1) - b < 0$. This concludes the proof of the claim. □

Since $L_{B+1}^+ \geq 0$ follows semantically from $L_B^+ \geq 0$ and I_B, we can derive $g_{2^q}^+ \geq 0$ from $L_0^+ \geq 0$ and the set of axioms $I_b \geq 0$ using 2^q semantic inferences of arity 2. Also, $L_{2^q}^+ \geq 0$ is semantically (but not syntactically) equivalent to $L(W \cup \{w\}) \geq 0$, so we can be ready for the next step with a semantic inference of arity 1. □

We can derive the upper bound inequality $-L(W \cup \{w\}) \geq 0$ similarly, the main differences being that we start with $-L(W) \geq 0$ and that we use the other half of the axioms, that is $I_b = -L(w) + J_b$.

We handle the sinks in a slightly different way. Instead of using the pebbling axioms directly, we first use the pebbling axioms of all the sinks together with the axioms enforcing that some sink is false in order to derive a set of inequalities similar to pebbling axioms but with -1 in place of $L(w)$. We then use the same derivation as in Lemma C.5.8 using these inequalities in place of the axioms and we obtain $L(W) - 1 \geq 0$ and analogously $-L(W) - 1 \geq 0$. Adding both inequalities leads to the contradiction $-2 \geq 0$.

To conclude the proof it is enough to observe that we do $\mathrm{O}(2^q)$ inference steps for each vertex in G', which has order nq, hence the total length of the refutation is $\mathrm{O}(nq2^q)$.

C.6 Concluding Remarks

In this paper, we show that the cutting planes proof system (CP) is stronger than its variant with polynomially bounded coefficients (CP*) with respect to simultaneous length and space. This is the first result in proof complexity demonstrating any situation where high-weight coefficients are more powerful than low-weight coefficients. We also prove an explicit separation between monotone Boolean formulas and monotone real formulas. Previously the result was only known to hold non-constructively. To obtain these results we strengthen a lifting theorem of [PR18] to allow the lifting to work with *any* gadget with sufficiently large rank, in particular with the equality gadget—a crucial ingredient for obtaining the separations discussed above.

This work raises a number of questions. Prior to our result, no explicit function was known separating monotone real circuits or formulas from monotone Boolean circuits or formula. Although we prove an explicit formula separation, it remains open to obtain an explicit function that separates monotone real circuits from monotone Boolean circuits.

The most glaring open problem related to our cutting planes contribution is to strengthen our result to a true length separation, without any assumption on the space complexity. It is natural to ask whether techniques inspired by [Sok17, GGKS18] can be of use. Another thing to note about our trade-off result for CP* is that it is not a "true trade-off": we know that length and space cannot be optimised simultaneously, but we do not know if there in fact exist small space refutations. An interesting problem is, therefore, to exhibit formulas that present "true trade-offs" for CP* but are easy with regard to space and length in CP.

It follow from our results that standard decision tree complexity, parity decision tree complexity, and Nullstellensatz degree are equal for the falsified clause search problem of lifted pebbling formulas. In view of this we can ask ourselves what complexity measure we are actually lifting. We know that for general search problem decision tree complexity is not enough for a lifting result. How about parity decision tree complexity? Or can we leverage the fact that we have "well-behaved" rectangle covers and small certificate complexity to lift weaker complexity models? It would be valuable to have a better understanding of the relation between gadgets, outer functions/relations and complexity measures.

Acknowledgements

Different subsets of the authors would like to acknowledge fruitful and enlightening conversations with different subsets of Arkadev Chattopadhyay, Pavel Hrube, Christian Ikenmeyer, Bruno Loff, Sagnik Mukhopadhyay, Igor Carboni Oliveira, Pavel Pudlák, and Dmitry Sokolov. We are also grateful for discussions regarding literature references with Albert Atserias, Paul Beame, and Massimo Lauria.

Part of this work was carried out while several of the authors were visiting the Simons Institute for the Theory of Computing in association with the DIMACS/Simons Collaboration on Lower Bounds in Computational Complexity, which is conducted with support from the National Science Foundation.

Or Meir was supported by the Israel Science Foundation (grant No. 1445/16). Toniann Pitassi and Robert Robere were supported by NSERC. Susanna F. de Rezende and Jakob Nordström were supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611, as well as by the Knut and Alice Wallenberg grant KAW 2016.0066. Jakob Nordström also received funding from the Swedish Research Council grants 621-2012-5645 and 2016-00782. Marc Vinyals was supported by the Prof. R Narasimhan post-doctoral award.

Paper D

Nullstellensatz Size-Degree Trade-offs from Reversible Pebbling

Susanna F. de Rezende, Or Meir, Jakob Nordström, and Robert Robere

Full length version of the article published in Proceedings of the 34th Annual Computational Complexity Conference (CCC '19), July 2019, pp.18:1—18:16.

Abstract

We establish an exactly tight relation between reversible pebblings of graphs and Nullstellensatz refutations of pebbling formulas, showing that a graph G can be reversibly pebbled in time t and space s if and only if there is a Nullstellensatz refutation of the pebbling formula over G in size $t+1$ and degree s (independently of the field in which the Nullstellensatz refutation is made). We use this correspondence to prove a number of strong size-degree trade-offs for Nullstellensatz, which to the best of our knowledge are the first such results for this proof system.

D.1 Introduction

In this work, we obtain strong trade-offs in proof complexity by making a connection to pebble games played on graphs. In this introductory section we start with a brief overview of these two areas and then explain how our results follow from connecting the two.

D.1.1 Proof Complexity

Proof complexity is the study of efficiently verifiable certificates for mathematical statements. More concretely, statements of interest claim to provide correct answers to questions like:

- Given a CNF formula, does it have a satisfying assignment or not?
- Given a set of polynomials over some finite field, do they have a common root?

There is a clear asymmetry here in that it seems obvious what an easily verifiable certificate for positive answers to the above questions should be, while it is not so easy to see what a concise certificate for a negative answer could look like. The focus of proof complexity is therefore on the latter scenario.

In this paper we study the algebraic proof system system *Nullstellensatz* introduced by Beame et al. [BIK$^+$94]. A *Nullstellensatz refutation* of a set of polynomials $\mathcal{P} = \{p_i \mid i \in [m]\}$ with coefficients in a field $\mathbb{F}$ is an expression

$$\sum_{i=1}^{m} r_i \cdot p_i + \sum_{j=1}^{n} s_j \cdot (x_j^2 - x_j) = 1 \tag{D.1}$$

(where r_i, s_j are also polynomials), showing that 1 lies in the polynomial ideal in the ring $\mathbb{F}[x_1, \ldots, x_n]$ generated by $\mathcal{P} \cup \{x_j^2 - x_j \mid j \in [n]\}$. By (a slight extension of) Hilbert's Nullstellensatz, such a refutation exists if and only if there is no common $\{0,1\}$-valued root for the set of polynomials $\mathcal{P}$.

Nullstellensatz can also be viewed as a proof system for certifying the unsatisfiability of CNF formulas. If we translate a clause like, e.g., $C = x \vee y \vee \neg z$ to the polynomial $p(C) = (1-x)(1-y)z = z - yz - xz + xyz$, then an assignment to the variables in a CNF formula $\mathcal{C} = \bigwedge_{i=1}^{m} C_i$ (where we think of 1 as true and 0 as false) is satisfying precisely if all the polynomials $\{p(C_i) \mid i \in [m]\}$ vanish.

The *size* of a Nullstellensatz refutation (D.1) is the total number of monomials in all the polynomials $r_i \cdot p_i$ and $s_j \cdot (x_j^2 - x_j)$ expanded out as linear combinations of monomials. Another, more well-studied, complexity measure for Nullstellensatz is *degree*, which is defined as $\max\{\deg(r_i \cdot p_i), \deg(s_j \cdot (x_j^2 - x_j))\}$.

In order to prove a lower bound d on the Nullstellensatz degree of refuting a set of polynomials $\mathcal{P}$, one can construct a d*-design*, which is a map D from degree-d polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ to $\mathbb{F}$ such that

1. D is linear, i.e., $D(\alpha p + \beta q) = \alpha D(p) + \beta D(q)$ for $\alpha, \beta \in \mathbb{F}$;
2. $D(1) = 1$;
3. $D(rp) = 0$ for all $p \in \mathcal{P}$ and $r \in \mathbb{F}[x_1, \ldots, x_n]$ such that $\deg(rp) \leq d$;
4. $D(x^2 s) = D(xs)$ for all $s \in \mathbb{F}[x_1, \ldots, x_n]$ such that $\deg(s) \leq d - 2$.

Designs provide a characterization of Nullstellensatz degree in that there is a d-design for $\mathcal{P}$ if and only if there is no Nullstellensatz refutation of $\mathcal{P}$ in degree d [Bus98]. Another possible approach to prove degree lower bounds is by computationally efficient versions of Craig's interpolation theorem. It was shown in [PS98] that constant-degree Nullstellensatz refutations yield polynomial-size monotone span programs, and that this is also tight: every span program is a unique interpolant for some set of polynomials refutable by Nullstellensatz. This connection has not been used to obtain Nullstellensatz degree lower bounds, however, due to the difficulty of proving span program lower bounds.

Lower bounds on Nullstellensatz degree have been proven for sets of polynomials encoding combinatorial principles such as the pigeonhole principle [BCE⁺98], induction principle [BP98], house-sitting principle [CEI96, Bus98], matching [BIK⁺97], and pebbling [BCIP02]. It seems fair to say that research in algebraic proof complexity soon moved on to stronger systems such as *polynomial calculus* [CEI96, ABRW02], where the proof that 1 lies in the ideal generated by $\mathcal{P} \cup \{x_j^2 - x_j \mid j \in [n]\}$ can be constructed dynamically by a step-by-step derivation. However, the Nullstellensatz proof system has been the focus of renewed interest in a recent line of works [RPRC16, PR17, PR18, dRMN⁺19] showing that Nullstellensatz lower bounds can be lifted to stronger lower bounds for more powerful computational models using composition with gadgets. The size complexity measure for Nullstellensatz has also received attention in recent papers such as [Ber18, AO19].

In this work, we are interested in understanding the relation between size and degree in Nullstellensatz. In this context it is relevant to compare and contrast Nullstellensatz with polynomial calculus as well as with the well-known *resolution* proof system [Bla37], which operates directly on the clauses of a CNF formula and repeatedly derives resolvent clauses $C \lor D$ from clauses of the form $C \lor x$ and $D \lor \neg x$ until contradiction, in the form of the empty clause without any literals, is reached. For resolution, size is measured by counting the number of clauses, and *width*, measured as the number of literals in a largest clause in a refutation, plays an analogous role to degree for Nullstellensatz and polynomial calculus.

By way of background, it is not hard to show that for all three proof systems upper bounds on degree/width imply upper bounds on size, in the sense that if a CNF formula over n variables can be refuted in degree/width d, then such a refutation can be carried out in size $n^{\mathrm{O}(d)}$. Furthermore, this upper bound has been proven to be tight up to constant factors in the exponent for resolution and polynomial calculus [ALN16], and it follows from [LLMO09] that this also holds for Nullstellensatz. In the other direction, it has been shown for resolution and polynomial calculus that strong enough lower bounds on degree/width imply lower bounds on size [IPS99, BW01]. This is known to be false for Nullstellensatz, and the pebbling formulas discussed in more detail later in this paper provide a counter-example [BCIP02].

The size lower bounds in terms of degree/width in [IPS99, BW01] can be established by transforming refutations in small size to refutations in small degree/width. This procedure blows up the size of the refutations exponentially, however. It is natural to ask whether such a blow-up is necessary or whether it is just an artifact of the proof. More generally, given that a formula has proofs in small size and small degree/width, it is an interesting question whether both measures can be optimized simultaneously, or whether there has to be a trade-off between the two.

For resolution this question was finally answered in [Tha16], which established that there are indeed strong trade-offs between size and width making the size blow-up in [BW01] unavoidable. For polynomial calculus, the analogous question remains open.

In this paper, we show that there are strong trade-offs between size and degree for Nullstellensatz. We do so by establishing a tight relation between Nullstellensatz refutations of pebbling formulas and reversible pebblings of the graphs underlying such formulas. In order to discuss this connection in more detail, we first need to describe what reversible pebblings are. This brings us to our next topic.

D.1.2 Pebble Games

In the *pebble game* first studied by Paterson and Hewitt [PH70], one places pebbles on the vertices of a directed acyclic graph (DAG) G according to the following rules:

- If all (immediate) predecessors of an empty vertex v contain pebbles, a pebble may be placed on v.
- A pebble may be removed from any vertex at any time.

The game starts and ends with the graph being empty, and a pebble should be placed on the (unique) sink of G at some point. The complexity measures to minimize are the total number of pebbles on G at any given time (the *pebbling space*) and the number of moves (the *pebbling time*).

The pebble game and its variants have been used to study flowcharts and recursive schemata [PH70], register allocation [Set75], time and space as Turing-machine resources [Coo74, HPV77], and algorithmic time-space trade-offs [Cha73, SS77, SS79, SS83, Tom78]. In the last two decades, pebble games have seen a revival in the context of proof complexity (see, e.g., [Nor13]), and pebbling has also turned out to be useful for applications in cryptography [DNW05, AS15]. An excellent overview of pebbling up to ca. 1980 is given in [Pip80] and some more recent developments are covered in the upcoming survey [Nor19].

Bennett [Ben89] introduced the *reversible pebble game* as part of a broader program [Ben73] aimed at eliminating or reducing energy dissipation during computation. Reversible pebbling has also been of interest in the context of quantum computing. For example, it was noted in [MSR$^+$18] that reversible pebble games can be used to capture the problem of "uncomputing" intermediate values in quantum algorithms.

The reversible pebble game adds the requirement that the whole pebbling performed in reverse order should also be a correct pebbling, which means that the rules for pebble placement and removal become symmetric as follows:

- If all predecessors of an empty vertex v contain pebbles, a pebble may be placed on v.
- If all predecessors of a pebbled vertex v contain pebbles, the pebble on v may be removed.

Reversible pebblings have been studied in [LV96, Krá04, KSS18] and have been used to prove time-space trade-offs in reversible simulation of irreversible computation in [LTV98, LMT00, Wil00, BTV01]. In a different context, Potechin [Pot10] implicitly used reversible pebbling to obtain lower bounds in monotone space complexity, with the connection made explicit in later works [CP14, FPRC13]. The paper [CLNV15] (to which this overview is indebted) studied the relative power of standard and reversible pebblings with respect to space, and also established PSPACE-hardness results for estimating the minimum space required to pebble graphs (reversibly or not).

D.1.3 Our Contributions

In this paper, we obtain an exactly tight correspondence between on the one hand reversible pebblings of DAGs and on the other hand Nullstellensatz refutations of pebbling formulas over these DAGs. We show that for any DAG G it holds that G can be reversibly pebbled in time t and space s if and only if there is a Nullstellensatz refutation of the pebbling formula over G in size $t + 1$ and degree s. This correspondence holds regardless of the field in which the Nullstellensatz refutation is operating, and so, in particular, it follows that pebbling formulas have exactly the same complexity for Nullstellensatz regardless of the ambient field.

We then revisit the time-space trade-off literature for the standard pebble game, focusing on the papers [CS80, CS82, LT82]. The results in these papers do not immediately transfer to the reversible pebble game, and we are not fully able to match the tightness of the results for standard pebbling, but we nevertheless obtain strong time-space trade-off results for the reversible pebble game.

This allows us to derive Nullstellensatz size-degree trade-offs from reversible pebbling time-space trade-offs as follows. Suppose that we have a DAG G such that:

1. G can be reversibly pebbled in time $t_1 \ll t_2$.
2. G can be reversibly pebbled in space $s_1 \ll s_2$.
3. There is no reversible pebbling of G that simultaneously achieves time t_1 and space s_1.

Then for Nullstellensatz refutations of the pebbling formula Peb_G over G (which will be formally defined shortly) we can deduce that:

1. Nullstellensatz can refute Peb_G in size $t_1 + 1 \ll t_2 + 1$.
2. Nullstellensatz can also refute Peb_G in degree $s_1 \ll s_2$.
3. There is no Nullstellensatz refutation of Peb_G that simultaneously achieves size $t_1 + 1$ and degree s_1.

We prove four such trade-off results, which can be found in Section D.4. The following theorem is one example of such a result (specifically, it is a simplified version of Theorem D.4.1).

Theorem D.1.1. *There is a family of* 3*-CNF formulas* $\{F_n\}_{n=1}^{\infty}$ *of size* $\Theta(n)$ *such that:*

1. *There is a Nullstellensatz refutation of* F_n *in degree* $s_1 = \mathrm{O}\big(\sqrt[6]{n}\log n\big)$.
2. *There is a Nullstellensatz refutation of* F_n *of nearly linear size and degree* $s_2 = \mathrm{O}\big(\sqrt[3]{n}\log n\big)$.
3. *Any Nullstellensatz refutation of* F_n *in degree at most* $\sqrt[3]{n}$ *must have exponential size.*

It should be noted that this is not the first time proof complexity trade-off results have been obtained from pebble games. Pebbling formulas were used in [Ben09, BN11] to obtain size-space trade-offs for resolution, and later in [BNT13] also for polynomial calculus. However, the current reductions between pebbling and Nullstellensatz are much stronger in that they go in both directions and are exact even up to additive constants.

With regard to Nullstellensatz, it was shown in [BCIP02] that Nullstellensatz degree is lower-bounded by standard pebbling price. This was strengthened in [dRMN$^+$19], which used the connection between designs and Nullstellensatz degree discussed above to establish that the degree needed to refute a pebbling formula exactly coincides with the reversible pebbling price of the corresponding DAG (which is always at least the standard pebbling price, but can be much larger). Our reduction significantly improves on [dRMN$^+$19] by constructing a more direct reduction, inspired by [GKRS18], that can simultaneously capture both time and space.

D.1.4 Outline of This Paper

After having discussed the necessary preliminaries in Section D.2, we present the reductions between Nullstellensatz and reversible pebblings in Section D.3. In Section D.4, we prove time-space trade-offs for reversible pebblings in order to obtain size-degree trade-offs for Nullstellensatz. Section D.5 contains some concluding remarks with suggestions for future directions of research.

D.2 Preliminaries

All logarithms in this paper are base 2 unless otherwise specified. For a positive integer n we write $[n]$ to denote the set of integers $\{1, 2, \ldots, n\}$.

A *literal* a over a Boolean variable x is either the variable x itself or its negation $\neg x$ (a *positive* or *negative* literal, respectively). A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals. A k*-clause* is a clause that contains at most k literals. A formula $\mathcal{C}$ in *conjunctive normal form (CNF)* is a conjunction of clauses $\mathcal{C} = C_1 \wedge \cdots \wedge C_m$. A k*-CNF formula* is a CNF formula consisting of k-clauses. We think of clauses and CNF formulas as sets, so that the order of elements is irrelevant and there are no repetitions. A truth value assignment ρ to the variables of a CNF formula $\mathcal{C}$ is satisfying if every clause in $\mathcal{C}$ contains a literal that is true under ρ.

D.2.1 Nullstellensatz

Let $\mathbb{F}$ be any field and let $\vec{x} = \{x_1, \ldots, x_n\}$ be a set of variables. We identify a set of polynomials $\mathcal{P} = \{p_i(\vec{x}) \mid i \in [m]\}$ in the ring $\mathbb{F}[\vec{x}]$ with the statement that all $p_i(\vec{x})$ have a common $\{0,1\}$-valued root. A *Nullstellensatz refutation* of this claim is a syntactic equality

$$\sum_{i=1}^{m} r_i(\vec{x}) \cdot p_i(\vec{x}) + \sum_{j=1}^{n} s_j(\vec{x}) \cdot (x_j^2 - x_j) = 1 \ , \tag{D.2}$$

where r_i, s_j are also polynomials in $\mathbb{F}[\vec{x}]$. We sometimes refer to the polynomials $p_i(\vec{x})$ as axioms and $(x_j^2 - x_j)$ as Boolean axioms.

As discussed in the introduction, Nullstellensatz can be used as a proof system for CNF formulas by translating a clause $C = \bigvee_{x \in P} x \vee \bigvee_{y \in N} \neg y$ to the polynomial $p(C) = \prod_{x \in P}(1 - x) \cdot \prod_{y \in N} y$ and viewing Nullstellensatz refutations of $\{p(C_i) \mid i \in [m]\}$ as refutations of the CNF formula $\mathcal{C} = \bigwedge_{i=1}^{m} C_i$.

The *degree* of a Nullstellensatz refutation (D.1) is $\max\{\deg(r_i(\vec{x}) \cdot p_i(\vec{x})), \deg(s_j(\vec{x}) \cdot (x_j^2 - x_j))\}$. We define the *size* of a refutation (D.2) to be the total number of monomials encountered when all products of polynomials are expanded out as linear combinations of monomials. To be more precise, let $\mathit{mSize}(p)$ denote the number of monomials in a polynomial p written as a linear combination of monomials. Then the size of a Nullstellensatz refutation on the form (D.1) is

$$\sum_{i=1}^{m} \mathit{mSize}\big(r_i(\vec{x})\big) \cdot \mathit{mSize}\big(p_i(\vec{x})\big) + \sum_{j=1}^{n} 2 \cdot \mathit{mSize}\big(s_j(\vec{x})\big) \ . \tag{D.3}$$

This is consistent with how size is defined for the "dynamic version" of Nullstellensatz known as polynomial calculus [CEI96, ABRW02], and also with the general size definitions for so-called algebraic and semialgebraic proof systems in [ALN16, Ber18, AO19].

We remark that this is not the only possible way of measuring size, however. It can be noted that the definition (D.3) is quite wasteful in that it forces us to represent the proof in a very inefficient way. Other papers in the semialgebraic proof complexity literature, such as [GHP02, KI06, DMR09], instead define size in terms of the polynomials in isolation, more along the lines of

$$\sum_{i=1}^{m} \Big(\mathit{mSize}\big(r_i(\vec{x})\big) + \mathit{mSize}\big(p_i(\vec{x})\big)\Big) + \sum_{j=1}^{n} \Big(\mathit{mSize}\big(s_j(\vec{x})\big) + 2\Big) \ , \tag{D.4}$$

or as the bit size or "any reasonable size" of the representation of all polynomials $r_i(\vec{x}), p_i(\vec{x})$, and $s_j(\vec{x})$.

In the end, the difference is not too important since the two measures (D.3) and (D.4) are at most a square apart, and for size we typically want to distinguish between polynomial and superpolynomial. In addition, and more importantly, in this paper we will only deal with k-CNF formulas with $k = \mathrm{O}(1)$, and in this setting the two definitions are the same up to a constant factor 2^k. Therefore, we will stick with (D.3), which matches best how size is measured in the closely related proof systems resolution and polynomial calculus, and which gives the cleanest statements of our results.[1]

When proving lower bounds for algebraic proof systems it is often convenient to consider a *multilinear* setting where refutations are presented in the ring $\mathbb{F}[\vec{x}]/\{x_j^2 - x_j \mid j \in [n]\}$. Since the Boolean axioms $x_j^2 - x_j$ are no longer needed, the refutation (D.2) can be written simply as

$$\sum_{i=1}^{m} r_i(\vec{x}) \cdot p_i(\vec{x}) = 1 \ , \tag{D.5}$$

where we assume that all results of multiplications are implicitly multilinearized. It is clear that any refutation on the form (D.2) remains valid after multilinearization, and so the size and degree measures can only decrease in a multilinear setting. In this paper, we prove our lower bound in our reduction in the multilinear setting and the upper bound in the non-multilinear setting, making the tightly matching results even stronger.

D.2.2 Reversible Pebbling and Pebbling Formulas

Throughout this paper $G = (V, E)$ denotes a directed acyclic graph (DAG) of constant fan-in with vertices $V(G) = V$ and edges $\mathsf{edges}G = E$. For an edge $(u, v) \in E$ we say that u is a *predecessor* of v and v a *successor* of u. We write $pred_G(v)$ to denote the sets of all predecessors of v, and drop the subscript when the DAG is clear from context. Vertices with no predecessors/successors are called *sources/sinks*. Unless stated otherwise we will assume that all DAGs under consideration have a unique sink z.

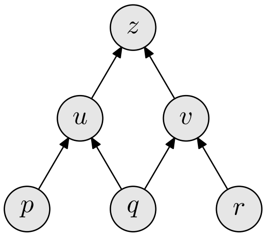
A *pebble configuration* on a DAG $G = (V, E)$ is a subset of vertices $\mathbb{P} \subseteq V$. A *reversible pebbling strategy* for a DAG G with sink z, or a *reversible pebbling* of G for short, is a sequence of pebble configurations $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_t)$ such that $\mathbb{P}_0 = \mathbb{P}_t = \emptyset$, $z \in \bigcup_{0 \le t \le t} \mathbb{P}_t$, and such that each configuration can be obtained from the previous one by one of the following rules:

1. $\mathbb{P}_{i+1} = \mathbb{P}_i \cup \{v\}$ for $v \notin \mathbb{P}_i$ such that $pred_G(v) \subseteq \mathbb{P}_i$ (a *pebble placement* on v).

2. $\mathbb{P}_{i+1} = \mathbb{P}_i \setminus \{v\}$ for $v \in \mathbb{P}_i$ such that $pred_G(v) \subseteq \mathbb{P}_i$ (a *pebble removal* from v).

The *time* of a pebbling $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_t)$ is $\mathsf{time}(\mathcal{P}) = t$ and the *space* is $\mathsf{space}(\mathcal{P}) = \max_{0 \le t \le t}\{|\mathbb{P}_t|\}$.

[1]We refer the reader to Section 2.4 in [AH18] for a more detailed discussion of the definition of proof size in algebraic and semialgebraic proof systems.



$$\begin{aligned} & x_p \\ \land\ & x_q \\ \land\ & x_r \\ \land\ & (\neg x_p \lor \neg x_q \lor x_u) \\ \land\ & (\neg x_q \lor \neg x_r \lor x_v) \\ \land\ & (\neg x_u \lor \neg x_v \lor x_z) \\ \land\ & \neg x_z \end{aligned}$$

$$\begin{aligned} & x_p - 1 \\ & x_q - 1 \\ & x_r - 1 \\ & x_p x_q (1 - x_u) \\ & x_q x_r (1 - x_v) \\ & x_u x_v (1 - x_z) \\ & x_z \end{aligned}$$

(a) Pyramid of height 2. (b) Pebbling formula in CNF. (c) Polynomial translation.

Figure D.1: Example pebbling contradiction for the pyramid graph of height 2.

We could also say that a reversible pebbling $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_t)$ should be such that $\mathbb{P}_0 = \emptyset$ and $z \in \mathbb{P}_t$, and define the time of such a pebbling to be $2t$. This is so since once we have reached a configuration containing z we can simply run the pebbling backwards (because of reversibility) until we reach the empty configuration again, and without loss of generality all time- and space-optimal reversible pebblings can be turned into such pebblings. For simplicity, we will often take this viewpoint in what follows.

For technical reasons it is sometimes important to distinguish between *visiting pebblings*, for which $z \in \mathbb{P}_t$, and *persistent pebblings*, which meet the more stringent requirement that $z \in \mathbb{P}_t = \{z\}$. (It can be noted that for the more relaxed standard pebble game discussed in the introductory section any pebbling can be assumed to be persistent without loss of generality.)

Pebble games can be encoded in CNF by so-called *pebbling formulas* [BW01], or *pebbling contradictions*. Given a DAG $G = (V, E)$ with a single sink z, we associate a variable x_v with every vertex v and add clauses encoding that

- the source vertices are all true;
- if all immediate predecessors are true, then truth propagates to the successor;
- but the sink is false.

In short, the pebbling formula over G consists of the clauses $x_v \lor \bigvee_{u \in pred(v)} \neg x_u$ for all $v \in V$ (note that if v is a source $pred(v) = \emptyset$), and the clause $\neg x_z$.

We encode this formula by a set of polynomials in the standard way. Given a set $U \subseteq V$, we denote by x_U the monomial $\prod_{u \in U} x_u$ (in particular, $x_\emptyset = 1$). For every vertex $v \in V$, we have the polynomial

$$A_v := (1 - x_v) \cdot x_{\mathrm{pred}(v)} \ , \tag{D.6}$$

and for the sink z we also have the polynomial

$$A_{\mathrm{sink}} := x_z \ . \tag{D.7}$$

See Figure D.1 for an illustration, including how the CNF formula is translated to a set of polynomials.

D.3 Reversible Pebblings and Nullstellensatz Refutations

In this section, we prove the correspondence between the reversible pebbling game on a graph G and Nullstellensatz refutation of the pebbling contradiction of G. Specifically, we prove the following result.

Theorem D.3.1. *Let G be a directed acyclic graph with a single sink, let ϕ be the corresponding pebbling contradiction, and let $\mathbb{F}$ be a field. Then, there is a reversible pebbling strategy for G with time at most t and space at most s if and only if there is a Nullstellensatz refutation for ϕ over $\mathbb{F}$ of size at most $t+1$ and degree at most s. Moreover, the same holds for multilinear Nullstellensatz refutations.*

We prove each of the directions of Theorem D.3.1 separately in Sections D.3.1 and D.3.2 below.

D.3.1 From Pebbling to Refutation

We start by proving the "only if" direction of Theorem D.3.1. Let

$$\mathbb{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_t) \tag{D.8}$$

be an optimal reversible pebbling strategy for G. Let $\mathbb{P}_{t'}$ be the first configuration in which there is a pebble on the sink z. Without loss of generality, we may assume that $t = 2 \cdot t'$: if the last $t - t'$ steps were more efficient than the first t' steps, we could have obtained a more efficient strategy by replacing the first t' steps with the (reverse of) the last $t - t'$ steps, and vice versa.

We use $\mathbb{P}$ to construct a Nullstellensatz refutation over $\mathbb{F}$ for the pebbling contradiction ϕ. To this end, we will first construct for each step $i \in [t']$ of $\mathbb{P}$ a Nullstellensatz derivation of the polynomial $x_{\mathbb{P}_{i-1}} - x_{\mathbb{P}_i}$. The sum of all these polynomials is a telescoping sum, and is therefore equal to

$$x_{\mathbb{P}_0} - x_{\mathbb{P}_{t'}} = 1 - x_{\mathbb{P}_{t'}} \quad . \tag{D.9}$$

We will then transform this sum into a Nullstellensatz refutation by adding the polynomial

$$x_{\mathbb{P}_{t'}} = A_{\text{sink}} \cdot x_{\mathbb{P}_{t'} - \{z\}} \quad . \tag{D.10}$$

We turn to constructing the aforementioned derivations. To this end, for every $i \in [t']$, let $v_i \in V$ denote the vertex which was pebbled or unpebbled during the i-th step, i.e., during the transition from $\mathbb{P}_{i-1}$ to $\mathbb{P}_i$. Then, we know that in both configurations $\mathbb{P}_{i-1}$ and $\mathbb{P}_i$ the predecessors of v_i have pebbles on them, i.e., $\text{pred}(v) \subseteq \mathbb{P}_{i-1}, \mathbb{P}_i$. Let us denote by $R_i = \mathbb{P}_i - \{v_i\} - \mathrm{pred}(v_i)$ the set of other vertices that have pebbles during the i-th step. Finally, let p_i be a number that equals to 1 if v_i was pebbled during the i-th step, and equals to -1 if v_i was unpebbled. Now, observe that

$$x_{\mathbb{P}_{i-1}} - x_{\mathbb{P}_i} = p_i \cdot x_{\mathbb{P}_{i-1}}(1 - x_{v_i}) = p_i \cdot x_{R_i} A_{v_i} \ , \tag{D.11}$$

where the last step follows since the predecessors of v_i are necessarily in $\mathbb{P}_{i-1}$. Therefore, our final refutation for ϕ is

$$\begin{aligned} \sum_{i=1}^{t'} A_{v_i} \cdot p_i \cdot x_{R_i} + A_{\mathrm{sink}} \cdot x_{\mathbb{P}_{t'} - \{z\}} &= x_{\mathbb{P}_{t'}} + \sum_{i=1}^{t'} x_{\mathbb{P}_{i-1}} - x_{\mathbb{P}_i} \\ &= x_{\mathbb{P}_{t'}} + (x_{\mathbb{P}_0} - x_{\mathbb{P}_{t'}}) \\ &= x_{\mathbb{P}_{t'}} + (1 - x_{\mathbb{P}_{t'}}) = 1 \ . \end{aligned} \tag{D.12}$$

Note, in fact, it is a multilinear Nullstellensatz refutation, since it contains only multilinear monomials and does not use the Boolean axioms. It remains to analyze its degree and size.

For the degree, observe that every monomial in the proof is of the form $x_{\mathbb{P}_i}$, and the degree of each such monomial is exactly the number of pebbles used in the corresponding configuration. It follows that the maximal degree is exactly the space of the pebbling strategy $\mathbb{P}$.

Let us turn to considering the size. Observe that for each of the configurations $\mathbb{P}_1, \ldots, \mathbb{P}_{t'}$, the refutation contains exactly two monomials: for all $i \in [t' - 1]$, one monomial for $\mathbb{P}_i$ is generated in the i-th step, and another in the $(i+1)$-th step, and for the configuration $\mathbb{P}_{t'}$ the second monomial is generated when we add $A_{\mathrm{sink}} \cdot x_{\mathbb{P}_{t'} - \{z\}}$. In addition, the refutation contains exactly one monomial for the configuration $\mathbb{P}_0$, which is generated in the first step. Hence, the total number of monomials generated in the refutation is exactly $2 \cdot t' + 1 = t + 1$, as required.

D.3.2 From Refutation to Pebbling

We turn to prove the "if" direction of Theorem D.3.1. We note that it suffices to prove it for multilinear Nullstellensatz refutations, since every standard Nullstellensatz refutation implies the existence of a multilinear one with at most the same size and degree. Let

$$\sum_{v \in V} A_v \cdot Q_v + A_{\mathrm{sink}} \cdot Q_{\mathrm{sink}} = 1 \tag{D.13}$$

be a multilinear Nullstellensatz refutation of ϕ over $\mathbb{F}$ of degree s. We use this refutation to construct a reversible pebbling strategy $\mathbb{P}$ for G.

To this end, we construct a "configuration graph" $\mathbb{C}$, whose vertices consist of all possible configurations of at most s pebbles on G (i.e., the vertices will be all subsets of V of size at most s). The edges of $\mathbb{C}$ will be determined by the polynomials Q_v of the refutation, and every edge $\{U_1, U_2\}$ in $\mathbb{C}$ will constitute a legal move in the reversible pebbling game (i.e., it will be legal to move from U_1 to U_2 and vice versa). We will show that $\mathbb{C}$ contains a path from the empty configuration $\emptyset$ to a configuration U_z that contains the sink z, and our pebbling strategy will be generated by walking on this path from $\emptyset$ to U_z and back.

The edges of the configuration graph $\mathbb{C}$ are defined as follows: Let $v \in V$ be a vertex of G, and let q be a monomial of Q_v that *does not contain* x_v. Let $W \subseteq V$ be the set of vertices such that $q = x_W$ (such a set W exists since the refutation is multilinear). Then, we put an edge e_q in $\mathbb{C}$ that connects $W \cup \mathrm{pred}(v)$ and $W \cup \mathrm{pred}(v) \cup \{v\}$ (we allow parallel edges). It is easy to see that the edge e_q connects configurations of size at most s, and that it indeed constitutes a legal move in the reversible pebbling game. We note that $\mathbb{C}$ is a bipartite graph: to see it, note that every edge e_q connects a configuration of an odd size to a configuration of an even size.

For the sake of the analysis, we assign the edge e_q a weight in $\mathbb{F}$ that is equal to coefficient of q in Q_v. We define *the weight of a configuration U* to be the sum of the weights of all the edges that touch U (where the addition is done in the field $\mathbb{F}$). We use the following technical claim, which we prove at the end of this section.

Claim D.3.2. *Let $U \subseteq V$ be a configuration in $\mathbb{C}$ that does not contain the sink z. If U is empty, then its weight is* 1. *Otherwise, its weight is* 0.

We now show how to construct the required pebbling strategy $\mathbb{P}$ for G. To this end, we first prove that there is a path in $\mathbb{C}$ from the empty configuration to a configuration that contains the sink z. Suppose for the sake of contradiction that this is not the case, and let $\mathbb{H}$ be the connected component of $\mathbb{C}$ that contains the empty configuration. Our assumption says that none of the configurations in $\mathbb{H}$ contains z.

The connected component $\mathbb{H}$ is bipartite since $\mathbb{C}$ is bipartite. Without loss of generality, assume that the empty configuration is in the left-hand side of $\mathbb{H}$. Clearly, the sum of the weights of the configurations on the left-hand side should be equal to the corresponding sum on the right-hand side, since they are both equal to the sum of the weights of the edges in $\mathbb{H}$. However, the sum of the weights of the configurations on the right-hand side is 0 (since all these weights are 0 by Claim D.3.2), while the sum of the weights of the left-hand side is 1 (again, by Claim D.3.2). We reached a contradiction, and therefore $\mathbb{H}$ must contain some configuration U_z that contains the sink z.

Next, let $\emptyset = \mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_{t'} = U_z$ be a path from the empty configuration to U_z. Our reversible pebbling strategy for G is

$$\mathbb{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_{t'-1}, \mathbb{P}_{t'}, \mathbb{P}_{t'-1}, \ldots, \mathbb{P}_0) \quad . \tag{D.14}$$

This is a legal pebbling strategy since, as noted above, every edge of $\mathbb{C}$ constitutes a legal move of the reversible pebbling game. The strategy $\mathbb{P}$ uses space s, since all the configurations in $\mathbb{C}$ contain at most s pebbles by definition. The time of $\mathbb{P}$ is $t = 2 \cdot t'$. It therefore remains to show that the size of the Nullstellensatz refutation from Equation D.13 is at least $t + 1$.

To this end, note that every edge e_q in the path corresponds to some monomial q in some polynomial Q_v. When the monomial q is multiplied by the axiom A_v, it generates two monomials in the proof: the monomial $q \cdot x_{\text{pred}(v)}$ and the monomial $q \cdot x_{\text{pred}(v)} \cdot x_v$. Hence, the Nullstellensatz refutation contains at least $2 \cdot t'$ monomials that correspond to edges from the path. In addition, the product $A_{\text{sink}} \cdot Q_{\text{sink}}$ must contains at least one monomial, since the refutation must use the sink axiom A_{sink} (because ϕ without this axiom is not a contradiction). It follows that the refutation contains at least $2 \cdot t' + 1 = t + 1$ monomials, as required. We conclude this section by proving Claim D.3.2.

Proof of Claim D.3.2. We start by introducing some terminology. First, observe that a monomial m may be generated multiple times in the refutation of Equation D.13, and we refer to each time it is generated as an *occurrence* of m. We say that an occurrence of m is *generated by a monomial* q_v *of* Q_v if it is generated by the product $A_v \cdot q_v$. Throughout the proof, we identify a configuration U with the monomial x_U.

We first prove the claim for the non-empty case. Let $U \subseteq V$ be a non-empty configuration. We would like to prove the weight of U is 0. Recall that the weight of U is the sum of the coefficients of the occurrences of U that are generated by monomials q_v *that do not contain the corresponding vertex* v. Observe that Equation D.13 implies that the sum of the coefficients of *all* the occurrences of U is 0: the coefficient of U on the right-hand side is 0, and it must be equal to the coefficient of U on the left-hand side, which is the sum of the coefficients of all the occurrences.

To complete the proof, we argue that every monomial q_v that does contain the vertex v contributes 0 to that sum. Let q_v be a monomial of Q_v that contains the vertex v and generates an occurrence of U. Let α be the coefficient of q. Then, it must hold that

$$\begin{aligned} A_v \cdot q_v &= x_{\text{pred}(v)} \cdot q_v - x_v \cdot x_{\text{pred}(v)} \cdot q_v \\ &= x_{\text{pred}(v)} \cdot q_v - x_{\text{pred}(v)} \cdot q_v \\ &= \alpha \cdot x_U - \alpha \cdot x_U \ , \end{aligned} \tag{D.15}$$

where the second equality holds since we q_v contains v and we are working with a multilinear refutation, and the third equality holds since we assumed that q_v generates an occurrence of U. It follows that q_v generates two occurrences of U, one with coefficient α and one with coefficient $-\alpha$, and therefore it contributes 0 to the sum of the coefficients of all the occurrences of U.

We have shown that the sum of the coefficients of all the occurrences of U is 0, and that the occurrences generated by monomials q_v that contain v contribute 0 to this sum. Therefore, the sum of coefficients of occurrences that are generated by monomials q_v that do not contain v must be 0, as required. In the case that U is the empty configuration, the proof is identical, except that the sum of the coefficients of all occurrences is 1, since the coefficient of $\emptyset$ is 1 on the right-hand side of Equation D.13. □

D.4 Nullstellensatz Trade-offs from Reversible Pebbling

In this section we prove Nullstellensatz refutation size-degree trade-offs for different degree regimes. Let us first recall what is known with regards to degree and size. In what follows, a Nullstellensatz refutation of a CNF formula F refers to a Nullstellensatz refutation of the translation of F to polynomials. As mentioned in the introduction, if a CNF formula over n variables can be refuted in degree d then it can be refuted in simultaneous degree d and size $n^{\mathrm{O}(d)}$. However, for Nullstellensatz it is not the case that strong enough degree lower bounds imply size lower bounds.

A natural question is whether for any given function $d_1(n)$ there is a family of CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

1. F_n has a Nullstellensatz refutation $d_1(n)$;
2. F_n has a Nullstellensatz refutation of (close to) linear size and degree $d_2(n) \gg d_1(n)$;
3. Any Nullstellensatz refutation of F_n in degree only slightly below $d_2(n)$ must have size nearly $n^{d_1(n)}$.

We present explicit constructions of formulas providing such trade-offs in several different parameter regimes. We first show that there are formulas that require exponential size in Nullstellensatz if the degree is bounded by some polynomial function, but if we allow slightly larger degree there is a nearly linear size proof.

Theorem D.4.1. *There is a family of explicitly constructible unsatisfiable 3-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

1. *There is a Nullstellensatz refutation of F_n in degree $d_1 = \mathrm{O}\big(\sqrt[6]{n}\log n\big)$.*
2. *For any constant $\epsilon > 0$, there is a Nullstellensatz refutation of F_n of size $\mathrm{O}(n^{1+\epsilon})$ and degree $d_2 = \mathrm{O}\big(d_1 \cdot \sqrt[6]{n}\big) = \mathrm{O}\big(\sqrt[3]{n}\log n\big)$.*
3. *There exists a constant $K > 0$ such that any Nullstellensatz refutation of F_n in degree at most $d = Kd_2/\log n = \mathrm{O}\big(\sqrt[3]{n}\big)$ must have size $\big(\sqrt[6]{n}\big)!$.*

We also analyse a family of formulas that can be refuted in close to logarithmic degree and show that even if we allow up to a certain polynomial degree, the Nullstellensatz size required is superpolynomial.

Theorem D.4.2. *Let $\delta > 0$ be an arbitrarily small positive constant and let $g(n)$ be any arbitrarily slowly growing monotone function $\omega(1) = g(n) \leq n^{1/4}$. Then there is a family of explicitly constructible unsatisfiable 3-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that:*

1. *There is a Nullstellensatz refutation of F_n in degree $d_1 = g(n)\log(n)$.*

2. *For any constant $\epsilon > 0$, there is a Nullstellensatz refutation of F_n of size* $O(n^{1+\epsilon})$ *and degree*
$$d_2 = O\big(d_1 \cdot n^{1/2}/g(n)^2\big) = O\big(n^{1/2}\log n/g(n)\big).$$
3. *Any Nullstellensatz refutation of F_n in degree at most*
$$d = O\big(d_2/n^{\delta}\log n\big) = O\big(n^{1/2-\delta}/g(n)\big)$$
must have size superpolynomial in n.

Still in the small-degree regime, we present a very robust trade-off in the sense that superpolynomial size lower bound holds for degree from $\log^2(n)$ to $n/\log(n)$.

Theorem D.4.3. *There is a family of explicitly constructible unsatisfiable* 3*-CNF formulas* $\{F_n\}_{n=1}^{\infty}$ *of size* $\Theta(n)$ *such that:*

1. *There is a Nullstellensatz refutation of F_n in degree* $d_1 = O(\log^2 n)$*.*
2. *For any constant $\delta > 0$, there is a Nullstellensatz refutation of F_n of size* $O(n)$ *and degree*
$$d_2 = O(d_1 \cdot n/\log^{3-\delta} n) = O(n/\log^{1-\delta} n).$$
3. *There exists a constant $K > 0$ such that any Nullstellensatz refutation of F_n in degree at most $d = Kd_2/\log^{\delta} n = O(n/\log n)$ must have size $n^{\Omega(\log\log n)}$.*

Finally, we study a family of formulas that have Nullstellensatz refutation of quadratic size and that present a smooth size-degree trade-off.

Theorem D.4.4. *There is a family of explicitly constructible unsatisfiable* 3*-CNF formulas* $\{F_n\}_{n=1}^{\infty}$ *of size* $\Theta(n)$ *such that any Nullstellensatz refutation of F_n that optimizes size given degree constraint $d = n^{\Theta(1)}$ (and less than n) has size* $\Theta\big(n^2/d\big)$*.*

We prove these results by obtaining the analogous time-space trade-offs for reversible pebbling and then applying the tight correspondence between size and degree in Nullstellensatz and time and space in reversible pebbling.

D.4.1 Reversible Pebbling Time-Space Trade-offs

Our strategy for proving reversible pebbling trade-offs will be to analyse standard pebbling trade-offs. Clearly lower bounds from standard pebbling transfer to reversible pebbling; the next theorem shows how, in a limited sense, we can also transfer *upper bounds*. It is based on a reversible simulation of irreversible computation proposed by [Ben89] and analysed precisely in [LS90].

Theorem D.4.5 ([Ben89, LS90]). *Let G be an arbitrary DAG and suppose G can be pebbled (in the standard way) using s pebbles in time $t \geq 2s$. Then for any $\epsilon > 0$, G can be reversibly pebbled in time $t^{1+\epsilon}/s^{\epsilon}$ using $\epsilon(2^{1/\epsilon}-1)s\log(t/s)$ pebbles.*

We also use the following general proposition, which allows upper bounding the reversible pebbling price of a graph by using its depth and maximum in-degree.

Proposition D.4.6. *Any DAG with maximum indegree ℓ and depth d has a persistent reversible pebbling strategy in space at most $d\ell + 1$.*

Proof. We will use the fact that if a graph has a persistent reversible strategy in space s then it has a visiting reversible strategy in space s.

The proof is by induction on the depth. For $d = 0$ we can clearly persistently reversibly pebble the graph with 1 pebble. For $d \geq 1$, we persistently reversibly pebble all but one of the (that is, at most $\ell - 1$) immediate predecessors of the sink one at a time. By the induction hypothesis, this can be done with at most $\ell - 2 + (d-1)\ell + 1 = d\ell - 1$ pebbles. At this point there are at most $\ell - 1$ predecessors of the sink which are pebbled and no other pebbles on the graph. Let v be the only non-pebbled predecessor of the sink. We do a visiting reversible pebbling of v until a pebble is placed on v. We now pebble the sink and then reverse the visiting pebbling of v until the subtree rooted at v has no pebbles on it. By the induction hypothesis, this can be done with at most $\ell + (d-1)\ell + 1 = d\ell + 1$ pebbles. All that is left to do is to to remove the $\ell - 1$ pebbles which are on predecessors of the sink. Again by the induction hypothesis, this can be done with $\ell + (d-1)\ell + 1$ pebbles. □

D.4.2 Carlson-Savage Graphs

The first family of graphs for which we present reversible pebbling trade-offs consists of the so-called Carlson-Savage graphs, which are illustrated in Figure D.2 and are defined as follows.

Definition D.4.7 (Carlson-Savage graph [CS80, CS82, Nor12]). The two-parameter graph family $\Gamma(c,r)$, for $c, r \in \mathbb{N}^+$, is defined by induction over r. The base case $\Gamma(c,1)$ is a DAG consisting of two sources s_1, s_2 and c sinks $\gamma_1, \ldots, \gamma_c$ with directed edges (s_i, γ_j), for $i = 1,2$ and $j = 1, \ldots, c$, i.e., edges from both sources to all sinks. The graph $\Gamma(c, r+1)$ has c sinks and is built from the following components:

- c disjoint copies $\Pi_r^{(1)}, \ldots, \Pi_r^{(c)}$ of a pyramid graph of height r.
- one copy of $\Gamma(c,r)$.
- c disjoint and identical line graphs called *spines*, where each spine is composed of r *sections*, and every section contains $2c$ vertices.

The above components are connected as follows: In every section of every spine, each of the first c vertices has an incoming edge from the sink of one of the first c pyramids, and each of the last c vertices has an incoming edge from one of the sinks of $\Gamma(c,r)$ (where different vertices in the same section are connected to different sinks).

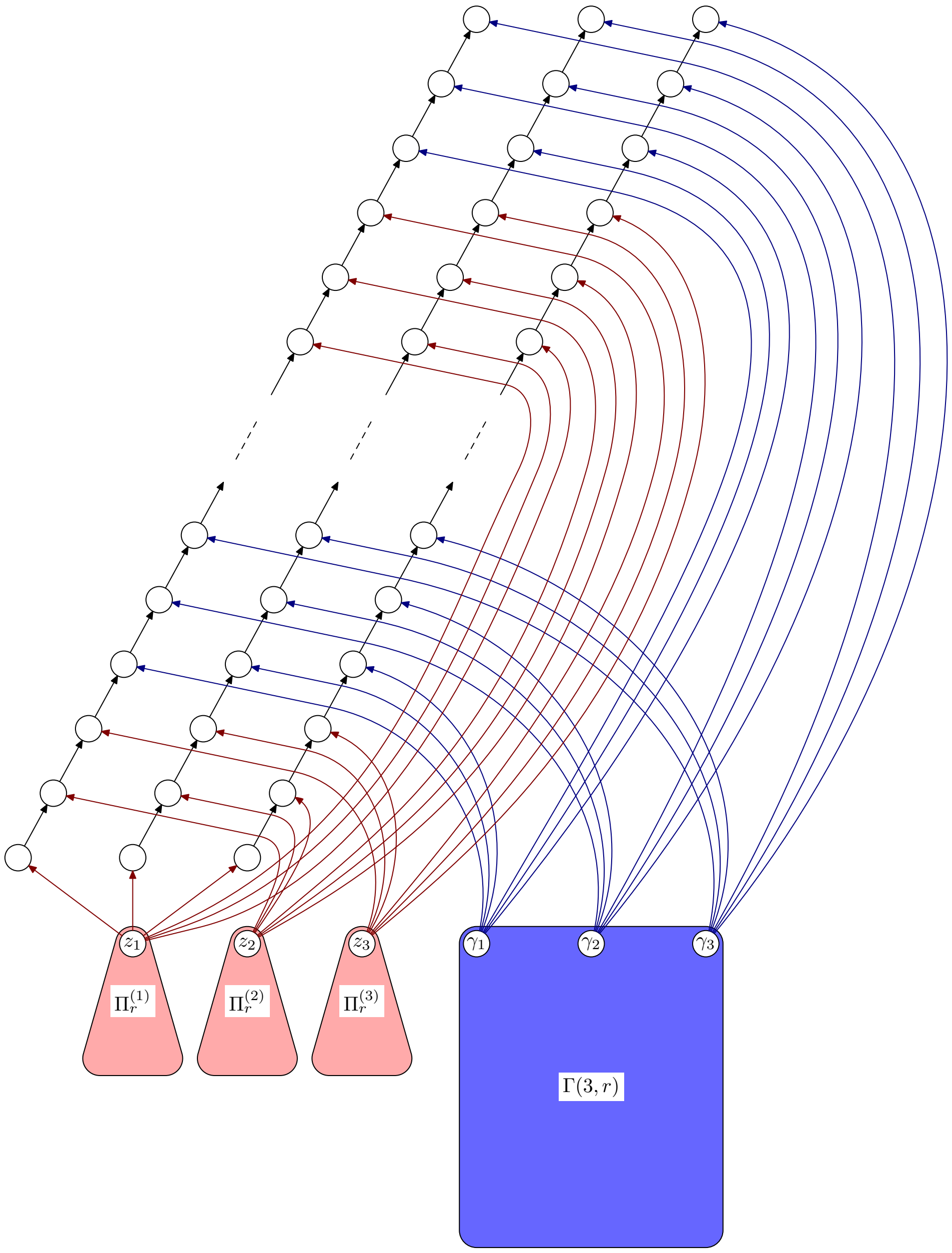


Figure D.2: Inductive definition of Carlson-Savage graph $\Gamma(3, r+1)$ with 3 spines and sinks.

Note that $\Gamma(c,r)$ has multiple sinks. We define a (reversible) pebbling of a multi-sink graph to be a (reversible) pebbling that places pebbles on each sink at some point (the pebbles do not need to be present in the last configuration). Let $\Gamma'(c,r)$ be the single-sink subgraph of $\Gamma(c,r)$ consisting of all vertices that reach the first sink of $\Gamma(c,r)$. Since all sinks are symmetric, pebbling $\Gamma'(c,r)$ and pebbling $\Gamma(c,r)$ are almost equivalent.

Proposition D.4.8. *Any (reversible) pebbling $\mathcal{P}$ of $\Gamma(c,r)$ induces a (reversible) pebbling $\mathcal{P}'$ of $\Gamma'(c,r)$ in at most the same space and the same time. From any (reversible) pebbling $\mathcal{P}'$ of $\Gamma'(c,r)$ we can obtain (reversible) pebbling $\mathcal{P}$ of $\Gamma(c,r)$ by (reversibly) pebbling each sink of $\Gamma(c,r)$ one at a time, that is, simulating $\mathcal{P}'$ c times, once for each sink. Note that $\mathsf{space}(\mathcal{P}) = \mathsf{space}(\mathcal{P}')$ and $\mathsf{time}(\mathcal{P}) = c \cdot \mathsf{time}(\mathcal{P}')$.*

Carlson and Savage proved the following properties of this graph.

Lemma D.4.9 ([CS82]). *The graphs $\Gamma(c,r)$ are of size $\Theta\big(cr^3 + c^2r^2\big)$, have in-degree 2, and have standard pebbling price $r+2$.*

Theorem D.4.10 ([CS82]). *Suppose that $\mathcal{P}$ is a standard pebbling of $\Gamma(c,r)$ in space less than $(r+2)+s$ for $0 < s \le c-3$. Then*

$$\mathsf{time}(\mathcal{P}) \ge \left(\frac{c-s}{s+1}\right)^r \cdot r! \ .$$

This lower bound holds for space up to $c+r-1$. By allowing only a constant factor more pebbles it is possible to pebble the graph (in the standard way) in linear time.

Lemma D.4.11 ([Nor12]). *The graphs $\Gamma(c,r)$ have standard pebbling strategies in simultaneous space $\mathrm{O}(c+r)$ and time linear in the size of the graphs.*

The standard pebbling price upper bound does not carry over to reversible pebbling because the line graph requires more pebbles in reversible pebbling than in standard pebbling. However, we can adapt the standard pebbling strategy to reversible pebbling using the following fact on the line graph.

Proposition D.4.12 ([LV96]). *The visiting reversible pebbling price of the line graph on n vertices is $\lceil \log(n+1) \rceil$ and the persistent reversible pebbling price is $\lfloor \log(n-1) \rfloor + 2$.*

Using this result, we get the following upper bound (which is slightly stronger then what we would get by applying Theorem D.4.5).

Lemma D.4.13. *The graphs $\Gamma(c,r)$ have reversible pebbling price at most $r(\log(cr)+3)$.*

Proof. The proof is by induction on r. Clearly, $\Gamma(c,1)$ can be reversibly pebbled with 3 pebbles. In order to pebble any sink of $\Gamma(c,r)$, we can reversibly pebble the corresponding spine with the space-efficient strategy for reversibly pebbling a line graph (as per Proposition D.4.12) and every time we need to place or remove a pebble on a given vertex of the spine, we reversibly pebble the subgraph that reaches this vertex. By Proposition D.4.6, pyramids of depth $r-1$ can be reversibly pebbled with $2(r-1)+1$ pebbles. Therefore, by induction on r we get that the reversible pebbling price of $\Gamma(c,r)$ is at most $\max\{(r-1)(\log(cr)+3), 2(r-1)+1\}+\log(2cr)+2 \leq (r-1)(\log(cr)+3)+\log(cr)+3$. □

In order to obtain nearly linear time reversible pebbling, we apply Theorem D.4.5 to Lemma D.4.11.

Lemma D.4.14. *For any $\epsilon > 0$, the graphs $\Gamma(c,r)$ have reversible pebbling strategies in simultaneous space $O(\epsilon 2^{1/\epsilon}(c+r)\log(cr))$ and time $O(n^{1+\epsilon})$ (where n denotes the number of vertices).*

We can now choose different values for the parameters c and r and obtain graphs with trade-offs in different space regimes. The first family of graphs we consider are those that exhibit exponential time lower-bounds.

Theorem D.4.15. *There are explicitly constructible families of single-sink DAGs $\{G_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ and maximum in-degree 2 such that:*

1. *The graph G_n has reversible pebbling price $s_1 = O\big(\sqrt[6]{n}\log n\big)$.*
2. *For any constant $\epsilon > 0$, there is a reversible pebbling of G_n in time $O(n^{1+\epsilon})$ and space*
$$s_2 = O\big(s_1 \cdot \sqrt[6]{n}\big) = O\big(\sqrt[3]{n}\log n\big) \ .$$
3. *There is a constant $K > 0$ such that any standard pebbling of G_n in space at most*
$$s = \frac{Ks_2}{\log n} = O\big(\sqrt[3]{n}\big)$$
must take time at least $\big(\sqrt[6]{n}\big)!$.

Proof. Let G_n be the single-sink subgraph of $\Gamma(c(n), r(n))$ consisting of all vertices that reach the first sink, for $c(n) = \sqrt[3]{n}$ and $r(n) = \sqrt[6]{n}$.

By Lemma D.4.9, G_n has $\Theta(n)$ vertices and by Proposition D.4.8, items 1–3 follow from Lemma D.4.13, Lemma D.4.14 and Theorem D.4.10, respectively. □

Given Theorem D.3.1 which proves the tight correspondence between reversible pebbling and Nullstellensatz refutations, Theorem D.4.1 follow from Theorem D.4.15.

It is also interesting to consider families of graphs that can be reversibly pebbled in very small space, close to the logarithmic lower bound on the number of pebbles required to reversibly pebble a single-sink DAG. In this small-space regime, we cannot expect exponential time lower bounds, but we can still obtain superpolynomial ones.

Theorem D.4.16. *Let $\delta > 0$ be an arbitrarily small positive constant and let $g(n)$ be any arbitrarily slowly growing monotone function $\omega(1) = g(n) \leq n^{1/4}$. Then there is a family of explicitly constructible single-sink DAGs $\{G_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ and maximum in-degree 2 such that:*

1. *The graph G_n has reversible pebbling price $s_1 \leq g(n)\log(n)$.*
2. *For any constant $\epsilon > 0$, there is a reversible pebbling of G_n in time $\mathrm{O}(n^{1+\epsilon})$ and space*
$$s_2 = \mathrm{O}\big(s_1 \cdot n^{1/2}/g(n)^2\big) = \mathrm{O}\big(n^{1/2}\log n/g(n)\big) \ .$$
3. *Any standard pebbling of G_n in space at most*
$$s = \mathrm{O}\big(s_2/n^{\delta}\log n\big) = \mathrm{O}\big(n^{1/2-\delta}/g(n)\big)$$
requires time superpolynomial in n.

Proof. The proof is analogous to that of Theorem D.4.16 with parameters $r(n) = g(n)$ and $c(n) = n^{1/2}/g(n)$. □

By applying Theorem D.3.1 to the above result we obtain Theorem D.4.2.

Remark D.4.17. We note that in the second items of both the foregoing theorems, we could have reduced the time of the reversible pebbling to $\mathrm{O}(n^{1+\mathrm{o}(1)})$ by applying Theorem D.4.5 with $\epsilon = \mathrm{O}\left(\frac{1}{\log\log n}\right)$. This would have come at a cost of an extra logarithmic factor in the corresponding space bounds.

D.4.3 Stacks of Superconcentrators

Lengauer and Tarjan [LT82] studied robust superpolynomial trade-offs for standard pebbling and showed that there are graphs that have standard pebbling price $\mathrm{O}(\log^2 n)$, but that any standard pebbling in space up to $Kn/\log n$, for some constant K, requires superpolynomial time. For reversible pebbling we get almost the same result for the same family of graphs.

Theorem D.4.18. *There are explicitly constructible families of single-sink DAGs $\{G_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ and maximum in-degree 2 such that:*

1. *The graph G_n has reversible pebbling price $s_1 = \mathrm{O}(\log^2 n)$.*
2. *For any constant $\delta > 0$, there is a reversible pebbling of G_n in time $\mathrm{O}(n)$ and space*
$$s_2 = \mathrm{O}(s_1 \cdot n/\log^{3-\delta} n) = \mathrm{O}(n/\log^{1-\delta} n) \ .$$
3. *There exists a constant $K > 0$ such that any standard pebbling $\mathcal{P}_n$ of G_n using at most pebbles $s = \frac{Ks_2}{\log^{\delta} n} = O(n/\log n)$ requires time $n^{\Omega(\log\log n)}$.*

Note that, together with Theorem D.3.1, this implies Theorem D.4.3. Now in order to prove this theorem we must first introduce the family of graphs we will consider.

Definition D.4.19 (Superconcentrator [Val75a]). A directed acyclic graph G is an m-superconcentrator if it has m sources $S = \{s_1, \ldots, s_m\}$, m sinks $Z = \{z_1, \ldots, z_m\}$, and for any subsets S' and Z' of sources and sinks of size $\left|S'\right| = \left|Z'\right| = \ell$ it holds that there are ℓ vertex-disjoint paths between S' and Z' in G.

Pippenger [Pip77] proved that there are superconcentrators of linear size and logarithmic depth, and Gabber and Galil [GG81] gave the first explicit construction. For concreteness, we will consider the explicit construction by Alon and Capalbo [AC03] which has better parameters.

Theorem D.4.20 ([AC03]). *For all integers $k \geq 6$, there are explicitly constructible m-superconcentrators for $m = \mathrm{O}(2^k)$ with $\mathrm{O}(m)$ edges, depth $\mathrm{O}(\log m)$, and maximum indegree $\mathrm{O}(1)$.*

It is easy to see that we can modify these superconcentrators so that the maximum indegree is 2 by substituting each vertex with indegree $\delta > 2$ by a binary tree with δ leafs. Note that this only increase the size and the depth by constant factors.

Corollary D.4.21. *There are m-superconcentrators with $\mathrm{O}(m)$ vertices, maximum indegree 2 and depth $\mathrm{O}(\log m)$.*

Given an m-superconcentrator G_m, we define a stack of r superconcentrators G_m to be r disjoint copies of G_m where each sink of the ith copy is connected to a different source of the $i+1$st copy for $i \in [r-1]$. Since we want single-sink DAGs, we add a binary tree with m leafs and depth $\lceil \log m \rceil$, and connect each sink of the rth copy of G_m to a different leaf of the tree. Lengauer and Tarjan [LT82] proved the following theorem for stacks of superconcentrators.

Theorem D.4.22 ([LT82]). *Let $\Phi(m,r)$ denote a stack of r (explicitly constructible) linear-size m-superconcentrator with bounded indegree and depth $\log m$. Then the following holds:*

1. *The standard pebbling price of $\Phi(m,r)$ is $\mathrm{O}(r \log m)$.*

2. *There is a linear-time standard pebbling strategy $\mathcal{P}$ for $\Phi(m,r)$ with $\mathsf{space}(\mathcal{P}) = \mathrm{O}(m)$.*

3. *If $\mathcal{P}$ is a standard pebbling strategy for $\Phi(m,r)$ in space $s \leq m/20$, then $\mathsf{time}(\mathcal{P}) \geq m \cdot \left(\frac{rm}{64s}\right)^r$.*

With this result in hand we can now proceed to prove Theorem D.4.18.

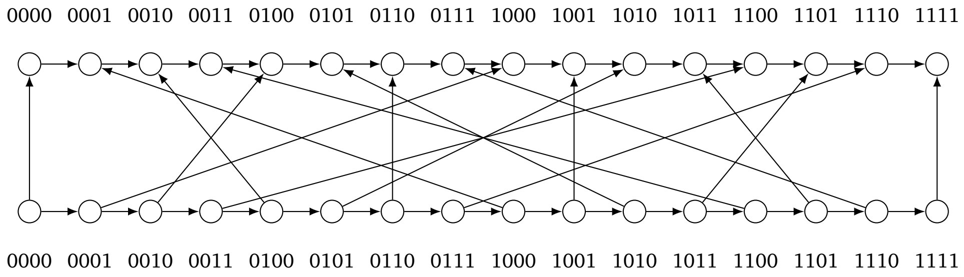


Figure D.3: A bit-reversal permutation graph.

Proof of Theorem D.4.18. Let G_n be a stack of $\log n$ linear-size $(n/\log n)$-superconcentrators as per Corollary D.4.21. Note that G_n has $\Theta(n)$ vertices, indegree 2 and depth $O(\log^2 n)$. By Proposition D.4.6 we have that G_n can be reversibly pebbled with $O(\log^2 n)$ pebbles (proving item 1).

By item 2 in Theorem D.4.22 and by choosing $\epsilon = 1/(\delta \log\log n)$ in Theorem D.4.5 we conclude that G_n can be reversibly pebbled in simultaneous time $O(n2^{1/\delta})$ and space $O(n/(\delta \log^{1-\delta} n))$, from which item 2 follows. Finally, item 3 in the theorem follows from item 3 in Theorem D.4.22. □

D.4.4 Permutation Graphs

Another family of graphs that has been studied in the context of standard pebbling trade-offs is that of permutation graphs.

Definition D.4.23. Given a permutation $\sigma \in \mathfrak{S}([n])$, the *permutation graph* $G(\sigma)$ consists of two lines $(x_1, \ldots, x_n)$ (the *bottom line*) and $(y_1, \ldots, y_n)$ (*top line*) which are connected as follows: for every $1 \leq i \leq n$, there is an edge from x_i to $y_{\sigma(i)}$.

Lengauer and Tarjan [LT82] proved that permutation graphs present the following smooth trade-off when instantiated with the permutation that reverses the binary representation of the index i (see Figure D.3 for an illustration).

Theorem D.4.24 ([LT82]). *Let G be a bit-reversal permutation graph on $2n$ vertices. For any $3 \leq s \leq n$, there is a standard pebbling in space s and time $O(n^2/s)$. Moreover, any standard pebbling $\mathcal{P}$ in space s satisfies $\mathsf{time}(\mathcal{P}) = \Omega(n^2/s)$.*

We show that these graphs also present a smooth reversible pebbling trade-off and, in particular, for $s = n^{\Theta(1)}$ and $s \leq n$, any reversible pebbling $\mathcal{P}$ in space s satisfies $\mathsf{time}(\mathcal{P}_n) = \Omega(n^2/s)$ and there are matching upper bounds. To this end, we use the following proposition.

Proposition D.4.25. *For every natural number k, the line graph over n vertices can be reversibly pebbled using $2k \cdot n^{1/k}$ pebbles in time $2^k \cdot n$.*

Proof. Observe that the line graph over n can be pebbled (in the standard way) using 2 pebbles in time $2n$. The proposition follows now by applying Theorem D.4.5 with $\epsilon = k/\log(n)$. □

Using Proposition D.4.25, we obtain the following result.

Theorem D.4.26. *Let G_n be a bit-reversal permutation graph on $2n$ vertices. Then G_n satisfies the following properties:*

1. *The reversible pebbling price of G_n is at most $2\log n + 2$.*
2. *If s satisfies $4\log n \leq s \leq 2n$ and k is such that $s = 4kn^{1/k}$, then there is a reversible strategy in simultaneous space s and time $\mathrm{O}\big(k2^{2k} \cdot n^2/s\big)$.*
3. *Any standard pebbling $\mathcal{P}_n$ of G_n satisfies $\mathsf{time}(\mathcal{P}_n) = \Omega\big(n^2/\mathsf{space}(\mathcal{P}_n)\big)$.*

Proof. The upper bounds (item 1 and item 2) hold for any permutation graph.

For item 1, we can simulate a reversible pebbling of the top line that uses space at most $\log n + 1$ (as per Proposition D.4.12), and every time we need a pebble on a vertex v of the bottom line in order to place or remove a pebble on the top line, we reversibly pebble the bottom line until v is pebbled (can be done with $\log n + 1$ pebbles), make the move on the top line, and then unpebble the bottom line.

To obtain item 2, we consider a two stage strategy. In the first phase, we place $n^{1/k}$ pebbles spaced equally apart on the bottom line. We refer to these pebbles as *fixed* pebbles, since they will remain on the graph until the sink is pebbled. In the second phase, we simulate a reversible pebbling on the top line with $2kn^{1/k}$ pebbles and every time we need a pebble on a vertex v on the bottom line to make a move on the top line, we reversibly pebble v (with $2(k-1)n^{1/k}$ pebbles) from the nearest fixed pebble, make the move on the top line, and then unpebble the segment on the bottom line.

All that is left to show is that this can be done within the space budget of $4kn^{1/k}$ in time $\mathrm{O}(2^{2k} \cdot n^2/s)$. For the first phase, we reversibly pebble $n^{1/k}$ segments of length $m = n^{1-1/k}$. By Proposition D.4.25, each of the segments can be reversibly pebbled using $2(k-1)n^{1/k} = 2(k-1)m^{k-1}$ pebbles in time $2^{k-1}n^{1-1/k}$. Since every segment must be pebbled and then unpebbled, the total time for the first phase is $2 \cdot 2^{k-1}n^{1-1/k} \cdot n^{1/k} = 2^k n$, and the total number of pebbles used is less than $2kn^{1/k}$: $n^{1/k}$ for the fixed pebbles and $2(k-1)n^{1/k}$ for pebbling each segment.

We turn to analyze the second phase. By Proposition D.4.25, the top line can be reversibly pebbled in simultaneous space $2kn^{1/k}$ and time $2^k n$. For each move in the top line, we need to pebble and unpebble a segment of length at most $n^{1-1/k}$. As argued before, this can be done in simultaneous space $2(k-1)n^{1/k}$ and time $2 \cdot 2^{k-1}n^{1-1/k}$. Therefore, at any point in the pebbling strategy there are at most $2kn^{1/k}$ pebbles on the bottom line and at most $2kn^{1/k}$ pebbles on the top line, and the total time of the pebbling is at most $2^k n + 2^{2k}n^{2-1/k} \leq 4k2^{2k}n^2/s$.

Finally, we observe that item 3 follows from the standard pebbling lower bound in Theorem D.4.24. □

From Theorem D.4.26 we obtain the following corollary that, together with Theorem D.3.1, implies Theorem D.4.4.

Corollary D.4.27. *Any reversible pebbling strategy $\mathcal{P}_n$ for G_n that optimizes time given space constraint $n^{\Theta(1)}$ (and less than n) exhibits a trade-off* $\mathsf{time}(\mathcal{P}_n) = \Theta\big(n^2/\mathsf{space}(\mathcal{P}_n)\big)$.

D.5 Concluding Remarks

In this paper we prove that size and degree of Nullstellensatz refutations in any field of pebbling formulas are exactly captured by time and space of the reversible pebble game on the underlying graph. This allows us to prove a number of strong size-degree trade-offs for Nullstellensatz. To the best of our understanding no such results have been known previously.

The most obvious, and also most interesting, open question is whether there are also size-degree trade-offs for the stronger polynomial calculus proof system. Such trade-offs cannot be exhibited by the pebbling formulas considered in this work, since such formulas have small-size low-degree polynomial calculus refutations, but the formulas exhibiting size-width trade-offs for resolution [Tha16] appear to be natural candidates.

Another interesting question is whether the tight relation between Nullstellensatz and reversible pebbling could make it possible to prove even sharper trade-offs for size versus degree in Nullstellensatz, where just a small constant drop in the degree would lead to an exponential blow-up in size. Such results for pebbling time versus space are known for the standard pebble game, e.g., in [GLT80]. It is conceivable that a similar idea could be applied to the reversible pebbling reductions in [CLNV15], but it is not obvious whether just adding a small amount of space makes it possible to carry out the reversible pebbling time-efficiently enough.

Finally, it can be noted that our results crucially depend on that we are in a setting with variables only for positive literals. For polynomial calculus it is quite common to consider the stronger setting with "twin variables" for negated literals (as in the generalization of polynomial calculus in [CEI96] to *polynomial calculus resolution* in [ABRW02]). It would be nice to generalize our size-degree trade-offs for Nullstellensatz to this setting, but it is not obvious whether the reductions in the current work could be made to work or not.

Acknowledgements

We are grateful for many interesting discussions about pebbling-related (and not-so-pebbling-related) matters with Toniann Pitassi, Marc Vinyals, and Arkadev Chattopadhyay.

This work was mostly carried out while the authors were visiting the Simons Institute for the Theory of Computing in association with the DIMACS/Simons Collaboration on

Lower Bounds in Computational Complexity, which is conducted with support from the National Science Foundation.

Or Meir was supported by the Israel Science Foundation (grant No. 1445/16). Robert Robere was supported by NSERC, and also conducted part of this work at DIMACS with support from the National Science Foundation under grant number CCF-1445755. Susanna F. de Rezende and Jakob Nordström were supported by the *Knut and Alice Wallenberg* grant KAW 2016.0066 *Approximation and Proof Complexity*. Jakob Nordström was also supported by the Swedish Research Council grants 621-2012-5645 and 2016-00782.

E

Paper E

Cumulative Space in Black-White Pebbling and Resolution

Joël Alwen, Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals

Full length version of the article published in Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS '17), January 2017, pp. 38:1–38:21.

Abstract

We study space complexity and time-space trade-offs with a focus not on peak memory usage but on overall memory consumption throughout the computation. Such a cumulative space measure was introduced for the computational model of parallel black pebbling by [Alwen and Serbinenko '15] as a tool for obtaining results in cryptography. We consider instead the nondeterministic black-white pebble game and prove optimal cumulative space lower bounds and trade-offs, where in order to minimize pebbling time the space has to remain large during a significant fraction of the pebbling.

We also initiate the study of cumulative space in proof complexity, an area where other space complexity measures have been extensively studied during the last 10–15 years. Using and extending the connection between proof complexity and pebble games in [Ben-Sasson and Nordström '08, '11], we obtain several strong cumulative space results for (even parallel versions of) the resolution proof system, and outline some possible future directions of study of this, in our opinion, natural and interesting space measure.

E.1 Introduction

The time and space complexity measures are at the heart of understanding computation. Unfortunately, there is little we can say about general computation models such as Boolean circuits, let alone Turing machines. But if we allow ourselves to work with simpler models of computation, then we have a better chance at understanding these resources, and in fact there has been impressive progress in restricted models like bounded-depth circuits.

One of the first success stories in this direction are pebble games. The original *(black) pebble game* is played by a single player on a directed acyclic graph (DAG) with a single sink and all vertices having bounded indegree and consists of two simple rules:

1. we can add a pebble to a vertex if all its direct predecessors have pebbles, and

2. we can remove a pebble from a vertex at any time.

The goal of the game is to place a pebble on the sink of the graph. Time is measured as the number of moves to reach this goal, and the space is the maximum number of pebbles needed simultaneously at any point during the pebbling.

Quite surprisingly, this seemingly simple and innocent game can be used to obtain strong results even for general computation models, as it is at the core of the $\mathsf{DTIME}(t) \subseteq \mathsf{SPACE}(t/\log t)$ space upper bound for Turing machines in [HPV77]. Pebbling was first used in [PH70] to study flowcharts and recursive schemata, and different variants of the game have later been applied to a rich selection of problems in computer science, including register allocation [Set75], algorithmic time and space trade-offs [Cha73], parallel time [DT85], communication complexity [RM99], monotone space complexity [CP14, FPRC13], cryptography [AS15, DNW05], and proof complexity [BN08, BW01, BEGJ00] (where it should be emphasized that the above list of references is far from exhaustive). An excellent overview of pebbling up to ca 1980 is given in [Pip80] and another in-depth treatment of some pebbling-related questions can be found in chapter 10 of [Sav98]. Some more recent developments are discussed in the upcoming survey [Nor19].

Let us briefly discuss what is known about space in proof complexity, since this is one of the two topics we are focusing on in this paper. The study of space in proof complexity was initiated in [ET01], which introduced the *clause space* measure for the well-known resolution proof system, a measure that has subsequently been thoroughly investigated. Informally, the clause space of a resolution proof can be defined as the maximal number of additional clauses—on top of the clauses in the original CNF formula—that a verifier needs to keep in memory at any time while checking the correctness of the proof.[1] While some formulas have proofs requiring only a small, sometimes even just constant, space overhead during verification, other formulas require a linear amount of extra space [ABRW02, BG03, ET01], and as shown in [ET01] no formula requires more than linear clause space in resolution.

Other papers have studied how space relates to other proof complexity measures. With respect to proof length, which can be viewed as a measure of (nondeterministic) running time, there is a wide range of trade-off results. It has been shown that there are

[1]Though slightly different from the definition in [ET01], this is equivalent up to a small additive constant.

formulas which have both short and space-efficient proofs, but as one of these measures is optimized the other one can blow up to almost worst-case behaviour [BN11]. Not only this, but there are even formulas where short proofs require more than the worst-case linear space [BBI16, BNT13]. Yet other papers have studied other space measures such as *total space* [ABRW02, Bon16, BGT14], measuring the total number of symbols in a proof, and space complexity has also been considered for other proof systems than resolution. We refer the reader to the survey [Nor13] for more details (although, for obvious reasons, it fails to cover the very latest results on total space).

All the space measures discussed above have in common the fact that they refer to the maximum space used at some point in the proof, but they are far from providing a complete picture of space usage during the whole proof. If we only know that a formula has high space complexity, it is not possible to distinguish between a formula that requires large space only at the beginning of the proof, say, and another that requires large space throughout the whole proof. This distinction might not be so important if we are considering the memory requirements of a verifier, since in this case we are chiefly interested in the maximum. However, it could be relevant for proof search: an algorithm that searches for a proof by producing clauses needs to discard many of them or risk exhausting its available memory. In this case, the difference between needing large space once versus at all times is the difference between making one lucky choice of which clauses to keep in memory versus being lucky all the time.

A similar issue occurs with so-called *memory-hard functions* in the context of cryptography. The idea behind memory-hard functions is that they should require a large amount of memory to evaluate, so that in order to compute such a function for many inputs as a part of a brute-force attack either an infeasible amount of memory is needed or the attack needs to be carried out sequentially. Yet, if the function only requires a large amount of memory during a limited time of the computation, then it is possible to reuse memory for different computations overlapping suitably in time as observed in [AB16]. Therefore, a more appropriate measure to analyse memory-hard functions is *cumulative space complexity* as introduced in [AS15], where one measures not the maximum memory consumption but the total memory usage aggregated over the time of the computation.

Although with hindsight this cumulative space complexity measure appears to be a very natural way of quantifying memory usage, it does not seem to have received too much attention in computational complexity theory, and to the best of our knowledge it has not been considered at all in the context of proof complexity. One of the main contributions of this paper is to transfer the concept of cumulative space to proof complexity and to initiate a study of this complexity measure for the resolution proof system.

Pebble games turn out to be a useful tool also for analysing cumulative space. For pebbling strategies cumulative space is straightforwardly defined as the sum over all steps of the pebbling of the number of pebbles on the DAG at each point in time. Thus, in the standard pebble game discussed above any DAG with n vertices can trivially be pebbled in time n and cumulative space $\mathrm{O}(n^2)$ by placing pebbles on all vertices in topological order. Since every vertex needs to be pebbled at some point, a trivial lower bound for the cumulative space is n. However, depending on the intended application one needs to consider other variations of this pebble game as discussed next.

In a proof complexity setting we need to study the *black-white pebble game*, which was introduced in [CS76] with the objective of modelling nondeterministic computations. Here white pebbles, corresponding to nondeterministic guesses, can be placed at any vertex at any time, but a white pebble can only be removed from a vertex when all direct predecessors have (black or white) pebbles, corresponding to that the correctness of the nondeterministic guess can be verified.

To model parallel computation in a cryptographic setting, [AS15] introduced yet another pebble game, namely the *parallel (black) pebble game*. In this game, all the pebbling moves that are legal at some point in time can be performed simultaneously in one single step. This change of rules does not affect the maximal space required to pebble a DAG, but typically changes the pebbling time. Any connected DAG with a single sink requires linear time to pebble sequentially, but for a parallel pebbling it is easy to see that the time required is upper-bounded by the depth of the graph (i.e., the length of a longest path). We remark that an attractive feature of parallel pebbling is that it better captures the difference between maximal and cumulative space. Note that in any sequential pebbling game placing s pebbles requires s time steps, and during the last $s/2$ steps there will be at least $s/2$ pebbles on the DAG. Thus, any pebbling in maximal space s requires cumulative space $\Omega\big(s^2\big)$. In contrast, in a parallel pebbling the cumulative space can be small even when the maximal space is large.

E.1.1 Our Pebbling Contributions

In this paper, we study the cumulative space measure in the context of black-white pebbling. In order to do so, we also extend black-white pebbling to a parallel version. As pebble games go, this is a very powerful model, since it turns out that any DAG can be pebbled with a parallel black-white pebbling in constant time and linear cumulative space. Perhaps somewhat surprisingly, however, it is still possible to prove nontrivial time-space trade-offs. It can be shown that the parallel and sequential versions of black-white pebbling are closely connected (as discussed in more detail later in the paper), and therefore in this overview the exposition is focused on sequential black-white pebbling.

The first question we address is how the large cumulative space can be in the worst case for sequential black-white pebbling. As noted above, a trivial (black-only) pebbling in linear time and space has cumulative space $\mathrm{O}\big(n^2\big)$ for any graph over n vertices. In the other direction, the $\Omega(n/\log n)$ space lower bound in [GT78] already gives a $\Omega(n^2/\log^2 n)$ cumulative space lower bound for sequential black-white pebbling, as explained above. One cannot get a better cumulative space lower bound by this simple argument from maximal space lower bounds, however, since any DAG of constant indegree can be pebbled in maximal space $\mathrm{O}(n/\log n)$ [HPV77].

We prove that the family of *grate graphs* in [Sch83] require $\Omega(n^2)$ cumulative space for sequential black-white pebbling. This shows that for cumulative space it is not possible to improve on the trivial quadratic upper bound, in contrast to the maximal space measure where it is always possible to save a logarithmic factor from the trivial linear upper bound. This is also different from the parallel black pebble game, where there is a $o(n^2)$ worst-case upper bound for cumulative space [AB16] and the best known cumulative space lower bound is $\Omega\big(n^2/\log n\big)$ [ABP16]. In fact, it turns out that the difference between the sequential black-white and parallel black pebble games can be very large. We also prove that (a modified version of) the *butterfly graphs* in [SS77] require cumulative space $\Omega(n^2/\log n)$ in the sequential black-white pebble game but can be pebbled in linear cumulative space in the parallel black pebble game. Butterfly graphs also show that graphs that require large cumulative space do not necessarily require large maximal space, as they have logarithmic depth and thus can be pebbled in logarithmic space as observed in [HPV77]. We obtain these results by studying the lower bounds on cumulative space in parallel black pebbling in [ABP16] in terms of *depth-robustness* of graphs, and extending these lower bounds to other pebble games and other families of graphs.

Our next set of results concern trade-offs between time and space. Here our starting point is the family of *bit-reversal permutation graphs* studied in [LT82] which can be pebbled either with 3 pebbles or (as any graph) in linear time, but for which any pebbling in time t and space s must satisfy $t = \Omega(n^2/s^2)$, where as before n is the number of vertices in the graph.

We strengthen this trade-off to cumulative space, proving that pebblings of these graphs in space s require cumulative space $\Omega\big(n^2/s\big)$, which in particular implies that a pebbling in time $O\big(n^2/s^2\big)$ must use space $\Omega(s)$ not only at some point but most of the time.[2] Furthermore, we establish an unconditional $\Omega\big(n^{3/2}\big)$ cumulative space lower bound, which provides another example of graphs that require (at least somewhat) large cumulative space but can be pebbled in very small (even constant) maximal space. Our proofs of these results work by adapting the *dispersion* technique from [ABP16]. This technique has the advantage that it isolates an abstract combinatorial property of the graph that makes the lower bound argument go through, and this cleaner approach enables us to prove these results not only for bit-reversal graphs but also for *random permutation graphs* (by showing that these graphs possess the required combinatorial property with high probability). To the best of our knowledge no trade-offs (even non-cumulative ones) were known for such graphs before for any flavour of the pebble game.

Finally, we consider a very concrete, extremal question regarding pebbling time-space trade-offs. It is an easy observation that any sequential black-white pebbling in constant space s can be carried out in time $O\big(n^s\big)$, since there are only $\sum_{k=0}^{s} 2^k\binom{n}{k}$ possible different configurations of s pebbles in the graph, and no configuration repeats

[2]Note, importantly, that such a space lower bound is *not* implied by the simple "space s implies cumulative space $\Omega\big(s^2\big)$" argument discussed previously.

in a pebbling (or else the intermediate moves can be removed). In fact, a bit more thought reveals that this time bound can be sharpened to $\mathrm{O}\big(n^{s-1}\big)$, since every configuration in space s is immediately followed by a pebble removal, and so we only need to consider distinct configurations of $s-1$ pebbles. It is a natural question whether this simple counting argument is in fact tight, so that there are graphs that can be pebbled in space s but where any such pebbling requires time $\Omega\big(n^{s-1}\big)$.

For pebbling space $s = 3$, the minimum space in which any nontrivial pebbling strategy is possible, the bit-reversal graphs in [LT82] discussed above show that the answer to this question is affirmative. It is not hard to see that by stacking $s-2$ bit-reversal DAGs on top of one another, identifying the top layer in one graph with the bottom layer in the graph above, one obtains graphs that can be pebbled in space s but where the obvious pebbling strategy achieving this bound requires time $\mathrm{O}\big(n^{s-1}\big)$. We prove that this trivial upper bound is indeed asymptotically tight for any constant s.

E.1.2 Our Proof Complexity Contributions

Turning now to proof complexity, we consider the main contribution of our paper to be that we initiate the study of the cumulative space measure. While the concept of cumulative space seems to be as natural as maximal space, we are not aware of it having been studied in the context of proof complexity before. As was the case for the first papers on (maximal) space complexity in resolution [ET01], in this first paper on cumulative space in proof complexity we focus on the resolution proof system.

An immediate observation is that proof length is always a lower bound on cumulative space, and so exponential lower bounds on proof length—as shown for resolution in [CS88, Hak85, Urq87] and many later papers—trivially imply exponential lower bounds on cumulative space. Therefore, it seems that the cumulative space measure will be of independent interest mostly for formulas which have reasonably short proofs. An obvious candidate family to study are pebbling formulas [BW01], which have proofs in linear length, but which exhibit a rich variety of properties with respect to space complexity depending on the underlying graphs in terms of which they are defined.

However, we also need to decide on an appropriate model of the resolution proof system in which to study cumulative space. In the context of pebbling we concluded that cumulative space makes most sense for parallel versions of the pebble games, and so it is natural to ask whether one should consider a parallel version of resolution when studying cumulative clause space. It is not hard to argue that such a parallel model of resolution could be interesting in its own right, since it might be useful as a tool to analyse attempts to parallelize state-of-the-art SAT solvers using so-called *conflict-driven clause learning (CDCL)* [BS97, MS99].

We define and study several different versions of the resolution proof systems with varying degrees of parallelity. The running time of parallel CDCL solvers has previously been analysed using resolution depth and the related *conflict resolution depth* and *sched-*

ule makespan measures introduced in [KSSS13], and our models of parallel resolution allow us to reason about space in addition to time.

Similarly to what is the case for pebble games, our most general model of parallel resolution, where clauses can be inferred not just by syntactic application of the resolution rule but by semantic inference, is extremely powerful, so much so that it can deal with any formula in a constant number of steps and linear space. Since we can establish a tight relation between space and parallel speedup also for resolution, however, we can still obtain lower bounds when the maximal space is limited.

Studying pebbling formulas in these different models of resolution, and revisiting the reductions between resolution and pebble games in [BN08, BN11], we can translate the pebbling results in Section E.1.1 to results for the resolution proof system. Summarizing very briefly, we exhibit different formulas that have

- proofs in linear length but require quadratic cumulative space,
- proofs in logarithmic space but require $\Omega(n^2/\log n)$ cumulative space, and
- trade-offs between proof length and cumulative space.

E.1.3 Paper Outline

The rest of this paper is organized as follows. In Section E.2 we present a more detailed overview of our pebbling results, introducing formal definitions of the pebble games and measures discussed above, and we give an analogous overview for resolution in Section E.3. Section E.4 contains detailed proofs of our pebbling theorems. We conclude in Section E.5 with a discussion of possible directions for future research.

E.2 Pebbling Results Overview

Let us start our pebbling overview by giving formal definitions of the basic concepts.

E.2.1 Definition of Pebble Games and Basic Properties

We say that a directed acyclic graph (DAG) $G = (V, E)$ with $|V| = n$ has *size* n. A vertex $v \in V$ has *indegree* δ if it has δ incoming edges $\{(u_1, v), \dots, (u_\delta, v)\} \subseteq E$, $u_i \neq u_j$ for $i \neq j$, and we say that G has indegree δ if the maximum indegree of any vertex of G is δ. A vertex with no incoming edges is called a *source* and a vertex with no outgoing edges is called a *sink*. We say that a vertex u is a *predecessor* of a vertex v if there exists a directed path from u to v; moreover, if this path consists of only one edge then u is a *direct predecessor* of v. We denote by $\mathsf{parents}(v)$ the set of all direct predecessors of v. For technical reasons, it will sometimes be convenient to allow paths of length 0 in the definition above, so that a vertex can be a predecessor of itself. We will sometimes consider graphs obtained from other graphs by removing subsets of vertices, and for

$U \subseteq V$ we write $G - U = \big(V \setminus U, E \setminus ((U \times V) \cup (V \times U)\big)$ to denote the DAG obtained from G by removing the vertices in U and all edges incident to U.

To get a unified description of all flavours of the pebble game discussed in Section E.1, it is convenient to define pebbling as follows.

Definition E.2.1 (Pebble games). Let $G = (V, E)$ be a DAG with a unique sink vertex z. The black-white pebble game on G is the following one-player game. At any time i, we have a *black-white pebbling configuration* $\mathbb{P}_i = (B_i, W_i)$ of black pebbles B_i and white pebbles W_i on the vertices of G, at most one pebble per vertex. The rules of how a pebble configuration $\mathbb{P}_{i-1} = (B_{i-1}, W_{i-1})$ can be changed to $\mathbb{P}_i = (B_i, W_i)$ are as follows:

1. A black pebble may be placed on a vertex v only if all immediate predecessors of v are covered by pebbles in both $\mathbb{P}_{i-1}$ and $\mathbb{P}_i$, i.e.,

$$v \in (B_i \setminus B_{i-1}) \Rightarrow \mathsf{parents}(v) \subseteq \mathbb{P}_{i-1} \cap \mathbb{P}_i \ .$$

   Note that, in particular, a black pebble can always be placed on a source vertex.

2. A black pebble on any vertex v in $\mathbb{P}_{i-1}$ can be removed in $\mathbb{P}_i$.

3. A white pebble can be placed on any vertex v in $\mathbb{P}_i$.

4. A white pebble on a vertex v in $\mathbb{P}_{i-1}$ may be removed in $\mathbb{P}_i$ only if all immediate predecessors of v are covered by pebbles in both $\mathbb{P}_{i-1}$ and $\mathbb{P}_i$, i.e.,

$$v \in (W_{i-1} \setminus W_i) \Rightarrow \mathsf{parents}(v) \subseteq \mathbb{P}_{i-1} \cap \mathbb{P}_i \ .$$

   In particular, a white pebble can always be removed from a source vertex.

A *legal pebbling* $\mathcal{P}$ of G is a sequence $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_t)$ where every configuration $\mathbb{P}_i$ can be obtained from $\mathbb{P}_{i-1}$ using the rules 1–4. A *complete pebbling* is a legal pebbling where $\mathbb{P}_0 = \mathbb{P}_t = (\emptyset, \emptyset)$ and $z \in \bigcup_{i=0}^{t}(B_i \cup W_i)$ (i.e., the sink is pebbled at some point).

A *black pebbling* is a pebbling where $W_i = \emptyset$ for all $i \in [t]$. A pebbling is *sequential* if at most one application of a single rule 1–4 is used to get from from $\mathbb{P}_{i-1}$ to $\mathbb{P}_i$ for all $i \in [t]$. In a *(fully) parallel* pebbling an arbitrary number of applications of the rules 1–4 can be made to $\mathbb{P}_{i-1}$ to obtain $\mathbb{P}_i$ (but note that all pebble placements and removals have to be legal with respect to $\mathbb{P}_{i-1}$, and cannot make use of any pebble placements or removals made in parallel). Finally, we also consider *parallel-black sequential-white* pebblings, which allows parallel applications of black pebble rules 1–2 to $\mathbb{P}_{i-1}$ to obtain $\mathbb{P}_i$, but only a single application of the white pebble rules 3–4. Note that, in the parallel setting, a simultaneous application of rules 1 and 4 on a same vertex replaces a white pebble by a black one.

The *time* of a pebbling $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_t)$ is $t(\mathcal{P}) = t$; the (maximal) *space* is $s(\mathcal{P}) = s = \max_{i \in [t]} |B_i| + |W_i|$; and the *cumulative space* is $c(\mathcal{P}) = c = \sum_{i \in [t]} |B_i| + |W_i|$ (where we observe that $c \leq st$).

Parallel black pebbling was introduced in [AS15], where it was pointed out that for certain graphs parallel pebblings can be much more efficient than sequential, while for others they cannot do any better. For example, if we are considering time-space tradeoffs, any sequential black pebbling in space s and time t of a bit-reversal graph must satisfy $st = \Omega\big(n^2\big)$ [LT82], while in the parallel black game one can pebble such graphs in linear time and space $\mathrm{O}(\sqrt{n})$ [AS15]. In contrast, it was shown in [ABP16] that there are graphs that can be pebbled sequentially in space s and time t satisfying $st = \mathrm{O}\big(n^2/\log n\big)$, but where these graphs even in the parallel model require not only $st = \Omega\big(n^2/\log n\big)$ but also cumulative space $\Omega\big(n^2/\log n\big)$.

Unlike the case of the black pebble game, we show that time and space in the black-white sequential and parallel games are closely related. Up to constant factors, it holds that if a parallel black-white pebbling $\mathcal{P}$ has maximal space s, then it is possible to save a factor s, but not more than a factor s, in time compared to a sequential black-white pebbling in the same space s.

Observation E.2.2. *Let $\mathcal{P}$ be a parallel black-white pebbling of a DAG G in time t, space s, and cumulative space c. Then there is a sequential black-white pebbling of G in time $2ts$, space $2s$, and cumulative space cs.*

Proof. Each parallel move places at most s pebbles and removes at most s pebbles, therefore we can simulate it by $2s$ sequential moves (making the pebble placements first, to make sure that these moves remain legal). ☐

Lemma E.2.3. *Let $\mathcal{P}$ be a sequential black-white pebbling of G in time t, space s, and cumulative space c, and let k be a positive integer. Then there is a parallel black-white pebbling of G in time $3(\lceil t/k \rceil + 1)$, space $s + \lfloor k/2 \rfloor$, and cumulative space $3\lfloor c/k \rfloor + t$.*

Proof. Suppose we divide $\mathcal{P}$ into at most $\lceil t/k \rceil + 1$ intervals of at most k moves. We can then reorder the pebbling moves within each of these intervals so that we do all placements first and removals afterwards. This is still essentially a valid pebbling, because each configuration is a superset of the corresponding configuration in $\mathcal{P}$, except that we can possibly have vertices temporarily covered by several pebbles. The space usage in any intermediate configuration increases to at most $s + \lfloor k/2 \rfloor$. We then collapse each subsequence into one parallel placement of white pebbles, one step replacing white pebbles with black pebbles as needed, and one parallel removal of black pebbles. If $\mathbb{P}$ is the last configuration of the subsequence, then the white pebbles that are replaced by black are the ones that are either black in $\mathbb{P}$ or that or not present at all in $\mathbb{P}$, that is, only the pebbles that are white in $\mathbb{P}$ are not replaced by black pebbles. This allows us to make all black pebble placements in parallel even though later black pebbles might be dependent on earlier pebble placements in the sequential pebbling, and similarly remove all white pebbles in parallel even though there might be dependencies inside the interval. The total time decreases to $3(\lceil t/k \rceil + 1)$. Note that this holds for any partition of $\mathcal{P}$ into at most $\lceil t/k \rceil + 1$ intervals of at most k moves.

To bound the cumulative space we need to specify which partition of $\mathcal{P}$ we consider. First note that if a configuration $\mathbb{P}_i$ of $\mathcal{P}$ has space s_i and we consider a sequential interval (of any length) starting at $i+1$ where x_j pebble placements are made, then the three parallel configurations corresponding to this interval have aggregate space at most $3s_i + 2x_j$. Now consider the following k-partition of configurations of $\mathcal{P}$: $\{\mathbb{P}_i, \mathbb{P}_{i+k}, \ldots, \mathbb{P}_{i+k(\lfloor (t-i)/k \rfloor)}\}$ for $i \in [1,k]$. Note that each part consists of at most $\lceil t/k \rceil$ configurations, evenly spaced. By an averaging argument, at least one of these k parts, say the i^*-th part, has cumulative space of at most $\lfloor c/k \rfloor$. We can now divide $\mathcal{P}$ into at most $\lceil t/k \rceil + 1$ intervals of at most k moves, where the 1st interval starts at $\mathbb{P}_1$ and ends at $\mathbb{P}_{i^*}$, the $(j+1)$-st interval starts at $\mathbb{P}_{i^*+(j-1)k+1}$ and ends at $\mathbb{P}_{i^*+jk}$, for $j = 1, 2, \ldots, \lfloor (t-i^*)/k \rfloor$ and if $\mathbb{P}_{i^*+k(\lfloor (t-i^*)/k \rfloor)}$ is not the last configuration, then there is one last interval which starts at $\mathbb{P}_{i^*+k(\lfloor (t-i^*)/k \rfloor)+1}$ and ends at $\mathbb{P}_t$. Let x_j be the number of placements in the jth interval and let s_j be the space of the configuration right before the beginning of the jth interval. Note that $\sum_{j \in [\lceil t/k \rceil + 1]} s_j \leq \lfloor c/k \rfloor$. The total cumulative space is therefore at most $\sum_{j \in [\lceil t/k \rceil + 1]} (3s_j + 2x_j) = 3 \sum_{j \in [\lceil t/k \rceil + 1]} s_{i+j} + 2 \sum_{j \in [t/k-1]} x_j \leq 3 \lfloor c/k \rfloor + 2t/2$. □

Observe that when $k = \Theta(s)$ the cumulative space in Lemma E.2.3 is dominated by the term t, so we only save a factor s in cumulative space when the sequential pebbling has cumulative space $c = \Theta(st)$. Since the graphs we will discuss in what follows have cumulative space lower bounds of this form, studying the sequential game already gives us all the information we want about the parallel game.

Corollary E.2.4. *Let $\mathcal{P}$ be a black-white pebbling of G in time t and space s. Then there is a parallel black-white pebbling of G in time at most $\lceil t/2s \rceil + 3$, space $4s$, and cumulative space $2t$.*

E.2.2 Robustness and High Cumulative Space Complexity

We proceed to define the concept of *$\mathcal{G}$-robustness* of graphs, which is inspired by [EGS75, PR80] and which will be central to our work.

Definition E.2.5 ($\mathcal{G}$-robustness). Let $\mathcal{G}$ be a family of DAGs and let $e, d \in \mathbb{N}^+$ be positive integers. We say that a DAG $G = (V, E)$ is (e,d)-*$\mathcal{G}$-robust* if for every subset of vertices $U \subseteq V$ of size at most e there exists a graph $H \in \mathcal{G}$ of size at least d that is a subgraph of $G - U$.

When $\mathcal{G}$ is the class of directed paths, then we say that G is *depth-robust*, and when $\mathcal{G}$ is the class of DAGs with one sink the DAG G is said to be *predecessor-robust*.[3]

For our pebbling lower bounds we are interested in graphs with very high robustness, i.e., for as large values of e and d as possible. Depth-robustness was first studied by

[3]This choice of terminology is inspired by [PR80], which discusses the dual notions of "depth-separators" and "predecessor-separators."

Erdős, Graham and Szemerédi [EGS75] who showed how to construct DAGs with indegree $\Theta(\log(n))$ possessing $(\Omega(n), \Omega(n))$-depth-robustness. However, in our applications it is important that the graphs have *constant* indegree. Valiant [Val77] showed that for constant indegree and linear depth the best we can hope for is $(O(n/\log n), O(n))$-depth-robustness. Fortunately for us, it was shown in [ABP16, PR80] that such extremal $(\Theta(n/\log n), \Theta(n))$-depth-robust graphs do exist. Conversely, if we want constant indegree with the parameter e linear in the graph size, then $(\epsilon n, n^{1-\epsilon})$-depth-robustness is the best we can hope for [Val77]. In [Sch83] a family of $(\Theta(n), \Theta(n^{1-\epsilon}))$-depth-robust graphs with constant indegree were presented.

The connection between depth-robustness and cumulative space was first made in [ABP16], where it was shown that an (e,d)-depth-robust graph requires parallel black cumulative space at least ed. In this work we give a more general theorem of this form for the case of $\mathcal{G}$-robustness. We then use this theorem to obtain the following lower bounds for depth-robust and predecessor-robust graphs.

Corollary E.2.6. *If G is an (e,d)-depth-robust DAG, then G requires sequential black-white cumulative space at least ed, and parallel-black sequential-white cumulative space at least $e\sqrt{d}$.*

Corollary E.2.7. *If G is an (e,d)-predecessor-robust DAG, then G requires sequential black-white cumulative space at least ed.*

Focusing on the range of parameters discussed above, we can see that it follows from Corollaries E.2.6 and E.2.7 that a $(\Theta(n/\log n), \Theta(n))$-depth-robust graph has sequential black-white cumulative space complexity $\Omega(n^2/\log n)$ and parallel-black sequential-white pebbling cumulative space complexity $\Omega(n^{3/2}/\log n)$.

A class of DAGs that are predecessor-robust are *grates*—graphs with n' sources and n' sinks such that after the removal of an arbitrary set of kn' vertices (for some constant k) there are still a linear number of sources and sinks that are all pairwise connected.[4] *Butterfly graphs* [SS77] are grates with $n = n' \log n'$ vertices that are $(\Theta(n/\log n), \Theta(n/\log n))$-predecessor-robust. Moreover, it is not hard to show that if we append n' single-sink DAGs of size $\log n'$, one to each source of the butterfly graph, the resulting graph is $(\Theta(n/\log n), \Theta(n))$-predecessor-robust. This implies that these graphs require sequential black-white cumulative space $\Omega(n^2/\log n)$. Note that butterfly graphs (also in the modified version just described) can be pebbled with $O(\log n)$ pebbles (since the graphs have depth $O(\log n)$), and thus it is not the case that high cumulative space implies large maximal space.

Theorem E.2.8. *Butterfly graphs of size n can be black pebbled with $O(\log n)$ pebbles but require sequential black-white cumulative space $\Omega(n^2/\log n)$.*

[4]Strictly speaking, grates have multiple sinks and so do not conform to the DAG requirements in Definition E.2.1. However, it is easy to turn any multi-sink DAG of interest into a single-sink DAG with essentially the same properties—we refer to Section E.4 for the details—and so we ignore this technicality here.

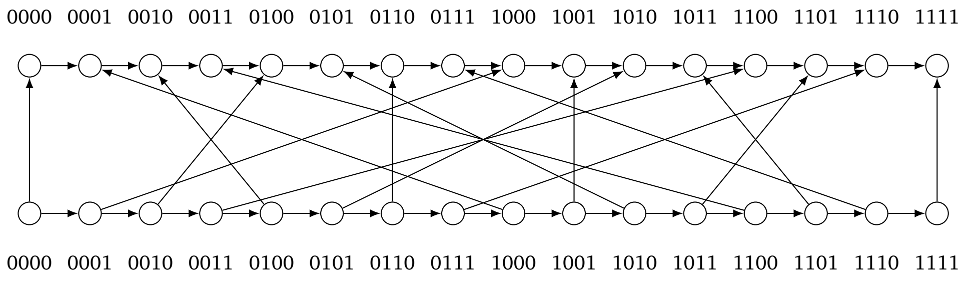


Figure E.1: A bit reversal permutation graph

It has been established that extremal depth-robustness is both a necessary [AB16] and sufficient [ABP16] condition to have high cumulative space in the parallel black game. In particular, using the fact that no graph of size n with constant indegree is $(\omega(n/\log n), \Theta(n))$-depth-robust, it was shown in [AB16] that any constant-indegree graph has parallel black cumulative space complexity $o\big(n^2/\log^{1-\epsilon} n\big)$, for any constant $\epsilon > 0$. A natural question is if this also holds for black-white pebbling. We show that this is not the case: there are graphs that have maximum cumulative space complexity $\Omega(n^2)$ in the black-white pebble game. This follows from Corollary E.2.7 and the existence of grates of size linear in the number of sources and sinks [Sch83].

Theorem E.2.9. *There are graphs of size n that require sequential black-white cumulative space $\Omega(n^2)$.*

E.2.3 Dispersion and Cumulative Space Trade-Offs

Another property of graphs that is important in the current paper is *dispersion*. This notion was used in [ABP16] to obtain another condition ensuring high parallel black cumulative space complexity. We define two similar concepts and then use them to obtain cumulative space trade-offs. The results we get are for two classes of *permutation graphs*—graphs that consist of two ordered paths of vertices, where in addition an edge is added from each vertex in the first path to its image under some specified permutation σ in the second path.

A family of permutations that will be of particular interest to us are the so-called *bit-reversal permutations*, which are defined for $n = 2^m$ and which simply reverse the binary representations of numbers. That is, if $j = (b_1 \cdots b_m)_{(2)}$, then the bit-reversal permutation σ sends j to $\sigma(j) = (b_m \cdots b_1)_{(2)}$ (see Figure E.1). It was previously known [LT82] that any sequential black-white pebbling of a bit-reversal permutation graph on $2n$ vertices in time t and space s satisfies $st = \Omega\big(n^2/s\big)$. Moreover, it was shown in [LT82] that this is tight up to constant factors and that there is a black-white pebbling in time t and space s such that $st = O\big(n^{3/2}\big)$.

We observe that while bit-reversal graphs are *not* $(2\sqrt{n}, 2\sqrt{n})$-depth-robust, they can be shown to be $(\sqrt{n}, n)$-predecessor-robust. Therefore, in constrast to [ABP16], where it was not possible to establish a parallel black cumulative space lower bound of $n^{3/2}$ using depth-robustness, we are able to obtain a black-white cumulative space lower bound of $n^{3/2}$ using predecessor-robustness.

Our reason for studying dispersion properties of bit-reversal graphs is to characterize how cumulative space increases when space decreases. We show that the time-space trade-off in [LT82] can be strengthened to a cumulative space trade-off. Our result implies that if $\mathcal{P}$ is a sequential black-white pebbling of a bit-reversal graph in space s and time n^2/s^2, then it needs to use space s not only at some point of the pebbling, but during a large part of the time.

Theorem E.2.10. *If G is a bit-reversal graph on $2n$ vertices, then it holds that in the sequential black-white pebble game G requires cumulative space $\Omega(n^{3/2})$ and any pebbling $\mathcal{P}$ of G in maximal space s has cumulative space $\Omega(n^2/s)$.*

An advantage of our approach is that we identify a general property of graphs that imply cumulative space trade-offs, so that the task of establishing a trade-off reduces to proving that the graph has this desired property. As a consequence of this simplification, we are able to prove the same kind of trade-off results not only for bit-reversal graphs but also for random permutation graphs, a class of graphs for which it seems nothing was known before. We note that a property of graphs is said to hold *asymptotically almost surely* on a random permutation graph on $2n$ vertices if it holds with probability that approaches 1 as n approaches infinity.

Theorem E.2.11. *If G is a random permutation graph on $2n$ vertices, then it holds asymptotically almost surely that in the sequential black-white pebble game G requires cumulative space $\Omega(n^{3/2})$ and any pebbling $\mathcal{P}$ of G in maximal space s has cumulative space $\Omega(n^2/s)$.*

E.2.4 Pebbling in Small Space Can Require Maximum Length

Let us finally consider the question of how long a shortest sequential pebbling of a graph can be, given constraints on the maximal pebbling space. Without loss of generality, a black pebbling in space s takes time at most $\binom{n}{\leq s} \leq n^s$, simply because there is no need to repeat any pebble configuration. A moment of thought reveals that in fact we get the upper bound $\binom{n}{s-1} + \binom{n}{\leq s-1} \leq n^{s-1}$, since every configuration in maximal space s is followed by an erasure yielding a space-$(s-1)$ configuration, and these configurations also do not repeat. For black-white pebbling the upper bound becomes $2^{s-1}\left(\binom{n}{s-1} + \binom{n}{\leq s-1}\right) \leq 2^{s-1}n^{s-1}$.

As discussed in the introduction, it can be read off from [LT82] that for space-3 pebblings the $\mathrm{O}(n^2)$ upper bound is tight up to constant factors—bit-reversal DAGs are examples of graphs for which pebblings in optimal space 3, or indeed any constant space, require quadratic time. We extend this result to any $s = \mathrm{O}(1)$ by exhibiting graphs that can be pebbled in space s but where any such pebbling requires time $\Omega\big(n^{s-1}\big)$. We do this by generalizing permutation graphs to multiple layers, where we have k directed path graphs of length n and $k-1$ layers of permutations between the vertices in consecutive paths (so that the permutation graphs considered in [LT82] are 2-layer bit-reversal graphs with paths of length n). We state two theorems below for the black and black-white sequential pebble games, and just as for the 2-layer graphs in [LT82] our bounds can be stated not just for minimal space but also an arbitrary space parameter s greater than this minimum.

Theorem E.2.12. *Let k be a constant and let B be a k-layer bit-reversal graph with paths of length n. Then for any s such that $k+1 \leq s \leq \sqrt{n}$ there exists a sequential black pebbling of B in space s and time $O(n^k/s^{2k-3})$. Furthermore, every sequential black pebbling of B in space s requires time $\Omega(n^k/s^{2k-3})$.*

Theorem E.2.13. *Let k be a constant and let B be a k-layer bit-reversal graph with paths of length n. Then for any s such that $k+1 \leq s \leq \sqrt{n}$ there exists a sequential black-white pebbling of B in space s and time $O(n^k/s^{2k-2})$. Furthermore, every sequential black-white pebbling of B in space s, requires time $\Omega(n^k/s^{2k-2})$.*

Our proofs of these results are inspired by the reasoning in [LT82] for 2-layer permutation graphs, but we also need to overcome some new challenges. The essence of the argument is that in order to place a pebble on the jth layer we need to do some work on the preceding layer. If we only have two layers the argument ends here, but when we want to apply the argument recursively we need to be more careful. Indeed, placing pebbles on the $(j-1)$st layer will now require placing more pebbles on the $(j-2)$nd layer, but if we choose the order in which we do the pebble placements wisely, we may be able to reuse part of the work in the $(j-2)$nd layer for several pebble placements in the $(j-1)$st layer. We are able to find a strategy to exploit this insight and obtain optimal upper bounds, but also to make the lower bound argument resilient enough to get asymptotically matching lower bounds.

E.3 Cumulative Space for the Resolution Proof System

We now proceed to describe in more detail the proof complexity results in our paper. We start this section by a brief review of some standard proof complexity preliminaries, after which we discuss how to refine the definition of the resolution proof system to be able to make meaningful and precise claims about maximal space and cumulative space. This then allows us to make the connection to the pebbling results in Section E.2 and what proof complexity implications they have.

A *literal* over a Boolean variable x is either x itself (a *positive literal*) or its negation $\neg x$ (a *negative literal*). A *clause* $C = a_1 \lor \cdots \lor a_k$ is a disjunction of literals a_i over pairwise disjoint variables. A *k-clause* is a clause that contains at most k literals. A *CNF formula* $F = C_1 \land \cdots \land C_m$ is a conjunction of clauses and a *k-CNF formula* is a CNF formula consisting of k-clauses. We think of clauses and CNF formulas as sets: order is irrelevant and there are no repetitions.

The standard definition of a *resolution refutation* $\pi : F \vdash \bot$ of an unsatisfiable CNF formula F—or a *resolution proof* for (the unsatisfiability of) F—is as an ordered sequence of clauses $\pi = (D_1, \ldots, D_t)$ such that $D_t = \bot$ is the empty clause containing no literals, and each clause D_i, $i \in [t]$, is either an *axiom* $D_i \in F$ or is derived from clauses D_j and D_k, $j, k < i$, by the *resolution rule*

$$\frac{B \vee x \qquad C \vee \neg x}{B \vee C}\ , \tag{E.1}$$

where we refer to $B \vee C$ as the *resolvent over x* of $B \vee x$ and $C \vee \neg x$.

In order to study space in general, and cumulative space in particular, we refine the above definition into a family of proof systems as follows.

Definition E.3.1 (Resolution). A resolution refutation $\pi : F \vdash \bot$ of a CNF formula F is a sequence of *configurations*, or sets of clauses, $\pi = (\mathbb{C}_0, \ldots, \mathbb{C}_t)$ such that $\mathbb{C}_0 = \emptyset$, $\bot \in \mathbb{C}_t$, and for all $i \in [t]$ we obtain $\mathbb{C}_i$ from $\mathbb{C}_{i-1}$ by applying exactly one of the following type of rules:

Axiom download Add $A \in F$.

Inference Add D derived from clauses in $\mathbb{C}_{i-1}$.

Erasure Remove clauses from $\mathbb{C}_{i-1}$.

We say that a refutation is (a) *sequential* if at every time step we apply the chosen rule exactly once; (b) *inference-parallel* if only one clause can be downloaded but the inference rule can be applied an arbitrary number of times (but always deriving from $\mathbb{C}_{i-1}$); and (c) *fully parallel* (or just *parallel*) if both axiom download and inference rules can be applied an arbitrary number of times (but note that we cannot mix applications of different rules in the same step). Furthermore, a refutation is said to be (1) *syntactic* if inferences use the resolution rule (E.1) and (2) *semantic* if instead any clause D such that $\mathbb{C}_{i-1} \vDash D$ can be inferred immediately.

The *length* of a resolution refutation π is the number of derivation steps t and the *size* is the total number of clauses introduced in downloads and inference steps (counted with repetitions). [5] The *maximal (clause) space*, or just *space*, of π is $\max\{|\mathbb{C}_i| : \mathbb{C}_i \in \pi\}$ and the *cumulative (clause) space* is $\sum_{\mathbb{C}_i \in \pi} |\mathbb{C}_i|$.

Note that Definition E.3.1 yields a total of six different flavours of resolution 1(a)–2(c) depending on the amount of parallelism and on whether inferences are syntactic or

[5]For standard resolution as defined in the literature it is most often the case that the length and size definitions coincide, and we could have achieved this here also by counting only axiom download and inference steps when measuring length. Including also erasure steps seems slightly more natural in the current context, however, and also changes the measure by at most a constant factor 2, which is completely immaterial for our purposes.

semantic. In what follows, we will discuss our motivation for considering these different models and what we can say about them.

A first, general comment is that from a proof complexity point of view we are mainly interested in *syntactic* versions of the proof systems in Definition E.3.1. Strictly speaking, the *semantic* versions are not even propositional proof systems in the sense of Cook and Reckhow [CR79], since we do not know how to verify semantic implications in polynomial time. In any semantic system we can download all axioms in the formula and then derive contradiction in a single inference step, and efficiently verifying such an inference means solving SAT in polynomial time. However, most results on (clause) space in the proof complexity literature actually hold in the stronger semantic setting. For maximal space this is not so surprising, since the semantic and syntactic space measures are within a constant factor of each other [ABRW02], but even for trade-offs one tends to get results in the semantic setting for free (with the notable exceptions of [BBI16, BNT13]).

Syntactic sequential resolution is the standard definition discussed at the beginning of this section (and note that for this version of resolution the length and size measures are essentially the same). A somewhat unsatisfactory feature of this model is that (analogously to what is the case for pebbling) a maximal space lower bound s immediately implies a cumulative space lower bound $\Omega(s^2)$. The reason is completely analogous: since we can only infer one new clause per time step, during the $s/2$ time steps before reaching space s we must have had at least $s/2$ clauses in memory. It turns out, however, that we can actually beat this lower bound in certain settings, and we also remark that cumulative length-space trade-offs do not necessarily follow from such trivial arguments and so make sense even for syntactic sequential resolution.

By allowing parallel application of inference steps we want to try to get away from cumulative space lower bounds that hold only for the trivial reason just discussed. In *syntactic inference-parallel resolution* we therefore allow clauses to be derived in parallel. As it turns out, anything we are currently able to prove for this model we can also establish for the stronger *semantic inference-parallel resolution* system.

We can also go in the other direction from the syntactic sequential model and introduce a parallelism of sorts by studying *semantic sequential resolution*. As already alluded to, this is a very powerful system since any formula can be refuted in linear size and space by downloading all its axioms in a linear number of steps and then deriving contradiction in just one semantic inference step, but nevertheless the space lower bounds and length-space trade-offs in [BN08, BN11] hold in this model, and can in fact be verified to hold even for semantic inference-parallel resolution.

The most challenging models in terms of lower bounds are the fully parallel ones. *Syntactic parallel resolution* could be viewed as a potentially interesting model for proving lower bounds on parallel SAT solvers using conflict-driven clause learning, where one could imagine an arbitrarily large number of solvers producing resolvents in parallel and having perfect access to shared memory. It is not hard to see that if a standard resolution proof is represented as a DAG in the natural way, then syntactic parallel length, which would be a proxy for execution time, is just the depth of this DAG.

In the semantic model, adding also parallel axiom downloads makes the proof system exceptionally powerful, since now any formula can be refuted in constant length 2, linear size, and linear cumulative space. This seems a bit too strong to be really interesting (and can be viewed as a reason for preferring the inference-parallel version described previously). However, even for semantic fully parallel resolution it is still possible to obtain nontrivial trade-off results if the maximal (non-cumulative) space is bounded. In fact, the speed-up from parallelism can be at most proportional to the maximal space. Note that Observation E.2.2 states an analogous fact for black-white pebbling.

Observation E.3.2. *Let π be a semantic parallel resolution refutation of a formula F in length L, maximal clause space s, and cumulative clause space c. Then there is a semantic sequential refutation of F in length Ls, maximal clause space $2s$, and cumulative clause space $2cs$.*

Proof. Each parallel axiom download or inference adds at most s new clauses, therefore we can simulate it by s sequential axiom downloads or inferences respectively. □

Similarly to Lemma E.2.3, it is also the case here that parallelism can indeed obtain a speed-up proportional to the extra space used.

Lemma E.3.3. *Let π be a syntactic sequential resolution refutation of a formula F in length L, maximal space s, and cumulative space c, and let $\ell \in \mathbb{N}^+$ be a positive integer. Then there is a semantic parallel resolution refutation of F in length $3(\lceil L/\ell \rceil + 1)$, maximal space $s + \lfloor \ell/2 \rfloor$, and cumulative space $3\lfloor c/\ell \rfloor + L$.*

Proof sketch. Analogously to the proof of Lemma E.2.3, we divide π into $\lceil L/\ell \rceil + 1$ intervals of at most ℓ steps each. We reorder derivation steps within every interval so that we do all axiom downloads first, inferences next, and removals at the end of the interval. We then collapse each sequence into one axiom download, one inference, and one removal step. □

Before discussing our main results, we observe that although proving strong lower bounds for the fully parallel versions of resolution looks like a formidable challenge, which we leave as future work, we can obtain a simple separation between semantic and syntactic fully parallel resolution.

Proposition E.3.4. *Every syntactic, fully parallel resolution refutation of a minimally unsatisfiable CNF formula with m clauses in space $s \leq m$ requires length at least $m/s + \log s - 2$.*

Proof. Let F be a minimally unsatisfiable CNF formula with m clauses and let $\pi = (\mathbb{C}_1, \ldots, \mathbb{C}_t)$ be a syntactic, fully parallel refutation of F in space $s \leq m$. We wish to show that $t \geq m/s + \log s - 2$. Let $t' \leq t$ be the smallest index such that $\bot \in \mathbb{C}_{t'}$. Note that $\pi = (\mathbb{C}_1, \ldots, \mathbb{C}_{t'})$ is also a syntactic, fully parallel refutation of F in space $s \leq m$.

We recursively define the notion of a *useful* clause (in time step i). The empty clause is always useful. A clause is useful in time step i if it is used in some time step $j \geq i$ to infer a useful clause.

Now let us work our way backwards in the refutation. The only useful clause in $\mathbb{C}_{t'}$ is $\bot$. Since the inference rule is binary, the number of useful clauses in the second-to-last configuration is at most 2. By the same reasoning, the number of useful clauses in the ith last configuration is at most 2^i. Hence, in the last $\log s$ steps there are at most $2s$ useful clauses.

Since F is minimally unsatisfiable, every axiom is needed to derive contradiction and thus must be useful in some configuration. Therefore, apart from the last $\log s$ configurations, we still need at least $(m - 2s)/s$ other configurations (by the fact that there are at most s clauses per configuration). We conclude that $t \geq t' \geq m/s + \log s - 2$. $\square$

In particular, any syntactic, fully parallel refutation of a minimally unsatisfiable CNF formula with m clauses requires length $\log m$, and a refutation in this length requires space $\Omega(m)$. This is in contrast to semantic fully parallel refutations, which have proofs in length 2, and no space lower bound other than the trivial m/L, where L is the length of the refutation.

By way of example, consider a pebbling formula on a directed path of size $m - 1$. A syntactic, fully parallel refutation in length $\log m$ requires space $\Omega(m)$, while there exists a semantic fully parallel refutation in length $\log m$ and space $2m/\log m + \mathrm{O}(1)$: just download the axioms corresponding to $2m/\log m$ consecutive vertices at a time and infer one new clause.

While this is technically a separation, it is also very brittle. There are syntactic, fully parallel refutations of this pebbling formula in length $2\log m$ and space $2m/\log m + \mathrm{O}(1)$. More generally, for any $k \in \mathbb{N}^+$, there are syntactic, fully parallel refutations in length $(1 + 1/k)\log m$ and space $2km/\log m + \mathrm{O}(1)$. Indeed, the refutation can be done as follows. First, download the axioms corresponding to evenly spaced vertices at distance $(\log m)/2k$. Then repeat the following two operations $\log m/2k$ times: download the clauses corresponding to the next vertex in the path and apply a parallel inference step. This leaves us with a path of length $km/\log m$, which we can trivially refute in length at most $\log(km/\log m) < \log m$ and space $km/\log m$.

It remains an interesting open problem whether there are stronger, more robust separations between semantical and syntactic, fully parallel resolution models.

Moving on from this philosophical discourse to a more concrete discussion of results, we note that most of the proof complexity consequences we derive from the pebbling results in Section E.2 are for semantic inference-parallel resolution, and thus hold for all models above except the fully parallel ones. We start by reporting a somewhat disappointing—and perhaps surprising—fact: even in semantic inference-parallel resolution cumulative space is at least maximal space squared.

Lemma E.3.5. *If F requires maximal space s in syntactic sequential resolution, then any semantic inference-parallel refutation of F has cumulative space $\Omega(s^2)$.*

Proof. We first note that, by Theorem 3.7 in [ABRW02], if syntactic sequential resolution requires space s, then semantic sequential resolution requires space at most $s/2$. Moreover, by Observation E.3.2, if semantic sequential resolution requires space $s/2$, then semantic parallel, and consequently also semantic inference-parallel, requires space at least $s/4$.

For simplicity let us think of each step in a semantic inference-parallel resolution refutation as being either an inference-plus-erasure step or a download step. Clearly, this can only affect the clause space measure by a factor 2, and thus there is a configuration that contains at least $s/8$ clauses.

An inference-plus-erasure step can be seen as a compression operation. Since the proof system is semantic, we only care about the information contained in a configuration, and since an inference step cannot increase the information but only add explicitly clauses that are already implied by the configuration, there is no need to add any extra clauses on top of the minimum amount needed to encode the semantic information we want the proof to maintain at this point. Therefore, we can assume the number of clauses only increases at download steps, and since these are sequential we can conclude that the number of clauses increases by at most 1 at every step. This means that we can apply the same argument as for syntactic sequential resolution above: during the $s/16$ time steps preceding a space-$s/8$ configuration we must have at least $s/16$ clauses in memory, and hence a cumulative lower bound $\Omega(s^2)$ follows. □

It is important to note, though, that Lemma E.3.5 has no implications for cumulative space trade-offs for formulas where the maximal space complexity is at most $O(\sqrt{N})$ measured in the formula size N, since in this setting the max-space-squared argument only implies a trivial $\Omega(N)$ cumulative space lower bound, and we present such trade-off results that do not follow from Lemma E.3.5 below. We also report results that asymptotically beat the maximal-space-squared lower bound for cumulative space.

In order to obtain these results, we need to review how our cumulative pebbling results in Section E.2 can be translated to claims about resolution refutations of so-called *XORified pebbling formulas*. We just state the reduction that we need below, since we can use it in a completely black-box fashion without knowing any details about what these formulas are. The interested reader is referred to [BN11] for the missing details.[6]

For the upper bound, the pebbling-to-resolution reduction follows from Theorem 2 (substitution space theorem) in [BN11] and the fact that syntactic sequential resolution can simulate black pebbling (which was perhaps first observed in [BIW04], see also Lemma 4 in [BN11]). From the proof of the substitution space theorem (see the proof of Theorem 2.1 in [BN10] for details) it is quite immediate to see that the space bound translates to cumulative space, and therefore we have the following result.

[6]It might be worth noting, though, that just as in [BN11] our results hold not only for pebbling formulas substituted with exclusive or—substitution with any so-called *non-authoritarian* (or *robust*) function that can never be fixed by restricting any single variable to some value works fine. Binary exclusive or is just the simplest example of such a function, whereas standard or is a simple non-example since setting a single variable to true fixes the value of the function to true.

Theorem E.3.6 ([BN11]). *Let $\mathcal{P}$ be a sequential black pebbling of a DAG G in time L, space s, and cumulative space c. Then there is a syntactic sequential resolution refutation of the XORified pebbling formula $Peb_G[\oplus]$ in length $O(L)$, maximal space $O(s)$, and cumulative clause space $O(c)$.*

For the lower-bound, all that needs to be done is to verify that the resolution-to-pebbling reduction in [BN10] preserves cumulative space and works not only for semantic sequential resolution but also for semantic inference-parallel resolution. This is not hard to see given that the main technical step in this proof is Lemma 6.3 (a result from [Ben09]) which is oblivious to inference steps—only axiom downloads are taken into consideration, and semantic inference-parallel resolution is sequential on axiom downloads.

Theorem E.3.7 (by the proof of Theorem 6.2 in [BN10]). *Let π be a semantic inference-parallel resolution refutation of a XORified pebbling formula $Peb_G[\oplus]$ in length L, maximal space s, and cumulative clause space c. Then there is a sequential black-white pebbling of the underlying DAG G in time $3L$, space s, and cumulative space $3c$.*

Analogously to what is the case in [BN11], the generic reductions in Theorems E.3.6 and E.3.7 can now be applied to a multitude of different graph families with different pebbling properties to yield CNF formulas with the same properties in resolution. Below we just give a sample of such results that we find particularly interesting.

For maximal space it is known that formulas refutable in linear size $O(N)$ never require space more than $O(N/\log N)$. For cumulative space the lower bound can be truly quadratic, however, beating the max-space-squared bound in Lemma E.3.5 by a factor $\log^2 N$.

Theorem E.3.8. *There is a family of 6-CNF formulas $\{F_N\}_{N\in\mathbb{N}^+}$ of size $\Theta(N)$ that have syntactic sequential resolution refutations in size $O(N)$, and hence also in maximal clause space $O(N/\log N)$, but for which any semantic inference-parallel refutation requires cumulative clause space $\Omega(N^2)$.*

This theorem follows from studying pebbling formulas defined in terms of *grate graphs* as in [Sch83] and using that the high predecessor-robustness of these graphs imply strong lower bounds on cumulative space as stated in Theorem E.2.9.

A natural question is what cumulative space tells us about maximal space, and in particular whether high cumulative space complexity implies that the maximal space complexity must also be large. This might sound intuitively plausible, but turns out to be false in a very strong sense.

Theorem E.3.9. *There is a family of 6-CNF formulas $\{F_N\}_{N\in\mathbb{N}^+}$ of size $\Theta(N)$ that can be refuted in syntactic sequential resolution in size $O(N)$ and also in maximal clause space $O(\log N)$, but for which any semantic inference-parallel refutation requires cumulative clause space $\Omega(N^2/\log N)$.*

Here the graphs we need are surprisingly simple, namely butterfly graphs. They again have high cumulative space, as stated in Theorem E.2.8, but since they are shallow the pebbling formulas generated from them have refutations in small maximal space.

Finally, we turn to the question of length-space trade-offs. We remark that in a cumulative space setting formulas for which small-space proofs require superpolynomial length, as in the strongest results in [BN11, BNT13, BBI16], are not too interesting, since length is trivially a lower bound on cumulative space. Rather, we focus on formulas for which small-space proofs incur only a polynomial blow-up in proof length. Can we find such formulas for which it holds not only that short proofs must have large maximal space s, but where such short proofs must be memory-intensive in that this amount of space s must be used essentially throughout the whole proof? The answer to this question is yes, and one example are pebbling formulas over the bit-reversal permutation graphs studied in [LT82]. The upper bound in the next theorem is from [Nor12] and the lower bound follows by combining the reduction in Theorem E.3.7 with the cumulative space lower bound for bit-reversal permutation graphs stated in Theorem E.2.10.

Theorem E.3.10. *There is a family of* 6*-CNF formulas* $\{F_N\}_{N\in\mathbb{N}^+}$ *of size* $\Theta(N)$ *such that for any* $s = \mathrm{O}\big(\sqrt{N}\big)$ *the formula* F_N *has a syntactic sequential resolution refutation in size* $\mathrm{O}\big(N^2/s^2\big)$ *and maximal clause space* $\mathrm{O}(s)$*, but any semantic inference-parallel refutation of* F_N *in maximal clause space* s *requires cumulative clause space* $\Omega(N^2/s)$*.*

In particular, a proof in maximal space s has length $\Omega(N^2/s^2)$, and if furthermore the proof has length $\mathrm{O}(N^2/s^2)$, then $\Omega(N^2/s^2)$ of the configurations have space $\Omega(s)$. Hence, these formulas have syntactic sequential resolution refutations in simultaneous length $\mathrm{O}(N)$ and space $\mathrm{O}\big(\sqrt{N}\big)$, but any semantic inference-parallel refutation with the same parameters has $\Omega(N)$ configurations with space $\Omega(\sqrt{N})$. We remark that this result makes sense even in the weaker syntactic sequential model, since maximal space $\Omega(\sqrt{N})$ only implies a trivial $\Omega(N)$ cumulative space lower bound.

As already noted, semantic fully parallel resolution is an extremely powerful model, since we can refute any formula with just one (parallel) axiom download step followed by one (semantic) inference step, but if we limit the available space then the usefulness of parallelism is restricted. Using Observation E.3.2 we can transfer the trade-offs above from inference-parallel to fully parallel semantic resolution by sacrificing a factor s. For example, the result below follows directly from this observation and Theorem E.3.10.

Corollary E.3.11. *There is a family of* 6*-CNF formulas* $\{F_N\}_{N\in\mathbb{N}^+}$ *of size* $\Theta(N)$ *such that for any* $s = \mathrm{O}\big(\sqrt{N}\big)$ *the formula* F_N *has a syntactic sequential resolution refutation in size* $\mathrm{O}\big(N^2/s^2\big)$ *and maximal clause space* $\mathrm{O}(s)$*, but any semantic parallel refutation of* F_N *in maximal clause space* s *requires cumulative clause space* $\Omega(N^2/s^2)$*.*

E.4 Pebbling Cumulative Space Lower Bounds and Trade-offs

In this section we return to pebbling and we show that two combinatorial properties of graphs, namely $\mathcal{G}$-robustness and subgraph-dispersion, yield lower bounds on pebbling cumulative space. We also prove that for any constant s there are graphs that can be pebbled in space s but where any such pebbling requires time $\Omega\left(n^{s-1}\right)$.

We first introduce some notation which we use throughout this section. Recall that the time of a pebbling $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_t)$, where $\mathbb{P}_i = (B_i, W_i)$, is $t(\mathcal{P}) = t$; the (maximal) space is $s(\mathcal{P}) = \max_{i \in [t]} |B_i| + |W_i|$; and the cumulative space is $cc(\mathcal{P}) = \sum_{i \in [t]} |B_i| + |W_i|$.

As mentioned in the introduction, we consider five different models of pebbling games: sequential black $(\cdot)$, parallel black $(||)$, sequential black-white (bw), parallel-black sequential-white $(pbsw)$, and parallel black-white pebbling $(||bw)$, Given a DAG G with one sink and a pebbling model $\mathcal{M} \in \{\cdot, ||, bw, pbsw, ||bw\}$, let $\mathcal{S}$ be the set of legal $\mathcal{M}$ pebbling of G. The minimum space required to $\mathcal{M}$-pebble G is defined as

$$s_{\mathcal{M}}(G) = \min_{\mathcal{P} \in \mathcal{S}} \Pi_s(\mathcal{P}) \ . \tag{E.2}$$

Similarly, the minimum time required to $\mathcal{M}$-pebble G is

$$t_{\mathcal{M}}(G) = \min_{\mathcal{P} \in \mathcal{S}} \Pi_t(\mathcal{P}) \ , \tag{E.3}$$

and the minimum cumulative space required to $\mathcal{M}$-pebble G is

$$cc_{\mathcal{M}}(G) = \min_{\mathcal{P} \in \mathcal{S}} \Pi_{cc}(\mathcal{P}) \ . \tag{E.4}$$

We omit the subscript $\mathcal{M}$ when it is clear from the context.

E.4.1 Robustness Implies Cumulative Space Lower Bounds

The first property we consider is $\mathcal{G}$-robustness. Recall that a DAG G is said to be (e,d)-$\mathcal{G}$-robust if for every $U \subseteq V(G)$ of size at most e it holds that there exists a graph $H \in \mathcal{G}$ of size at least d that is a subgraph of $G - U$.

For the special case of depth-robustness, i.e., $\mathcal{G}$-robustness when $\mathcal{G}$ is the class of directed paths, it was proven in [ABP16] that if G is (e,d)-depth-robust then the cumulative space complexity of parallel-black pebbling G is at least ed. We generalise this result for other classes of DAGs $\mathcal{G}$ and other pebbling models.

Before enunciating the main theorem of this section, we state two facts which will be used in the proof. Both these facts hold for any pebbling model we consider in this paper.

Fact E.4.1. *Any complete legal pebbling of a DAG G with only one sink must pebble all vertices in G at least once.*

Fact E.4.2. *Let G be a single-sink DAG, and H be a single-sink subgraph of G. If $\mathcal{P} = (\mathbb{P}_i, \mathbb{P}_{i+1}, \ldots, \mathbb{P}_m)$ is a legal pebbling of G, then $\mathcal{P}' = (\mathbb{P}_i \cap H, \mathbb{P}_{i+1} \cap H, \ldots, \mathbb{P}_m \cap H)$ is a legal pebbling of H. Moreover, if $\mathcal{P}$ is a complete pebbling of G, then $\mathcal{P}'$ is a complete pebbling of H.*

The following notation is used throughout this section. Let $\mathcal{G}$ be a class of graphs each with a single sink, let $\mathcal{M}$ be a pebbling model and let d be a positive integer. We define

$$\tau_d(\mathcal{G}, \mathcal{M}) = \min\{\Pi_t^{\mathcal{M}}(H) : H \in \mathcal{G},\ |V(H)| \geq d\} - 1 \ , \tag{E.5}$$

that is, $\tau_d(\mathcal{G}, \mathcal{M})$ is largest number such that any graph $H \in \mathcal{G}$ of size at least d requires at least time $\tau_d(\mathcal{G}, \mathcal{M}) + 1$ to be $\mathcal{M}$-pebbled.

Theorem E.4.3. *Let $\mathcal{G}$ be a class of graphs each with a single sink and let e and d be positive integers. If G is an (e, d)-$\mathcal{G}$-robust DAG then for any $\mathcal{M} \in \{\cdot, \|, bw, pbsw, \|bw\}$ it holds that $\Pi_{cc}^{\mathcal{M}}(G) > e \cdot \tau_d(\mathcal{G}, \mathcal{M})$.*

Proof. Fix a pebbling model $\mathcal{M} \in \{\cdot, \|, bw, pbsw, \|bw\}$ and let $\tau_d = \tau_d(\mathcal{G}, \mathcal{M})$. All statements and all complexity measures in this proof refer to $\mathcal{M}$.

We prove the contrapositive of the theorem statement, that is, we show that if $\Pi_{cc}(G) \leq e\tau_d$ then there exists a set of vertices $U \subseteq V(G)$, $|U| \leq e$, such that no $H \in \mathcal{G}$ is a subgraph of $G - U$ and therefore G is not (e, d)-$\mathcal{G}$-robust.

Assume $\Pi_{cc}(G) \leq e\tau_d$ and let $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_t)$, where $\mathbb{P}_i = (B_i, W_i)$, be a pebbling of G with minimal cumulative space complexity, that is $\sum_{i=1}^{t} |B_i| + |W_i| = \Pi_{cc}(G)$. For $i \in [0, \tau_d - 1]$ let

$$\mathcal{P}_i = \{\mathbb{P}_i, \mathbb{P}_{i+\tau_d}, \mathbb{P}_{i+2\tau_d}, \ldots\} \ . \tag{E.6}$$

Note that $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{\tau_d - 1}\}$ form a τ_d-partition of the pebbling configurations in $\mathcal{P}$. Moreover, let

$$U_i = \{v \in B_j \cup W_j \mid \mathbb{P}_j \in \mathcal{P}_i\} \tag{E.7}$$

for $i \in [0, \tau_d - 1]$. By construction $\sum_{i=0}^{\tau_d - 1} |U_i| \leq \Pi_{cc}(G) \leq e\tau_d$. By an averaging argument at least one of the U_i's, say U_{i^*}, has size at most e.

It remains to show that $G - U_{i^*}$ contains no subgraph $H \in \mathcal{G}$ of size (at least) d. Suppose, for the sake of contradiction, that some $H \in \mathcal{G}$ of size at least d is a subgraph of $G - U_{i^*}$. By Fact E.4.2, $\mathcal{P}' = (\mathbb{P}_0 \cap H, \mathbb{P}_1 \cap H, \ldots, \mathbb{P}_t \cap H)$ is a complete pebbling of H. However, $V(H) \cap U_{i^*} = \emptyset$, that is, there are no pebbles on H in any configurations in $\mathcal{P}_{i^*} = \{\mathbb{P}_{i^*}, \mathbb{P}_{i^*+\tau_d}, \mathbb{P}_{i^*+2\tau_d}, \ldots\}$. This implies that H must be completely pebbled between $\mathbb{P}_{i^*+j\tau_d}$ and $\mathbb{P}_{i^*+(j+1)\tau_d}$ for some $j \in \mathbb{N}$. But this is a contradiction since, by definition of τ_d, H requires at least time $\tau_d + 1$ to be completely pebbled. $\square$

Let us first consider what Theorem E.4.3 implies for different pebbling models in the case of depth-robustness, i.e., when $\mathcal{G}$ is the class of directed paths. For parallel black pebbling, the notion of depth-robustness was already studied in [ABP16] where they prove the following result.

Corollary E.4.4 ([ABP16]). *If G is an (e,d)-depth-robust graph, then the cumulative space complexity of parallel-black pebbling G is at least ed.*

We show that Theorem E.4.3 implies the same bound for black-white pebbling and a somewhat weaker bound for parallel-black sequential-white pebbling.

Corollary E.2.6 (Restated). *If G is an (e,d)-depth-robust DAG, then G requires sequential black-white cumulative space at least ed, and parallel-black sequential-white cumulative space at least $e\sqrt{d}$.*

Proof. For sequential black-white lower bound, by Theorem E.4.3 it is enough to show that the path on d vertices requires $d+1$ time to sequential black-white pebble. This follows trivially from Fact E.4.1 and the fact that in the sequential model at most one pebble is placed on G per time step.

For the parallel-black sequential-white lower bound, again by Theorem E.4.3 it is enough to show that the path on d vertices requires $\sqrt{d}+1$ time to pebble in this model. Let L be a path on d vertices and let ℓ be the number of white pebbles placed on L during the pebbling. There must be (at least) one subpath L' of L of size $(d-\ell)/(\ell+1)$ that is never pebbled with a white pebble. By Fact E.4.1, each node in L' must have been pebbled at some point, so L' must be completely pebbled with black pebbles. Since a node can not be black pebbled before its predecessors are pebbled, placing black pebbles requires time at least $(d-\ell)/(\ell+1)$. Observing that we are sequential on placement of white pebbles yields a total time for placements of at least $\ell+(d-\ell)/(\ell+1)$ which is greater than $\sqrt{d}$ for $\ell \geq 0$, and hence a total time including removals of at least $\sqrt{d}+1$. □

Let us now consider the relation between predecessor-robustness and cumulative space complexity. We first note that predecessor-robustness does not imply hardness for parallel black pebbling since there are very shallow graphs which are predecessor-robust (as, for example, the butterfly graphs discussed in Section E.2). However, for sequential black-white pebbling it follow from Theorem E.4.3 that (e,d)-predecessor-robustness implies cumulative space at least ed. In particular, this implies that the linear sized grates of [Sch83] have $\Theta(n^2)$ black-white cumulative space complexity and that butterfly graphs, which can be black pebbled with $\log n$ pebbles, require $\Theta(n^2/\log n)$ black-white cumulative space.

Corollary E.2.7 (Restated). *If G is an (e,d)-predecessor-robust DAG, then G requires sequential black-white cumulative space at least ed.*

Proof. As was the case for the sequential black-white lower bound in Corollary E.2.6, this corollary also follows trivially from Theorem E.4.3 and Fact E.4.1. □

E.4.2 Dispersion and Cumulative Space Trade-offs

In [ABP16] it is proven that (path-)dispersion implies cumulative space lower bounds for parallel black pebbling. In this section we introduce the more general property subgraph-dispersion and show that it implies both cumulative space lower bounds and cumulative space trade-offs for black-white pebbling. We also prove that random permutations are path-dispersed and therefore exhibit such trade-offs.

E.4.2.1 Dispersion Implies Cumulative Space Trade-offs

Intuitively, A DAG is (k, z, g)-path-dispersed if it contains a path that can be partitioned into k segments such that each segment has at least z disjoint incoming paths of length at least g. The following definitions make this concept precise.

Definition E.4.5. A DAG is (k, z, g)-path-dispersed if it contains a directed path $\mathcal{L}$ that can be partitioned into k subpaths $L^1, L^2, \ldots, L^k$ such that for every L^i there are z directed paths $\phi^i_1, \phi^i_2, \ldots, \phi^i_z$, each with at least g vertices, that satisfy the following:

- for every $i \in [k]$ and every $j, j' \in [z]$ such that $j \neq j'$ it holds that $\phi^i_j \cap \phi^i_{j'} = \emptyset$;
- for every $i, i' \in [k]$ and every $j \in [z]$, it holds that $L^i \cap \phi^{i'}_j = \emptyset$; and
- for every $i \in [k]$ and every $j \in [z]$ there is a vertex $v^i_j \in L^i$ such that there is an edge from the sink of ϕ^i_j to v^i_j and moreover, if $j' \in [z]$ and $j \neq j'$, then $v^i_j \neq v^i_{j'}$.

Note that the paths ϕ^i_j coming into one subpath L^i have to be disjoint among themselves, but not necessarily among paths coming into other subpaths.

Definition E.4.6. A DAG is (k, z, g)-subgraph-dispersed if it contains k disjoint subgraphs $L^1, L^2, \ldots, L^k$ such that for every L^i there are z subgraphs $\phi^i_1, \phi^i_2, \ldots, \phi^i_z$, each with at least g vertices, that satisfy the following:

- for every $i \in [k]$ and every $j, j' \in [z]$ such that $j \neq j'$ it holds that $\phi^i_j \cap \phi^i_{j'} = \emptyset$;
- for every $i, i' \in [k]$ and every $j \in [z]$ it holds that $L^i \cap \phi^{i'}_j = \emptyset$; and
- for every $i \in [k]$ and every $j \in [z]$ there is a vertex $v^i_j \in L^i$ such that v^i_j is the only sink of the subgraph induced by $\{v^i_j\} \cup V(\phi^i_j)$ and moreover, if $j' \in [z]$ and $j \neq j'$, then $v^i_j \neq v^i_{j'}$.

Note that a (k, z, g)-path-dispersed graph is also (k, z, g)-subgraph-dispersed. Although we prove that subgraph-dispersion implies high cumulative space complexity, we only apply this result to obtain cumulative space trade-offs on graphs that are path-dispersed.

Lemma E.4.7. *If G is a (k,z,g)-subgraph-dispersed graph, then the black-white cumulative space complexity of pebbling G is*

$$\Pi_{cc}^{bw}(G) \geq k \min\{gz/2, z^2/4\} \ . \tag{E.8}$$

Furthermore, if $\mathcal{P}$ is a sequential black-white pebbling of G in space $\Pi_s(\mathcal{P}) = s \leq z/2$, then

$$\Pi_{cc}(\mathcal{P}) \geq k \min\{g(z-s), g(z-2s)+s^2\} \ . \tag{E.9}$$

Proof. Consider k disjoint subgraphs, $L^1, L^2, \dots, L^k$, and the corresponding subgraphs ϕ_j^i for $i \in [k]$ and $j \in [z]$ that witness the fact that G is (k,z,g)-subgraph-dispersed graph. We refer to the subgraphs ϕ_j^i as *incoming-subgraphs*.

We keep an account for each L^i. For each pebbling move, we charge some of the pebbles in the configuration to each account, ensuring that we do not charge the same pebble in the same configuration to more than one account. It is enough to prove that the number of pebbles charged to each L^i is at least $\min\{gz/2, z^2/4\}$ and, if $\Pi_s(\mathcal{P}) = s \leq z/2$, at least $\min\{gz, g(z-2s)+s^2\}$.

We charge pebbles according to the following rules:

1. When a black pebble is placed or a white pebble is removed from L^i, charge all pebbles on incoming-subgraphs to the account of L^i.

2. When a black is placed or a white is removed from some incoming-subgraph, charge, for all $i \in [k]$, all pebbles on L^i to the account of L^i.

Observe that at every configuration a pebble is never charged to two different supgraphs L^i. This is so because if a black pebble is placed or a white pebble is removed from L^i then it is not being placed or removed from any ϕ nor from any other $L^{i'}$, $i \neq i'$, because these subgraphs are disjoint and hence any pebble charged at this configuration is only being charged to L^i. Moreover, if a black is placed or a white is removed from some ϕ_j^i, then the pebbles charged to each L^i are only the pebbles on L^i, and since these subgraphs are disjoint, no pebble is being charged twice.

We now focus on a given subgraph L^i, so we drop the superscript and denote it L. We consider a time interval during which L is pebbled, i.e., there is no pebble on L before the the interval, every vertex in L is pebbled at some point, there is no pebble on L after the interval, and there is some pebble on L at all times during the interval. This interval exists because L has a unique sink.

We order incoming-subgraphs according to the time they are first "used" for pebbling L, i.e., a black pebble is placed or a white pebble is removed from the vertex in L which is the sink of the incoming-subgraph. Let $\phi_1, \phi_2, \dots, \phi_z$ be the incoming-subgraphs according to this ordering. Note that when ϕ_j is used there is at least one pebble in ϕ_j.

For each incoming-subgraph ϕ_j we distinguish three cases depending on when, during the pebbling of L, the incoming-subgraph ϕ_j is completely unpebbled.

Case 1 (01): There is a pebble on ϕ_j from the moment it is used until the end of the interval. We charge at least $z - j$ pebbles according to rule 1, since there is a pebble on ϕ_j when the last $z - j$ incoming-subgraphs are used.

Case 2 (10): There is a pebble on ϕ_j from the beginning of the interval until the moment it is used. We charge at least j pebbles according to rule 1, since there is a pebble on ϕ_j when the first j incoming-subgraphs are used.

Case 3 (010): There is no pebble on ϕ_j at the beginning nor at the end of the interval. Note that in this case ϕ_j must be completely pebbled within the interval. We charge at least $|\phi_j| \geq g$ pebbles according to rule 2, since there is always a pebble on L while ϕ_j is pebbled.

Consider a partition $\Phi_{01} \dot\cup \Phi_{10} \dot\cup \Phi_{010}$ of the incoming-subgraphs according to what case they fall into. If there is an incoming-subgraph that falls into both Case 1 and 2, then we can choose arbitrarily whether to include it in Φ_{01} or Φ_{10}. Let $c(L)$ be the number of pebbles charged to the account of L. We have that $c(L) \geq \sum_{j \in \Phi_{01}} (z - j) + \sum_{j \in \Phi_{10}} j + g|\Phi_{010}|$. The minimum is attained when $\{z - j : \phi_j \in \Phi_{01}\} = \{j : \phi_j \in \Phi_{10}\} = [|\Phi_{01}|]$, and therefore $c(L) \geq |\Phi_{01}|^2 + g|\Phi_{010}|$.

Without any restriction on the maximal space, there are two possible minima: if $g \geq z/2$, then the minimum is attained when $|\Phi_{01}| = |\Phi_{10}| = z/2$, in which case we get $\Pi_{cc}(L) \geq z^2/4$; and if $g < z/2$ then the minimum is attained when $|\Phi_{01}| = |\Phi_{10}| = g$ and $|\Phi_{010}| = z - 2g$, in which case we get $\Pi_{cc}(L) \geq g^2 + g(z - 2g) = g(z - g) \geq gz/2$.

If we enforce that the maximal space is at most $s \leq z/2$, then $|\Phi_{01}| \leq s$ and $|\Phi_{10}| \leq s$ and we again get two minima, analogously to the two above. If $g \geq s$ the minimum is attained when $|\Phi_{01}| = |\Phi_{10}| = s$ and $|\Phi_{010}| = z - 2s$, giving $\Pi_{cc}(L) \geq g(z - 2s) + s^2$; and if $g < s$ then the minimum is again attained when $|\Phi_{01}| = |\Phi_{10}| = g$ and $|\Phi_{010}| = z - 2g$ and yields $\Pi_{cc}(L) \geq g(z - g) \geq g(z - s)$. □

E.4.2.2 Permutation Graphs are Disperse

We now use Lemma E.4.7 to show that two subclasses of permutation graphs, namely bit-reversal and random permutation graphs, exhibit cumulative space trade-offs.

Definition E.4.8. Given an interval $[a, b]$ and a permutation $\sigma \in \mathfrak{S}([a, b])$, the *permutation graph* $G(\sigma)$ is the graph G with vertex set $\{x_a, x_{a+1}, \ldots, x_b, y_a, y_{a+1}, \ldots, y_b\}$ and edges $\{(x_i, x_{i+1}), (y_i, y_{i+1}) \mid a \leq i < b\} \cup \{(x_i, y_{\sigma(i)}) \mid a \leq i \leq b\}$.

Recall that for $n = 2^m$ the bit-reversal permutation reverses the binary representation of a number $j \in [0, n - 1]$, i.e., if $j = (b_1 \ldots b_m)_{(2)}$ then $\sigma(j) = (b_m \ldots b_1)_{(2)}$. Lemma E.4.7 implies the following theorem, which is a more precise version of Theorem E.2.10.

Theorem E.4.9. *If G is a bit-reversal permutation graph on $2n$ vertices, then the sequential black-white cumulative space complexity of pebbling G is*

$$\Pi_{cc}^{bw}(G) \geq \frac{n^{3/2}}{4} \ , \tag{E.10}$$

and any sequential black-white pebbling $\mathcal{P}$ of G in space $s = \Pi_s(\mathcal{P})$ is such that

$$\Pi_{cc}(\mathcal{P}) \geq \frac{n^2}{9s} + \frac{n}{6} \ . \tag{E.11}$$

Proof. For every $z \leq n = 2^m$ that is a power of 2 it is easy to see that G is $(n/z, z, n/z)$-path-dispersed by partitioning the path $(y_1, y_2, \ldots, y_n)$ in G in n/z paths of length z. From Lemma E.4.7 we get that $\Pi_{cc}^{bw}(G) \geq n^{3/2}/4$ by taking $z = 2^{\lceil m/2 \rceil}$. Moreover, Lemma E.4.7 also implies that if $\mathcal{P}$ is a sequential black-white pebbling of G in space $s = \Pi_s(\mathcal{P}) \leq z/2$, then

$$\Pi_{cc}(\mathcal{P}) \geq \frac{n^2(z-2s)}{z^2} + \frac{ns}{z} \min\left\{\frac{n}{z}, s\right\} \ . \tag{E.12}$$

If $s \geq n/3$, then $\Pi_{cc}(\mathcal{P}) \geq \frac{n^2}{9s} + \frac{n}{6}$ holds trivially. So let us assume that $s \leq n/3$, and hence by setting $z = 2^{\lceil \log 3s \rceil}$, we have that $z \leq n$. Note that $3s \leq z \leq 6s$, and that $\min_{3s \leq z \leq 6s}(z-2s)/z^2 = 1/9s$. The result follows by observing that $\min\{n/z, s\} \geq 1$. □

We now focus on the study of random permutation graphs. Consider the following distribution of permutation graphs.

Let $j_1, j_2 \ldots, j_n, k_1, k_2 \ldots, k_n$ be integers in $[1, n]$ chosen independently uniformly at random. Let $G' = (X \cup Y, A)$ be a digraph such that

- $X = \{x_1, x_2, \ldots, x_n\}$,
- $Y = \{y_1, y_2, \ldots, y_n\}$, and
- $A = \{(x_i, x_{i+1}), (y_i, y_{i+1}) \mid 1 \leq i < n\} \cup \{(x_{j_i}, y_{k_i}) \mid 1 \leq i \leq n\}$.

We perform two operations on G' to obtain the graph G: edge contraction and degree reduction. Edge contraction consists of contracting one by one any edge (x_i, x_{i+1}) or (y_i, y_{i+1}) that has an endpoint which is not adjacent to any vertex in the opposite partition. Degree reduction consists of substituting vertices of high degree by paths and distributing the edges adjacent to that vertex to the vertices on the path. Formally, for all vertices $x \in X$ that are adjacent to $d \geq 2$ vertices in Y, say $\{y_{i_1}, \ldots, y_{i_d}\}$, substitute x by a path on d vertices, say $\{x_{i_1}, \ldots, x_{i_d}\}$, and for $\ell \in [d]$ include the edge $(x_{i_\ell}, y_{i_{\sigma(\ell)}})$, where σ is a random permutation in $\mathfrak{S}_d$. The analogous operation is done for vertices $y \in Y$ of high degree.

We claim this distribution is equivalent to choosing a permutation uniformly at random, but we omit the proof. We are now ready to prove the following theorem.

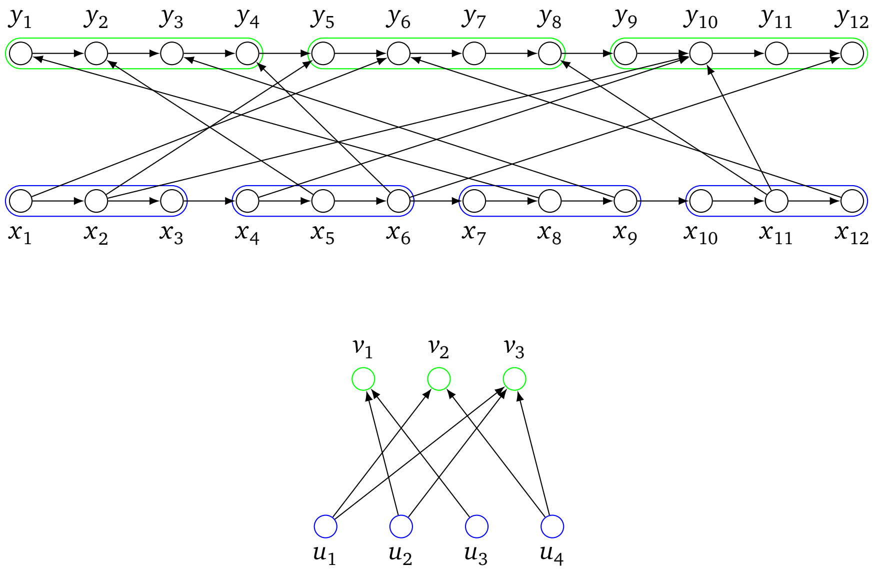

y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12}
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
v_1 v_2 v_3
u_1 u_2 u_3 u_4

Figure E.2: An example of a random graph G' and its corresponding bipartite graph H, for $k = 4$ and $z = 3$

Proposition E.4.10. *A random permutation graph is asymptotically almost surely an $(\alpha k, (\beta - 1/2)z, \alpha k)$-path-dispersed graph, for $\alpha, \beta < 1$, $\alpha + \beta - \alpha\beta \ll (1 - e^{-n/kz})$ and $kz = \mathrm{O}(n)$.*

Proof. Let G' be a digraph sampled as described above, and let G be any graph obtained from G' by edge contraction and degree reduction. We define a bipartite graph H from G' in the following manner. Let $H = ((U, V), F)$ with $U = \{u_1, u_2, \ldots, u_k\}$ and $V = \{v_1, v_2, \ldots, v_z\}$. Each node $u_i \in U$ is a super-node that represents the set $\{x_{(i-1)n/k+1}, x_{(i-1)n/k+2}, \ldots, x_{in/k}\}$ and similarly each node $v_i \in V$ is a super-node that represents the set $\{y_{(i-1)n/z+1}, y_{(i-1)n/z+2}, \ldots, y_{in/z}\}$, where for simplicity we assume both k and z divide n. For a node $w \in U \cup V$ let $R(w) \subset X \cup Y$ be the set of vertices that w represents. There is an edge from $u \in U$ to $v \in V$ if there is at least one edge from some vertex in $R(u)$ to some vertex in $R(v)$ (see Figure E.2).

Proposition E.4.10 follows from the following claim which we prove later on.

Claim E.4.11. *Let Z be an integer random variable in $[1, kz]$ that represents the number of edges in H. Let $p = (1-1/kz)^n$ and $q = (1-1/(kz-1))^n$. Then for any $0 < \delta \leq (1-p)$*

$$\Pr[Z \leq \delta kz] \leq \frac{p(q-p)}{(1-p-\delta)^2} \quad .$$

Note that $\lim_{n\to\infty} q - p = 0$ and that $\lim_{n\to\infty} p \ll 1$ if $kz = \mathrm{O}(n)$. This means that for any $\delta \ll (1-p)$ the probability that Z is less than δkz goes to 0 as n increases.

In other words, with high probability the number of edges in the bipartite graph is extremely close to its expected value.

All that remains to show is that if $Z > \delta kz$, then G is $(\alpha k, (\beta - 1/2)z, \alpha k)$-path-dispersed. The proposition then follows by choosing $\delta = \alpha + \beta - \alpha\beta \ll (1 - e^{-n/kz}) \leq (1 - p)$ so that, by Claim E.4.11, the probability that G is not path-dispersed goes to 0 as n increases.

Suppose $Z > \delta kz$. Then for any α, β that satisfy $\alpha + \beta - \alpha\beta \leq \delta$ there are αk nodes in U with outdegree at least βz, and βz nodes in V with indegree at least αk. This is indeed the case, because if there were not αk nodes in U with outdegree at least βz, then there would be strictly less than $(1-\alpha)\beta kz + \alpha kz = (\alpha + \beta - \alpha\beta)kz \leq \delta kz$ edges in H, and the same holds if there were not βz nodes in V with indegree at least αk.

We say a node $u \in U$ ($v \in V$, respectively) is *significant* if it has outdegree (indegree, respectively) at least βz (αk, respectively). Let $u \in U$ be a significant node and let $v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}$ be the significant nodes adjacent to u, where $i_j < i_k$ if $j < k$. Observe that, since u has outdegree at least βz and at least βz of the z nodes in V are significant, it must be the case that $\ell \geq (2\beta - 1)z$.

Now note that between any vertex (in G) represented by v_{i_j} and any vertex represented by $v_{i_{j+2}}$ there is a path of length at least αk (since $v_{i_{j+1}}$ has indegree at least αk). By considering only v_{i_j} for j odd, we can conclude that G is $(\alpha k, (\beta - 1/2)z, \alpha k)$-path-dispersed. □

Proof of Claim E.4.11. Let us consider the probability of a given edge not being present in H. Let $u, u' \in U$ and $v, v' \in V$ be nodes chosen independently at random conditioned on $u \neq u'$ or $v \neq v'$. We define $p = \Pr[uv \notin E] = (1 - 1/kz)^n$ and $q = \Pr[u'v' \notin E \mid uv \notin E] = (1 - 1/(kz-1))^n$, so that $\Pr[uv \notin E \text{ and } u'v' \notin E] = pq$.

From Chebyshev's inequality, we have that

$$\Pr[Z \leq \delta kz] \leq \Pr[|\mathrm{E}[Z] - Z| \geq |\mathrm{E}[Z] - \delta kz|] \tag{E.13}$$

$$\leq \frac{\mathrm{Var}(Z)}{(\mathrm{E}[Z] - \delta kz)^2} \tag{E.14}$$

$$= \frac{\mathrm{E}[Z^2] - \mathrm{E}[Z]^2}{(\mathrm{E}[Z] - \delta kz)^2} \quad . \tag{E.15}$$

By linearity of expectation we have that $\mathrm{E}[Z] = kz(1-p)$. To compute $\mathrm{E}[Z^2]$, let X_v be an integer random variable in $[1, z]$ that represents the degree of $v \in V$. Note that $Z = \sum_{v \in V} X_v$. Let Y_{uv} be a binary random variable that indicates whether $uv \in E$. We observe that

$$\mathrm{E}[X_v^2] = \sum_{u,u' \in U} \mathrm{E}[Y_{uv} Y_{u'v}] = z(1-p) + z(z-1)(1 - 2p + pq) \ , \tag{E.16}$$

and that for $v' \neq v$

$$\mathrm{E}[X_v X_{v'}] = \sum_{u,u' \in U} \mathrm{E}[Y_{uv} Y_{u'v'}] = z^2 \Pr[Y_{uv} = 1 \text{ and } Y_{u'v'} = 1] = z^2(1 - 2p + pq) \ . \tag{E.17}$$

Therefore, we obtain

$$\begin{aligned} \mathrm{E}[Z^2] &= \sum_{v,v' \in V} \mathrm{E}[X_v X_{v'}] && \text{(E.18)} \\ &= k\,\mathrm{E}[X_v^2] + k(k-1)\,\mathrm{E}[X_v X_{v' \neq v}] && \text{(E.19)} \\ &= (kz)^2(1-2p+pq) - kzp(q-p)\ . && \text{(E.20)} \end{aligned}$$

Thus, we can conclude that

$$\Pr[Z \le \delta kz] \le \frac{\mathrm{E}[Z^2] - \mathrm{E}[Z]^2}{(\mathrm{E}[Z] - \delta kz)^2} \le \frac{(kz)^2 p(q-p)}{(kz(1-p) - \delta kz)^2} \le \frac{p(q-p)}{(1-p-\delta)^2}\ . \qquad \square$$

We can now conclude that random permutation graphs exhibit cumulative space trade-offs. The theorem below is a more precise version of Theorem E.2.11.

Theorem E.4.12. *If G is a random permutation graph, then asymptotically almost surely the black-white cumulative space complexity of pebbling G is*

$$\Pi_{cc}^{bw}(G) \ge n^{3/2}/125\ , \qquad \text{(E.21)}$$

and any black-white pebbling $\mathcal{P}$ of G in space $s = \Pi_s(\mathcal{P})$ is such that

$$\Pi_{cc}(\mathcal{P}) \ge n^2/1250s\ . \qquad \text{(E.22)}$$

Proof. Let $k = n/4z$, $\alpha = 4/5$ and $\beta = 9/10$, so that $\alpha + \beta - \alpha\beta = 98/100$ and $(1 - e^{-n/kz}) = (1 - e^{-4}) > 0.9816$. Note that the conditions of Proposition E.4.10 are satisfied and we can therefore conclude that with high probability G is $(n/5z, 2z/5, n/5z)$-path-dispersed. By setting $z = \sqrt{n}$, Lemma E.4.7 implies that $\Pi_{cc}^{bw}(G) \ge n^{3/2}/5^3$; and by setting $z = 10s$, Lemma E.4.7 implies that any black-white pebbling $\mathcal{P}$ of G in space s is such that $\Pi_{cc}(\mathcal{P}) \ge n^2/1250s$. $\square$

E.4.3 Pebbling in Small Space Can Require Maximum Length

In this section we prove Theorems E.2.12 and E.2.13 which give tight upper and lower bounds for black and for black-white pebbling a k-layer bit-reversal graph.

The graphs we work with are stacks of permutation graphs, which as mentioned in Section E.2 consist of k directed path graphs of length n and $k-1$ layers of bit-reversal permutations between the vertices $1, 2, \dots, n$ in consecutive paths. Layer 1 is the layer that contains the source, and layer k contains the sink.

Definition E.4.13. Given an interval $[a,b]$ and a permutation $\sigma \in \mathfrak{S}([a,b])$, a k-layer permutation graph is the graph $B = (W,E)$ such that $W = \{x_{i,j} \mid (i,j) \in [k] \times [a,b]\}$ and $E = \{(x_{i,j}, x_{i,j+1}) \mid (i,j) \in [k] \times [a,b-1]\} \cup \{(x_{i,j}, x_{i+1,\sigma(j)}) \mid (i,j) \in [k-1] \times [a,b]\}$.

More concretely we use the bit-reversal permutation which for $n = 2^m$ reverses the binary representation of a number $j \in [0, n-1]$, i.e., if $j = (b_1 \dots b_m)_{(2)}$ then $\sigma(j) = (b_m \dots b_1)_{(2)}$. Observe that the bit-reversal permutation is an involution, that is $\sigma^2 = 1$. For simplicity we assume that m is even so that $\sqrt{n}$ is an integer.

As a warm-up before proving general upper bounds, we begin by proving a $k\sqrt{n}$ upper bound on space. The main ideas we need for the general case are already present in the proof of Lemma E.4.14.

Lemma E.4.14. *Let B be a k-layer bit-reversal graph. There is a sequential black pebbling $\mathcal{P}$ of G in space $\Pi_s(\mathcal{P}) = k\sqrt{n} + \mathrm{O}(1)$ and time $\Pi_t(\mathcal{P}) \leq k^2 n^{3/2}$.*

Proof. We divide each path in $\sqrt{n}$ blocks of length $\sqrt{n}$. Observe that the image of a block are $\sqrt{n}$ evenly separated vertices at distance $\sqrt{n}$ (see Figure E.3, where the predecessors of the black block are marked in grey).

We pebble the graph layer by layer, from layer 1 to layer k, keeping the invariant that when we begin to pebble a layer i there are $\sqrt{n}$ pebbles in each layer below, one at the beginning of each block. With "pebble layer i" we mean that we are pebbling it for the first time. If we are currently pebbling layer i, we call layers even or odd depending on the distance to layer i. We have two types of operations, each of which consists of several moves:

1. advance a (new) pebble through a block on an odd layer, and
2. advance each pebble on an even layer to the next position.

To pebble layer i we add a pebble at the beginning of the layer (which we can by the invariant) and then we proceed in $\sqrt{n}$ rounds. In a round we apply $\sqrt{n}$ times operation 2 so that every pebble in layer i advances $\sqrt{n}$ steps, and then we add a pebble at the beginning of layer i again. At each round the number of pebbles in the layer increases by 1, until after round $\sqrt{n}$ we have $\sqrt{n}$ pebbles on positions that are multiples of $\sqrt{n}$.

We maintain the following invariant between operations: there are $\sqrt{n}$ pebbles on each layer below, and while pebbles on odd layers are on positions that are multiples of $\sqrt{n}$, after the j-th operation on layer i, pebbles on even layers are on positions congruent to j modulo $\sqrt{n}$. Observe that after a round, this is $\sqrt{n}$ operations, all pebbles are on positions that are multiples of $\sqrt{n}$.

We prove inductively that given the operation invariant the cost of an operation on layer ℓ is $\ell\sqrt{n}$. If want to advance all $\sqrt{n}$ pebbles on an even layer ℓ to the next position, say a position that is congruent to j modulo $\sqrt{n}$, we just need to advance one (new) pebble along the preimage of the destination vertices, which is the $\sigma^{-1}(j)$-th block in layer $\ell - 1$, and at each of these advances we are able to advance one of the pebbles on layer ℓ. Since there is a pebble at the beginning of each block on layer $\ell - 1$, this can be done by one application of operation 1 on layer $\ell - 1$, which by the induction hypothesis has cost $(\ell - 1)\sqrt{n}$. The total cost is therefore $\sqrt{n} + (\ell - 1)\sqrt{n}$.

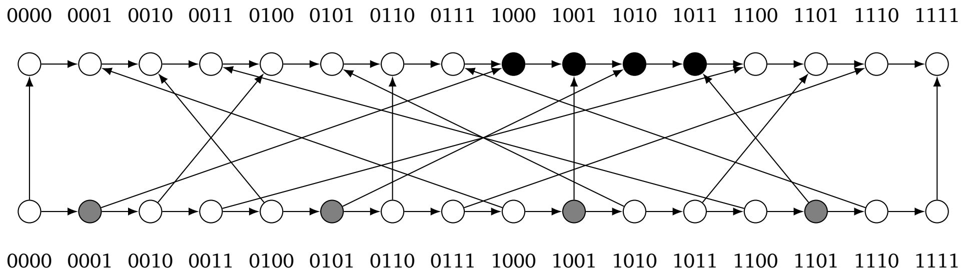


Figure E.3: The predecessors of a block are evenly distributed

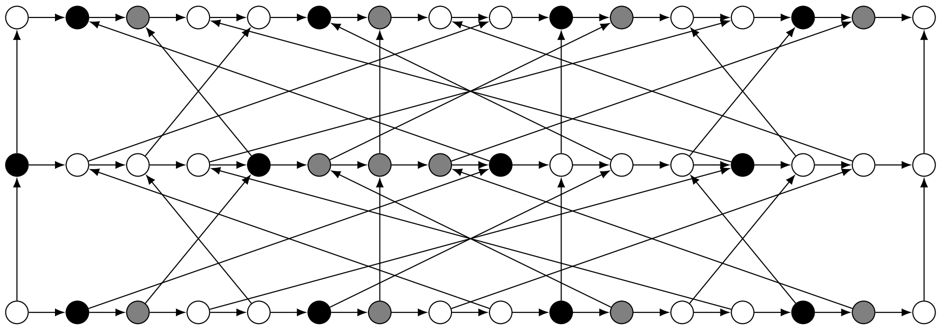


Figure E.4: An operation on layer i requires the same operation on layer $i-2$

If instead we want to advance a pebble through block $\sigma^{-1}(j)$ on an odd layer ℓ, then we need to have pebbles in the preimage of the block, which is the set of vertices in layer $\ell-1$ that are congruent to $\sigma^{-2}(j)=j$ modulo $\sqrt{n}$. By the operation invariant, the set of vertices in layer $\ell-1$ that are congruent to $j-1$ modulo $\sqrt{n}$ all have pebbles, so it is enough to advance each pebble on layer $\ell-1$ to the next position (see Figure E.4, where the configuration after the 2nd operation is marked in black and the nodes needed for the 3rd operation are marked in grey). This can be done with by one application of operation 2 on layer $\ell-1$, which by the induction hypothesis has cost $(\ell-1)\sqrt{n}$. Again the total cost is $\sqrt{n}+(\ell-1)\sqrt{n}$.

Therefore, the cost of pebbling layer i is $in^{3/2}$ since it requires n applications of operation 2. We can conclude that the total time of pebbling the graph, that is, the time needed to pebble each layer, is $\sum_{i=1}^{k} in^{3/2} \le k^2 n^{3/2}$. □

Lemma E.4.15. *Let B be a k-layer bit-reversal graph. There is a sequential black-white pebbling $\mathcal{P}$ of B in space $\Pi_s(\mathcal{P})=2k\sqrt{n}+O(1)$ and time $\Pi_t(\mathcal{P})\le kn$.*

Proof. We use the same approach as the black-only case, except that now we can initially setup each layer with white pebbles, and we convert these white pebbles into black whenever we have the opportunity.

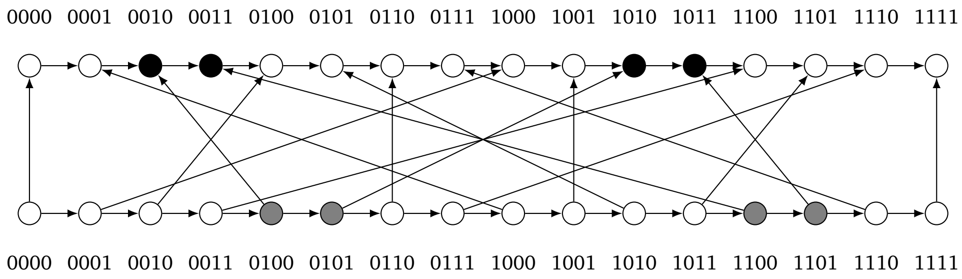


Figure E.5: The predecessors of evenly distributed blocks are evenly distributed blocks

More precisely, we initially place $k\sqrt{n}$ white pebbles, one at the beginning of each block. Then we apply $\sqrt{n}$ even layer operations on layer k. Observe that at the end of each operation on each odd layer we replace a white pebble with a black pebble, and at the very end we replace all white pebbles with black pebbles on each even layer.

The final cost is the cost to do $\sqrt{n}$ advance operations on the last layer, this is time kn. □

The general upper bounds use the same approach, except that some times the blocks we want to pebble through do not have a pebble at the beginning, so we have to do a setup phase. By choosing the order in which we pebble blocks, however, we are able to reuse the same setup for pebbling s consecutive blocks, thus saving on time.

Lemma E.4.16. *Let B be a k-layer bit-reversal graph. Then for any s such that $k+1 \le s \le \sqrt{n}$ there is a sequential black pebbling $\mathcal{P}$ of B in space $\Pi_s(\mathcal{P}) = 2k^2 s + O(1)$ and time $\Pi_t(\mathcal{P}) \le n^k/s^{2k-3} + O(n^{k-1}/s^{2k-5})$.*

Proof. We divide each path in n/s blocks of length s. Observe that the preimage of s blocks at distance n/s are s (different) blocks at distance n/s (see Figure E.5).

Our approach is again to pebble the graph layer by layer, but now we only keep s pebbles in each layer. More precisely, when we begin to pebble a layer i there are s pebbles on positions that are multiples of n/s on the layers below.

We have one type of operation: advance every pebble on layer i through a block. This operation combines the two operations on Lemma E.4.14, and in fact it uses them internally. In order to apply an operation on layer i, we setup the pebbles on each layer below so that layer ℓ is the preimage of layer $\ell+1$ for $\ell \in [i-1]$. Then, for s rounds, we do the following, in ascending layer order. On an even layer, we advance each pebble to the next position. On an odd layer, we advance a pebble through whichever block happens to be the image of the previous layer.

Let $T(i)$ be the time of applying one operation on layer i and $\Sigma(i) = \sum_{\ell=1}^{i} T(i)$. The time to setup layer ℓ is at most the time of applying n/s^2 operations on layer ℓ, which gives a cost of $\sum_{\ell\in[i-1]} n/s^2 T(\ell)$ for the setup phase. Then for each round we only need to do s moves in each layer, which gives a cost of is^2 for s rounds.

We get the recurrence

$$\begin{aligned} T(1) &= s^2 & T(i) &= (n/s^2)\Sigma(i-1) + is^2 \\ \Sigma(1) &= T(1) & \Sigma(i) &= \Sigma(i-1) + T(i) \ , \end{aligned}$$

which we can solve to get

$$\begin{aligned} T(i) &= \sum_{\ell\in[i]} \binom{i}{\ell-1} n^{i-\ell}/s^{2(i-\ell-1)} \\ \Sigma(i) &= \sum_{\ell\in[i]} \binom{i+1}{\ell-1} n^{i-\ell}/s^{2(i-\ell-1)} \ . \end{aligned}$$

The total time to pebble the graph is the time needed for n/s operations on each layer from 1 to k, that is $(n/s)\Sigma(k) = n^k/s^{2k-3} + \mathrm{O}(n^{k-1}/s^{2k-5})$. □

Lemma E.4.17. *Let B be a k-layer bit-reversal graph. Then for any s such that $k+1 \leq s \leq \sqrt{n}$ there is a sequential black-white pebbling $\mathcal{P}$ of B in space $\Pi_s(\mathcal{P}) = 2k^2 s + \mathrm{O}(1)$ and time $\Pi_t(\mathcal{P}) \leq n^k/s^{2k-2} + \mathrm{O}(n^{k-1}/s^{2k-4})$.*

Proof. As in the proof of Lemma E.4.15, we begin by adding s white pebbles on each layer, so that we only need n/s^2 operations per layer. □

We prove the lower bounds for space smaller than $\sqrt{n}$ since for space larger than $\sqrt{n}$ we can just observe that a k-layer permutation graph contains a 2-layer permutation graph and use the lower bounds in [LT82].

Lemma E.4.18. *Let B be a k-layer bit-reversal graph. Let $\mathcal{P}$ be a sequential black pebbling of B in space $\Pi_s(\mathcal{P}) = s \leq \sqrt{n}/4$. Then*

$$\Pi_t(\mathcal{P}) \geq n^k/2^{3k}s^{2k-3} \ .$$

Proof. It is enough to prove that $T \geq n^k/2^{3k-1}s^{2k-3}$ for s a power of 2. Consider a pebbling in space s. We divide each path in $n/2s$ blocks of length $2s$.

We show that pebbling a block, this is starting with an empty block and placing a pebble on the sink, on the i-th layer requires time $n^{i-1}/2^{3i-2}s^{2i-4}$ for $i \geq 2$. Observe that to pebble the graph we have to consecutively pebble $n/2s$ blocks on the k-th layer, for a total of $n^k/2^{3k-1}s^{2k-3}$ as we wanted to show.

For $i = 2$, consider the $2s$ predecessors in layer 1 of the block we want to pebble. By construction of the bit-reversal permutation, the distance between these predecessors is $n/2s$. Since there are at most $s-1$ pebbles in layer 1, there are at least $s+1$ such predecessors whose closest pebble is at distance $n/2s$. Therefore, at least $n/2$ steps on layer 1 are needed before each predecessor has a pebble.

For $i > 2$, by the same argument there is at least one predecessor in layer $i-1$ whose closest pebble is at distance $n/2s$, therefore we can find $n/4s^2-2 \geq n/8s^2$ empty blocks in layer $i-1$ that need to be consecutively pebbled during the time interval we are considering. By induction hypothesis, pebbling a block on the $(i-1)$-th layer requires time $n^{i-2}/2^{3i-5}s^{2i-6}$. Since the blocks are pebbled in disjoint time intervals we can add the costs, so the total time is $n^{i-1}/2^{3i-2}s^{2i-4}$. □

Observe that in the induction step, even if we try to use the fact that there are $s+1$ blocks in layer $i-1$ that need to be pebbled, we cannot simply sum the individual cost of each block: their pebbling times can overlap, so a pebble placed in layer $i-1$ or below and used for one block in layer i may also be used for another block in layer i.

Lemma E.4.19. *Let B be a k-layer bit-reversal graph. Let $\mathcal{P}$ be a sequential black-white pebbling of B in space $\Pi_s(\mathcal{P}) = s \leq \sqrt{n}/4$. Then*

$$\Pi_t(\mathcal{P}) \geq n^k/2^{3k-3}s^{2k-2} \ .$$

Proof. It is enough to prove that $T \geq n^k/2^{3k-3}s^{2k-2}$ for s a power of 2. Consider a pebbling in space s. As in the proof of Lemma E.4.18, we divide each path in $n/2s$ blocks of length $2s$.

For black-white pebbling we say that a block is pebbled during some time interval if it is empty at the beginning, there is a pebble at the end of the block at some point in the interval, and there are only black pebbles left at the end of the interval. In contrast to the proof of Lemma E.4.18, we cannot assume that blocks need to be pebbled consecutively, but we can still show that, for any $i \in [k]$, in any time interval of length $n^{i-1}/2^{3i-4}s^{2i-4}$ at most s blocks are pebbled.

This is true for $i = 1$. For $i > 1$, consider a set of s blocks and the first time interval in which a block is completely pebbled. Arguing analogously to the proof of Lemma E.4.18, at the beginning of the interval there are s vertices in layer $i-1$ whose closest pebble is at distance $n/2s$, so we can find $s(n/4s^2-2) \geq n/8s$ blocks in layer $i-1$ that are also empty at the beginning of the interval. By induction hypothesis, we can find $n/8s^2$ sets of blocks that are being pebbled in consecutive time intervals, and each such interval lasts for at least $n^{i-2}/2^{3i-7}s^{2i-6}$ steps. Now, it is possible that some of these moves could be reused to make progress on some other block in layer i than the one we are considering, but since the pebbling uses only s pebbles and we assumed that this is the first block to be pebbled, there are at most $s-1$ other blocks on layer i that may also finish being pebbled, those that already had a pebble on them. At the end of the time interval, we discard these blocks from the set, consider the next time interval in which a block is completely pebbled, and repeat until we finish handling all blocks.

Observe that to pebble the graph we have to pebble $n/2s$ blocks on the k-th layer, and since we can only pebble at most s blocks in an interval of length $n^{k-1}/2^{3k-4}s^{2k-4}$ we need at least $n/2s^2$ intervals of this length. This gives a total of $n^k/2^{3k-3}s^{2k-2}$ steps, as stated in the lemma. □

E.5 Concluding Remarks

In this paper we study space complexity with a focus not on peak memory usage but on aggregated memory consumption over the whole computation. We consider two computational models, namely pebble games on DAGs and the resolution proof system in proof complexity.

For black-white pebbling we prove optimal cumulative space lower bounds and also time-space trade-offs where in order to achieve optimal time the space needs to be large not only at a single point in time but throughout essentially the whole computation. We do so by studying the concepts of *depth-robustness* and *dispersion* of graphs, drawing on and extending work in [AB16, AS15, ABP16] and other papers, and proving that different graph families of interest possess these properties.

In the context of proof complexity we are not aware of the cumulative space measure having been studied before, and so our first contribution here is to give a suitable formal definition, and also to consider different, more or less parallel, versions of the resolution proof system in which it makes sense to study cumulative space. We then use, and slightly extend, the reductions between pebbling and resolution in [BN08, BN11] to transfer our lower bounds and trade-off results for pebbling also to resolution.

Since, to the best of our knowledge, ours is the first paper to study cumulative space both for black-white pebbling and for proof complexity, it is perhaps not so surprising that there is a wealth of open problems that this paper does not resolve. Below, we briefly discuss some possible directions for future research.

One set of questions on which we make progress but which we do not answer completely concern the relation between maximal space and cumulative space. For sequential black-white pebblings of n-vertex DAGs we prove an optimal $\Omega(n^2)$ cumulative space lower bound for a particular family of DAGs, but for graphs that can be pebbled in maximal space $O(\log n)$ we only obtain a $\Omega(n^2/\log n)$ cumulative space lower bound and for graphs that can be pebbled in space $O(1)$ the best cumulative bound we can get is $\Omega(n^{3/2})$. Could it be the case that there are graphs that can be pebbled in maximal space $O(1)$ but nevertheless require cumulative space $\Omega(n^2)$? Or do strong enough cumulative space lower bounds by necessity imply also nontrivial maximal space lower bounds?

We briefly mentioned a pebble game between sequential black-white and parallel black-white pebbling, the parallel-black sequential-white game, as an example of how to apply the depth-robustness lemma to other pebbling models. Does this pebble game have other interesting properties or applications?

It has been shown for parallel black pebbling that extremal depth-robustness is both necessary and sufficient for a graph to have high cumulative space complexity. We prove that for black-white pebbling predecessor-robustness is sufficient to imply high cumulative space, but leave open whether this condition is necessary or not.

For standard time-space trade-offs in sequential pebbling, it was shown in [LT82] that bit-reversal DAGs have a black pebbling trade-off of the form $t = \Theta(n^2/s)$ whereas for black-white pebbling the trade-off is a slightly weaker $t = \Theta\big(n^2/s^2\big)$. It was conjectured in [LT82] that there are other permutation graphs for which the black-white pebbling trade-off could also be shown to be an optimal $t = \Theta\big(n^2/s\big)$. One natural candidate class of graphs to consider in this context are graphs obtained from random permutations, and this is the original reason why we were interested to study them in this paper. So far we were only able to obtain trade-offs with the same parameters as for bit-reversal DAGs, but it is an interesting question whether our tools could be sharpened to prove even stronger trade-offs results for random permutation graphs.

Turning to our proof complexity results, they can be seen to be yet another contribution to the sequence of papers [Nor09a, NH13, BN08, Nor12, BN11] obtaining space bounds and time-space trade-offs in proof complexity by instead studying pebble games and reductions between pebblings of DAGs and resolution refutations of so-called pebbling formulas defined in terms of these DAGs. While these connections have turned out to be very fruitful, it would also be interesting to go beyond pebbling formulas and explore whether cumulative space results could be obtained for, e.g., Tseitin formulas on long and narrow rectangular grids as studied in [BBI16, BNT13] or for other formulas.

One motivation behind our models of parallel resolution was the connection to parallel SAT solving, but our models do not take into account practical limitations such as the number of computing nodes or the communication between nodes. Could there be natural ways to incorporate such limitations, and could this also provide a better understanding of parallel resolution?

Another, somewhat related, question is whether formulas possessing strong cumulative space lower bounds are hard also in practice for (sequential or parallel) SAT solvers. Just maximal space lower bounds do not seem to be sufficient to imply practical hardness, as shown, e.g., in the fairly extensive empirical experiments on pebbling formulas in [JMNŽ12], but perhaps cumulative space could be a more relevant concept in this context.

Finally, it can be noted that our study of cumulative space in proof complexity as initiated in this paper is limited to the resolution proof system. This is mostly because resolution is the proof system where space complexity is best understood, and where the toolbox for studying these questions is most well developed. However, different concepts of maximal space and time-space trade-offs have been studied also for other proof systems such as polynomial calculus [ABRW02, BNT13, BG15, FLM$^+$13, FLN$^+$15] and cutting planes [dRNV16, GPT15, GP18, HN12], and it would be interesting to extend the study of cumulative space to these proof systems.

Acknowledgements

The authors wish to thank Yuval Filmus for making us aware of each other's existence and thus providing the stimulus for the joint work that led up to this paper. Different subsets of the authors are grateful to Ilario Bonacina, with whom we had stimulating discussions during various stages of this project and who, in particular, provided valuable insights on dispersion, and to Adam Schill Collberg and Jan Elffers for helpful discussions on random permutation graphs.

The first author performed part of this work while at KTH Royal Institute of Technology. The second, third and fourth authors were funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611. The third author was also supported by Swedish Research Council grants 621-2010-4797 and 621-2012-5645.

Bibliography

[AB87] Noga Alon and Ravi B. Boppana, *The monotone circuit complexity of Boolean functions*, Combinatorica **7** (1987), no. 1, 1–22.

[AB16] Joël Alwen and Jeremiah Blocki, *Efficiently computing data-independent memory-hard functions*, Proceedings of the 36th Annual International Cryptology Conference (CRYPTO '16), August 2016, pp. 241–271.

[ABdR$^+$18] Albert Atserias, Ilario Bonacina, Susanna F. de Rezende, Massimo Lauria, Jakob Nordström, and Alexander A. Razborov, *Clique is hard on average for regular resolution*, Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18), June 2018, pp. 866–877.

[ABP16] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak, *Depth-robust graphs and their cumulative memory complexity*, Tech. Report 2016/875, Cryptology ePrint Archive, September 2016.

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson, *Space complexity in propositional calculus*, SIAM Journal on Computing **31** (2002), no. 4, 1184–1211, Preliminary version in *STOC '00*.

[AC03] Noga Alon and Michael Capalbo, *Smaller explicit superconcentrators*, Internet Mathematics **1** (2003), no. 2, 151–163.

[AdRNV17] Joël Alwen, Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals, *Cumulative space in black-white pebbling and resolution*, Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS '17), Leibniz International Proceedings in Informatics (LIPIcs), vol. 67, January 2017, pp. 38:1–38:21.

[AH18] Albert Atserias and Tuomas Hakoniemi, *Size-degree trade-offs for sums-of-squares and Positivstellensatz proofs*, Tech. Report 1811.01351, arXiv.org, November 2018.

[AKS98] Noga Alon, Michael Krivelevich, and Benny Sudakov, *Finding a large hidden clique in a random graph*, Random Structures and Algorithms **13** (1998), no. 3-4, 457–466.

[ALN16] Albert Atserias, Massimo Lauria, and Jakob Nordström, *Narrow proofs may be maximally long*, ACM Transactions on Computational Logic **17** (2016), no. 3, 19:1–19:30, Preliminary version in *CCC '14*.

[And85] A. E. Andreev, *On a method for obtaining lower bounds for the complexity of individual monotone functions*, Soviet Mathematics Doklady **31** (1985), no. 3, 530–534, English translation of a paper in *Doklady Akademii Nauk SSSR*.

[And87] Ian Anderson, *Combinatorics of finite sets*, Oxford University Press, 1987.

[AO19] Albert Atserias and Joanna Ochremiak, *Proof complexity meets algebra*, ACM Transactions on Computational Logic **20** (2019), 1:1–1:46, Preliminary version in *ICALP '17*.

[AS15] Joël Alwen and Vladimir Serbinenko, *High parallel complexity graphs and memory-hard functions*, Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC '15), June 2015, pp. 595–603.

[BBG$^+$17] Patrick Bennett, Ilario Bonacina, Nicola Galesi, Tony Huynh, Mike Molloy, and Paul Wollan, *Space proof complexity for random* 3*-CNFs*, Information and Computation **255** (2017), 165–176.

[BBI16] Paul Beame, Chris Beck, and Russell Impagliazzo, *Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space*, SIAM Journal on Computing **45** (2016), no. 4, 1612–1645, Preliminary version in *STOC '12*.

[BC96] Samuel R. Buss and Peter Clote, *Cutting planes, connectivity and threshold logic*, Archive for Mathematical Logic **35** (1996), 33–63.

[BCE$^+$98] Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi, *The relative complexity of NP search problems*, Journal of Computer and System Sciences **57** (1998), no. 1, 3–19, Preliminary version in *STOC '95*.

[BCIP02] Joshua Buresh-Oppenheim, Matthew Clegg, Russell Impagliazzo, and Toniann Pitassi, *Homogenization and the polynomial calculus*, Computational Complexity **11** (2002), no. 3-4, 91–108, Preliminary version in *ICALP '00*.

[BEGJ00] María Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen, *On the relative complexity of resolution refinements and cutting planes proof* *systems*, SIAM Journal on Computing **30** (2000), no. 5, 1462–1484, Preliminary version in *FOCS '98*.

[Ben73] Charles H. Bennett, *Logical reversibility of computation*, IBM Journal of Research and Development **17** (1973), no. 6, 525–532.

[Ben89] ______, *Time/space trade-offs for reversible computation*, SIAM Journal on Computing **18** (1989), no. 4, 766–776.

[Ben09] Eli Ben-Sasson, *Size-space tradeoffs for resolution*, SIAM Journal on Computing **38** (2009), no. 6, 2511–2525, Preliminary version in *STOC '02*.

[Ber18] Christoph Berkholz, *The relation between polynomial calculus, Sherali-Adams, and sum-of-squares proofs*, Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS '18), Leibniz International Proceedings in Informatics (LIPIcs), vol. 96, February 2018, pp. 11:1–11:14.

[BFI$^+$18] Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere, *Stabbing planes*, Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS '18), Leibniz International Proceedings in Informatics (LIPIcs), vol. 94, January 2018, pp. 10:1–10:20.

[BG03] Eli Ben-Sasson and Nicola Galesi, *Space complexity of random formulae in resolution*, Random Structures and Algorithms **23** (2003), no. 1, 92–109, Preliminary version in *CCC '01*.

[BG15] Ilario Bonacina and Nicola Galesi, *A framework for space complexity in algebraic proof systems*, Journal of the ACM **62** (2015), no. 3, 23:1–23:20, Preliminary version in *ITCS '13*.

[BGL13] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria, *Parameterized complexity of DPLL search procedures*, ACM Transactions on Computational Logic **14** (2013), no. 3, 20:1–20:21, Preliminary version in *SAT '11*.

[BGLR12] Olaf Beyersdorff, Nicola Galesi, Massimo Lauria, and Alexander A. Razborov, *Parameterized bounded-depth Frege is not optimal*, ACM Transactions on Computation Theory **4** (2012), no. 3, 7:1–7:16, Preliminary version in *ICALP '11*.

[BGT14] Ilario Bonacina, Nicola Galesi, and Neil Thapen, *Total space in resolution*, Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS '14), October 2014, pp. 641–650.

[BHP10] Paul Beame, Trinh Huynh, and Toniann Pitassi, *Hardness amplification in proof complexity*, Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC '10), June 2010, pp. 87–96.

[BHvMW09] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh (eds.), *Handbook of satisfiability*, Frontiers in Artificial Intelligence and Applications, vol. 185, IOS Press, February 2009.

[BIK$^+$94] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák, *Lower bounds on Hilbert's Nullstellensatz and propositional proofs*, Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS '94), November 1994, pp. 794–806.

[BIK$^+$97] Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiri Sgall, *Proof complexity in algebraic systems and bounded depth Frege systems with modular counting*, Computational Complexity **6** (1997), no. 3, 256–298.

[BIS07] Paul Beame, Russell Impagliazzo, and Ashish Sabharwal, *The resolution complexity of independent sets and vertex covers in random graphs*, Computational Complexity **16** (2007), no. 3, 245–297, Preliminary version in *CCC '01*.

[BIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson, *Near optimal separation of tree-like and general resolution*, Combinatorica **24** (2004), no. 4, 585–603.

[BK73] Coen Bron and Joep Kerbosch, *Algorithm 457: Finding all cliques of an undirected graph*, Communications of the ACM **16** (1973), no. 9, 575–577.

[Bla37] Archie Blake, *Canonical expressions in Boolean algebra*, Ph.D. thesis, University of Chicago, 1937.

[BN08] Eli Ben-Sasson and Jakob Nordström, *Short proofs may be spacious: An optimal separation of space and length in resolution*, Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '08), October 2008, pp. 709–718.

[BN10] ______, *Understanding space in proof complexity: Separations and trade-offs via substitutions*, Technical Report TR10-125, Electronic Colloquium on Computational Complexity (ECCC), August 2010.

[BN11] ______, *Understanding space in proof complexity: Separations and trade-offs via substitutions*, Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11), January 2011, pp. 401–416.

[BN16] Christoph Berkholz and Jakob Nordström, *Near-optimal lower bounds on quantifier depth and Weisfeiler-Leman refinement steps*, Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '16), July 2016, pp. 267–276.

[BNT13] Chris Beck, Jakob Nordström, and Bangsheng Tang, *Some trade-off results for polynomial calculus*, Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13), May 2013, pp. 813–822.

[Bon16] Ilario Bonacina, *Total space in resolution is at least width squared*, Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP '16), Leibniz International Proceedings in Informatics (LIPIcs), vol. 55, July 2016, pp. 56:1–56:13.

[BP96] Paul Beame and Toniann Pitassi, *Simplified and improved resolution lower bounds*, Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS '96), October 1996, pp. 274–282.

[BP98] Samuel R. Buss and Toniann Pitassi, *Good degree bounds on Nullstellensatz refutations of the induction principle*, Journal of Computer and System Sciences **2** (1998), no. 57, 162–171, Preliminary version in *CCC '96*.

[BPR97] María Luisa Bonet, Toniann Pitassi, and Ran Raz, *Lower bounds for cutting planes proofs with small coefficients*, Journal of Symbolic Logic **62** (1997), no. 3, 708–728, Preliminary version in *STOC '95*.

[BPS07] Paul Beame, Toniann Pitassi, and Nathan Segerlind, *Lower bounds for Lovász–Schrijver systems and beyond follow from multiparty communication complexity*, SIAM Journal on Computing **37** (2007), no. 3, 845–869, Preliminary version in *ICALP '05*.

[BS90] Ravi B. Boppana and Michael Sipser, *The complexity of finite functions*, Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity (J. van Leeuwen, ed.), MIT Press, 1990, pp. 757–804.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag, *Using CSP look-back techniques to solve real-world SAT instances*, Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97), July 1997, pp. 203–208.

[BSD$^+$19] Debjyoti Bhattacharjee, Mathias Soeken, Srijit Dutta, Anupam Chattopadhyay, and Giovanni De Micheli, *Reversible pebble games for reducing qubits in hierarchical quantum circuit synthesis*, 49th IEEE International Symposium on Multiple-Valued Logic, ISMVL Fredericton, Canada, 2019.

[BTV01] Harry Buhrman, John Tromp, and Paul Vitányi, *Time and space bounds for reversible simulation*, Journal of physics A: Mathematical and general **34** (2001), 6821–6830, Preliminary version in *ICALP '01*.

[Bus98] Samuel R. Buss, *Lower bounds on Nullstellensatz proofs via designs*, Proof Complexity and Feasible Arithmetics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 39, American Mathematical Society, 1998, Available at `http://www.math.ucsd.edu/~sbuss/ResearchWeb/designs/`, pp. 59–71.

[BW01] Eli Ben-Sasson and Avi Wigderson, *Short proofs are narrow—resolution made simple*, Journal of the ACM **48** (2001), no. 2, 149–169, Preliminary version in *STOC '99*.

[CCT87] William Cook, Collette R. Coullard, and György Turán, *On the complexity of cutting-plane proofs*, Discrete Applied Mathematics **18** (1987), no. 1, 25–38.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo, *Using the Groebner basis algorithm to find proofs of unsatisfiability*, Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96), May 1996, pp. 174–183.

[Cha73] Ashok K. Chandra, *Efficient compilation of linear recursive programs*, Proceedings of the 14th Annual Symposium on Switching and Automata Theory (SWAT '73), 1973, pp. 16–25.

[Cha13] Siu Man Chan, *Just a pebble game*, Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC '13), June 2013, pp. 133–143.

[CHKX04] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia, *Linear FPT reductions and computational lower bounds*, Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC '04), June 2004, pp. 212–221.

[Chu36] Alonzo Church, *An unsolvable problem of elementary number theory*, American Journal of Mathematics **58** (1936), no. 2, 345–363.

[Chv73] Vašek Chvátal, *Edmonds polytopes and a hierarchy of combinatorial problems*, Discrete Mathematics **4** (1973), no. 1, 305–337.

[CKLM18] Arkadev Chattopadhyay, Michal Koucky, Bruno Loff, and Sagnik Mukhopadhyay, *Simulation beats richness: New data-structure lower bounds*, Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18), June 2018, pp. 1013–1020.

[CLNV15] Siu Man Chan, Massimo Lauria, Jakob Nordström, and Marc Vinyals, *Hardness of approximation in PSPACE and separation results for pebble* *games (Extended abstract)*, Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15), October 2015, pp. 466–485.

[Coo71] Stephen A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71), 1971, pp. 151–158.

[Coo74] ______, *An observation on time-storage trade off*, Journal of Computer and System Sciences **9** (1974), no. 3, 308–316, Preliminary version in *STOC '73*.

[CP90] Randy Carraghan and Panos M. Pardalos, *An exact algorithm for the maximum clique problem*, Operations Research Letters **9** (1990), no. 6, 375–382.

[CP14] Siu Man Chan and Aaron Potechin, *Tight bounds for monotone switching networks via Fourier analysis*, Theory of Computing **10** (2014), 389–419, Preliminary version in *STOC '12*.

[CR79] Stephen A. Cook and Robert Reckhow, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic **44** (1979), no. 1, 36–50, Preliminary version in *STOC '74*.

[CS76] Stephen A. Cook and Ravi Sethi, *Storage requirements for deterministic polynomial time recognizable languages*, Journal of Computer and System Sciences **13** (1976), no. 1, 25–37, Preliminary version in *STOC '74*.

[CS80] David A. Carlson and John E. Savage, *Graph pebbling with many free pebbles can be difficult*, Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC '80), 1980, pp. 326–332.

[CS82] ______, *Extreme time-space tradeoffs for graphs with small space requirements*, Information Processing Letters **14** (1982), no. 5, 223–227.

[CS88] Vašek Chvátal and Endre Szemerédi, *Many hard examples for resolution*, Journal of the ACM **35** (1988), no. 4, 759–768.

[CZ12] Renato Carmo and Alexandre Züge, *Branch and bound algorithms for the maximum clique problem under a unified framework*, Journal of the Brazilian Computer Society **18** (2012), no. 2, 137–151.

[DF95] Rodney Downey and Michael R. Fellows, *Fixed-parameter tractability and completeness II: Completeness for W[1]*, Theoretical Computer Science A **141** (1995), no. 1–2, 109–131.

[DM18] Irit Dinur and Or Meir, *Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity*, Computational Complexity **27** (2018), no. 3, 375–462.

[DMR09] Stefan S. Dantchev, Barnaby Martin, and Martin Rhodes, *Tight rank lower bounds for the Sherali–Adams proof system*, Theoretical Computer Science **410** (2009), no. 21–23, 2054–2063.

[DMS11] Stefan S. Dantchev, Barnaby Martin, and Stefan Szeider, *Parameterized proof complexity*, Computational Complexity **20** (2011), 51–85, Preliminary version in *FOCS '07*.

[DNW05] Cynthia Dwork, Moni Naor, and Hoeteck Wee, *Pebbling and proofs of work*, Proceedings of the 25th Annual International Cryptology Conference (CRYPTO '05), Lecture Notes in Computer Science, vol. 3621, Springer, August 2005, pp. 37–54.

[dRMN$^+$19] Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals, *Lifting with simple gadgets and applications to circuit and proof complexity*, Submitted manuscript, 2019.

[dRMNR19] Susanna F. de Rezende, Or Meir, Jakob Nordström, and Robert Robere, *Nullstellensatz size-degree trade-offs from reversible pebbling*, Proceedings of the 34th Annual IEEE Conference on Computational Complexity (CCC '19), July 2019, pp. 18:1–18:16.

[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)*, Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS '16), October 2016, pp. 295–304.

[DT85] Patrick W. Dymond and Martin Tompa, *Speedups of deterministic machines by synchronous parallel machines*, Journal of Computer and System Sciences **30** (1985), no. 2, 149–161, Preliminary version in *STOC '83*.

[DvMW11] Scott Diehl, Dieter van Melkebeek, and Ryan Williams, *An improved time-space lower bound for tautologies*, Journal of Combinatorial Optimization **22** (2011), no. 3, 325–338, Preliminary version in *COCOON '09*.

[EGS75] Paul Erdős, Ronald L. Graham, and Endre Szemerédi, *On sparse graphs with dense long paths.*, Tech. report, Stanford University, 1975.

[EIRS01] Jeff Edmonds, Russell Impagliazzo, Steven Rudich, and Jirí Sgall, *Communication complexity towards lower bounds on circuit depth*, Computational Complexity **10** (2001), no. 3, 210–246.

[ET01] Juan Luis Esteban and Jacobo Torán, *Space bounds for resolution*, Information and Computation **171** (2001), no. 1, 84–97, Preliminary versions of these results appeared in *STACS '99* and *CSL '99*.

[Fah02] Torsten Fahle, *Simple and fast: Improving a branch-and-bound algorithm for maximum clique*, Proceedings of the 10th Annual European Symposium on Algorithms (ESA '02), Lecture Notes in Computer Science, vol. 2461, 2002, pp. 485–498.

[FLM$^+$13] Yuval Filmus, Massimo Lauria, Mladen Mikša, Jakob Nordström, and Marc Vinyals, *Towards an understanding of polynomial calculus: New separations and lower bounds (Extended abstract)*, Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP '13), Lecture Notes in Computer Science, vol. 7965, Springer, July 2013, pp. 437–448.

[FLN$^+$15] Yuval Filmus, Massimo Lauria, Jakob Nordström, Noga Ron-Zewi, and Neil Thapen, *Space complexity in polynomial calculus*, SIAM Journal on Computing **44** (2015), no. 4, 1119–1153, Preliminary version in *CCC '12*.

[For00] Lance Fortnow, *Time-space tradeoffs for satisfiability*, Journal of Computer and System Sciences **60** (2000), no. 2, 337–353, Preliminary version in *CCC '97*.

[FPRC13] Yuval Filmus, Toniann Pitassi, Robert Robere, and Stephen A. Cook, *Average case lower bounds for monotone switching networks*, Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS '13), November 2013, pp. 598–607.

[Fra84] Péter Frankl, *A new short proof for the Kruskal–Katona theorem*, Discrete Mathematics **48** (1984), no. 2, 327–329.

[Fre93] Gottlob Frege, *Grundgesetze der arithmetik*, vol. 1, H. Pohle, 1893.

[Gál01] Anna Gál, *A characterization of span program size and improved lower bounds for monotone span programs*, Computational Complexity **10** (2001), no. 4, 277–296, Preliminary version in *STOC '98*.

[GG81] Ofer Gabber and Zvi Galil, *Explicit constructions of linear-sized superconcentrators*, Journal of Computer and System Sciences **22** (1981), no. 3, 407–420.

[GGKS18] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov, *Monotone circuit lower bounds from resolution*, Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18), June 2018, pp. 902–911.

[GHP02] Dima Grigoriev, Edward A. Hirsch, and Dmitrii V. Pasechnik, *Exponential lower bound for static semi-algebraic proofs*, Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP '02), Lecture Notes in Computer Science, vol. 2380, Springer, July 2002, pp. 257–268.

[GHR92] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov, *Majority gates VS. general weighted threshold gates*, Computational Complexity **2** (1992), 277–300, Preliminary version in *CCC '92*.

[GJPW17] Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson, *Randomized communication vs. partition number*, Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP '17), Leibniz International Proceedings in Informatics (LIPIcs), vol. 80, July 2017, pp. 52:1–52:15.

[GJW18] Mika Göös, Rahul Jain, and Thomas Watson, *Extension complexity of independent set polytopes*, SIAM Journal on Computing **47** (2018), no. 1, 241–269.

[GKPW17] Mika Göös, Pritish Kamath, Toniann Pitassi, and Thomas Watson, *Query-to-communication lifting for P^NP*, Proceedings of the 32nd Annual Computational Complexity Conference (CCC '17), Leibniz International Proceedings in Informatics (LIPIcs), vol. 79, July 2017, pp. 12:1–12:16.

[GKRS18] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov, *Adventures in monotone complexity and TFNP*, Technical Report TR18-163, Electronic Colloquium on Computational Complexity (ECCC), September 2018.

[GLM$^+$16] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman, *Rectangles are nonnegative juntas*, SIAM Journal on Computing **45** (2016), no. 5, 1835–1869, Preliminary version in *STOC '15*.

[GLT80] John R. Gilbert, Thomas Lengauer, and Robert E. Tarjan, *The pebbling problem is complete in polynomial space*, SIAM Journal on Computing **9** (1980), no. 3, 513–524, Preliminary version in *STOC '79*.

[GMWW17] Dmitry Gavinsky, Or Meir, Omri Weinstein, and Avi Wigderson, *Toward better formula lower bounds: The composition of a function and a universal relation*, SIAM Journal on Computing **46** (2017), no. 1, 114–131.

[Göd31] Kurt Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, Monatshefte für Mathematik und Physik **38** (1931), no. 1, 173–198.

[Gom63] Ralph E. Gomory, *An algorithm for integer solutions of linear programs*, Recent Advances in Mathematical Programming (R.L. Graves and P. Wolfe, eds.), McGraw-Hill, New York, 1963, pp. 269–302.

[GP18] Mika Göös and Toniann Pitassi, *Communication lower bounds via critical block sensitivity*, SIAM Journal on Computing **47** (2018), no. 5, 1778–1806, Preliminary version in *STOC '14*.

[GPT15] Nicola Galesi, Pavel Pudlák, and Neil Thapen, *The space complexity of cutting planes refutations*, Proceedings of the 30th Annual Computational Complexity Conference (CCC '15), Leibniz International Proceedings in Informatics (LIPIcs), vol. 33, June 2015, pp. 433–447.

[GPW15] Mika Göös, Toniann Pitassi, and Thomas Watson, *Deterministic communication vs. partition number*, Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15), October 2015, pp. 1077–1088.

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson, *The landscape of communication complexity classes*, Computational Complexity **27** (2018), no. 2, 245–304, Preliminary version in *ICALP '16*.

[GS92] Michelangelo Grigni and Michael Sipser, *Monotone complexity*, Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity, Cambridge University Press, 1992, pp. 57–75.

[GT78] John R. Gilbert and Robert E. Tarjan, *Variations of a pebble game on graphs*, Technical Report STAN-CS-78-661, Stanford University, 1978, Available at `http://infolab.stanford.edu/TR/CS-TR-78-661.html`.

[Hak85] Armin Haken, *The intractability of resolution*, Theoretical Computer Science **39** (1985), no. 2-3, 297–308.

[Hås99] Johan Håstad, *Clique is hard to approximate within $n^{1-\epsilon}$*, Acta Mathematica **182** (1999), 105–142, Preliminary version in *FOCS '96*.

[HN12] Trinh Huynh and Jakob Nordström, *On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity (Extended abstract)*, Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12), May 2012, pp. 233–248.

[HP18] Pavel Hrubeš and Pavel Pudlák, *A note on monotone real circuits*, Information Processing Letters **131** (2018), 15–19.

[HPV77] John Hopcroft, Wolfgang Paul, and Leslie Valiant, *On time versus space*, Journal of the ACM **24** (1977), no. 2, 332–337, Preliminary version in *FOCS '75*.

[HW93] Johan Håstad and Avi Wigderson, *Composition of the universal relation*, Advances In Computational Complexity Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, American Mathematical Society, 1993, pp. 119–134.

[IP01] Russell Impagliazzo and Ramamohan Paturi, *On the complexity of k-SAT*, Journal of Computer and System Sciences **62** (2001), no. 2, 367–375, Preliminary version in *CCC '99*.

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall, *Lower bounds for the polynomial calculus and the Gröbner basis algorithm*, Computational Complexity **8** (1999), no. 2, 127–144.

[JMNŽ12] Matti Järvisalo, Arie Matsliah, Jakob Nordström, and Stanislav Živný, *Relating proof complexity measures and practical hardness of SAT*, Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12), Lecture Notes in Computer Science, vol. 7514, Springer, October 2012, pp. 316–331.

[Joh98] Jan Johannsen, *Lower bounds for monotone real circuit depth and formula size and tree-like cutting planes*, Information Processing Letters **67** (1998), no. 1, 37–41.

[Joh01] ______, *Depth lower bounds for monotone semi-unbounded fan-in circuits*, RAIRO-Theoretical Informatics and Applications **35** (2001), no. 3, 277–286.

[Juk11] Stasys Jukna, *Extremal combinatorics with applications in computer science*, 2nd ed., Springer, 2011.

[Kar72] Richard M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations, The IBM Research Symposia Series, Springer, 1972, pp. 85–103.

[Kar76] ______, *The probabilistic analysis of some combinatorial search algorithms*, Algorithms and Complexity: New Directions and Recent Results, Academic Press, New York, 1976, pp. 1–19.

[Kat68] Gyula O. H. Katona, *A theorem of finite sets*, Theory of Graphs, Akadémiai Kiadó, 1968, pp. 187–207.

[KI06] Arist Kojevnikov and Dmitry Itsykson, *Lower bounds of static Lovász–Schrijver calculus proofs for Tseitin tautologies*, Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP '06), Lecture Notes in Computer Science, vol. 4051, Springer, July 2006, pp. 323–334.

[KJ07] Janez Konc and Dušanka Janežič, *An improved branch and bound algorithm for the maximum clique problem*, MATCH Communications in Mathematical and in Computer Chemistry **58** (2007), 569–590.

[KM18] Sajin Koroth and Or Meir, *Improved composition theorems for functions and relations*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM '18), Leibniz International Proceedings in Informatics (LIPIcs), vol. 116, August 2018, pp. 48:1–48:18.

[KMR17] Pravesh K. Kothari, Raghu Meka, and Prasad Raghavendra, *Approximating rectangles by juntas and weakly-exponential lower bounds for LP relaxations of CSPs*, Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC '17), June 2017, pp. 590–603.

[KN97] Eyal Kushilevitz and Noam Nisan, *Communication complexity*, Cambridge University Press, 1997.

[Knu94] Donald E. Knuth, *The sandwich theorem*, The Electronic Journal of Combinatorics **1** (1994), no. A1, 1–48.

[KPPY84] Maria Klawe, Wolfgang J. Paul, Nicholas Pippenger, and Mihalis Yannakakis, *On monotone formulae with restricted depth*, Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC '84), 1984, pp. 480–487.

[Kra95a] Jan Krajíček, *Bounded arithmetic, propositional logic, and complexity theory*, Cambridge University Press, New York, 1995.

[Kra95b] ———, *On Frege and extended Frege proof systems*, Feasible Mathematics II, 1995, pp. 284–319.

[Kra97] ———, *Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic*, Journal of Symbolic Logic **62** (1997), no. 2, 457–486.

[Kra98] ———, *Interpolation by a game*, Mathematical Logic Quarterly **44** (1998), no. 4, 450–458.

[Krá04] Richard Královič, *Time and space complexity of reversible pebbling*, RAIRO – Theoretical Informatics and Applications **38** (2004), no. 02, 137–161.

[Kru63] Joseph B. Kruskal, *The number of simplices in a complex*, Mathematical Optimization Techniques (Richard Bellman, ed.), University of California Press, 1963, pp. 251–278.

[KRW95] Mauricio Karchmer, Ran Raz, and Avi Wigderson, *Super-logarithmic depth lower bounds via the direct sum in communication complexity*, Computational Complexity **5** (1995), no. 3/4, 191–204, Preliminary version in *CCC '91*.

[KS90] Bala Kalyanasundaram and Georg Schnitger, *Rounds versus time for the two person pebble game*, Information and Computation **88** (1990), no. 1, 1–17, Preliminary version in *STACS '89*.

[KSS18] Balagopal Komarath, Jayalal Sarma, and Saurabh Sawlani, *Pebbling meets coloring: Reversible pebble game on trees*, Journal of Computer and System Sciences **91** (2018), 33–41.

[KSSS13] George Katsirelos, Ashish Sabharwal, Horst Samulowitz, and Laurent Simon, *Resolution and parallelizability: Barriers to the efficient parallelization of SAT solvers*, Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI '13), July 2013.

[Kuč95] Luděk Kučera, *Expected complexity of graph partitioning problems*, Discrete Applied Mathematics **57** (1995), no. 2, 193–212.

[KW90] Mauricio Karchmer and Avi Wigderson, *Monotone circuits for connectivity require super-logarithmic depth*, SIAM Journal on Discrete Mathematics **3** (1990), no. 2, 255–265, Preliminary version in *STOC '88*.

[Lev73] Leonid A. Levin, *Universal sequential search problems*, Problemy peredachi informatsii **9** (1973), no. 3, 115–116, In Russian. Available at `http://mi.mathnet.ru/ppi914`.

[LLMO09] Jesús A. De Loera, Jon Lee, Susan Margulies, and Shmuel Onn, *Expressing combinatorial problems by systems of polynomial equations and Hilbert's Nullstellensatz*, Combinatorics, Probability and Computing **18** (2009), no. 4, 551–582.

[LM19] Bruno Loff and Sagnik Mukhopadhyay, *Lifting theorems for equality*, Proceedings of the 36th Symposium on Theoretical Aspects of Computer Science (STACS '19), Leibniz International Proceedings in Informatics (LIPIcs), vol. 126, March 2019, pp. 50:1–50:19.

[LMT00] Klaus-Jörn Lange, Pierre McKenzie, and Alain Tapp, *Reversible space equals deterministic space*, Journal of Computer and System Sciences **60** (2000), no. 2, 354–367.

[LNNW95] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson, *Search problems in the decision tree model*, SIAM Journal on Discrete Mathematics **8** (1995), no. 1, 119–132, Preliminary version in *FOCS '91*.

[Lov79] László Lovász, *On the shannon capacity of a graph*, IEEE Transactions on Information theory **25** (1979), no. 1, 1–7.

[LPRT17] Massimo Lauria, Pavel Pudlák, Vojtěch Rödl, and Neil Thapen, *The complexity of proving that a graph is Ramsey*, Combinatorica **37** (2017), no. 2, 253–268, Preliminary version in *ICALP '13*.

[LRS15] James R. Lee, Prasad Raghavendra, and David Steurer, *Lower bounds on the size of semidefinite programming relaxations*, Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC '15), June 2015, pp. 567–576.

[LS90] Robert Y. Levin and Alan T. Sherman, *A note on Bennetts time-space tradeoff for reversible computation*, SIAM Journal on Computing **19** (1990), 673–677.

[LT82] Thomas Lengauer and Robert E. Tarjan, *Asymptotically tight bounds on time-space trade-offs in a pebble game*, Journal of the ACM **29** (1982), no. 4, 1087–1130, Preliminary version in *STOC '79*.

[LTV98] Ming Li, John Tromp, and Paul Vitányi, *Reversible simulation of irreversible computation*, Physica D: Nonlinear Phenomena **120** (1998), no. 1–2, 168–176.

[LV96] Ming Li and Paul Vitányi, *Reversibility and adiabatic computation: Trading time and space for energy*, Proceedings of the Royal Society of London, Series A **452** (1996), no. 1947, 769–789.

[McC17] Ciaran McCreesh, *Solving hard subgraph problems in parallel*, Ph.D. thesis, University of Glasgow, 2017.

[Mil00] *The Millennium Problems of the Clay Mathematics Institute*, May 2000, See `http://www.claymath.org/millennium-problems`.

[MMZ$^+$01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik, *Chaff: Engineering an efficient SAT solver*, Proceedings of the 38th Design Automation Conference (DAC '01), June 2001, pp. 530–535.

[MS72] George Marsaglia and George P. H. Styan, *When does* $\mathrm{rank}(A + B) = \mathrm{rank}(A) + \mathrm{rank}(B)$*?*, Canadian Mathematical Bulletin **15** (1972), no. 3, 451–452.

[MS99] João P. Marques-Silva and Karem A. Sakallah, *GRASP: A search algorithm for propositional satisfiability*, IEEE Transactions on Computers **48** (1999), no. 5, 506–521, Preliminary version in *ICCAD '96*.

[MSR$^+$18] Giulia Meuli, Mathias Soeken, Martin Roetteler, Nikolaj Bjorner, and Giovanni De Micheli, *Reversible pebbling game for quantum memory management*, Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2018.

[Mur71] Saburo Muroga, *Threshold logic and its applications*, Wiley, 1971.

[NH13] Jakob Nordström and Johan Håstad, *Towards an optimal separation of space and length in resolution*, Theory of Computing **9** (2013), 471–557, Preliminary version in *STOC '08*.

[NÖ03] Sampo Niskanen and Patric R. J. Östergård, *Cliquer User's Guide, Version 1.0*, Tech. Report T48, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, 2003.

[Nor09a] Jakob Nordström, *Narrow proofs may be spacious: Separating space and width in resolution*, SIAM Journal on Computing **39** (2009), no. 1, 59–121, Preliminary version in *STOC '06*.

[Nor09b] ______, *A simplified way of proving trade-off results for resolution*, Information Processing Letters **109** (2009), no. 18, 1030–1035, Preliminary version in ECCC report TR07-114, 2007.

[Nor12] ______, *On the relative strength of pebbling and resolution*, ACM Transactions on Computational Logic **13** (2012), no. 2, 16:1–16:43, Preliminary version in *CCC '10*.

[Nor13] ______, *Pebble games, proof complexity and time-space trade-offs*, Logical Methods in Computer Science **9** (2013), no. 3, 15:1–15:63.

[Nor19] ______, *New wine into old wineskins: A survey of some pebbling classics with supplemental results*, Manuscript in preparation. To appear in *Foundations and Trends in Theoretical Computer Science*. Current draft version available at `http://www.csc.kth.se/~jakobn/research/`, 2019.

[NP85] Jaroslav Nešetřil and Svatopluk Poljak, *On the complexity of the subgraph problem*, Commentationes Mathematicae Universitatis Carolinae **026** (1985), no. 2, 415–419.

[NW93] Noam Nisan and Avi Wigderson, *Rounds in communication complexity revisited*, SIAM Journal on Computing **22** (1993), no. 1, 211–219, Preliminary version in *STOC '91*.

[Öst02] Patric R. J. Östergård, *A fast algorithm for the maximum clique problem*, Discrete Applied Mathematics **120** (2002), no. 1–3, 197–207.

[PH70] Michael S. Paterson and Carl E. Hewitt, *Comparative schematology*, Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, 1970, pp. 119–127.

[Pip77] Nicholas Pippenger, *Superconcentrators*, SIAM Journal on Computing **6** (1977), no. 2, 298–304.

[Pip80] ______, *Pebbling*, Technical Report RC8258, IBM Watson Research Center, 1980, in Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science, Japan.

[Pit16] Toniann Pitassi, Manuscript, 2016.

[Pot10] Aaron Potechin, *Bounds on monotone switching networks for directed connectivity*, Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '10), October 2010, pp. 553–562.

[PR80] Wolfgang J. Paul and Rüdiger Reischuk, *On alternation II. A graph theoretic approach to determinism versus non-determinism*, Acta Informatica **14** (1980), no. 4, 391–403, Preliminary version in *GITCS '79*.

[PR17] Toniann Pitassi and Robert Robere, *Strongly exponential lower bounds for monotone computation*, Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC '17), June 2017, pp. 1246–1255.

[PR18] ______, *Lifting Nullstellensatz to monotone span programs over any field*, Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18), June 2018, pp. 1207–1219.

[Pro12] Patrick Prosser, *Exact algorithms for maximum clique: A computational study*, Algorithms **5** (2012), no. 4, 545–587.

[PS98] Pavel Pudlák and Jirí Sgall, *Algebraic models of computation and interpolation for algebraic proof systems*, Proof Complexity and Feasible Arithmetics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 39, American Mathematical Society, 1998, Available at `http://users.math.cas.cz/~pudlak/span.pdf`, pp. 279–296.

[PTC77] Wolfgang J. Paul, Robert E. Tarjan, and James R. Celoni, *Space bounds for a game on graphs*, Mathematical Systems Theory **10** (1977), 239–251.

[Pud97] Pavel Pudlák, *Lower bounds for resolution and cutting plane proofs and monotone computations*, Journal of Symbolic Logic **62** (1997), no. 3, 981–998.

[PV76] Nicholas Pippenger and Leslie G. Valiant, *Shifting graphs and their applications*, Journal of the ACM **23** (1976), no. 3, 423–432.

[Raz85] Alexander A. Razborov, *Lower bounds for the monotone complexity of some Boolean functions*, Soviet Mathematics Doklady **31** (1985), no. 2, 354–357, English translation of a paper in *Doklady Akademii Nauk SSSR*.

[Raz90] ______, *Applications of matrix methods to the theory of lower bounds in computational complexity*, Combinatorica **10** (1990), no. 1, 81–93.

[RM99] Ran Raz and Pierre McKenzie, *Separation of the monotone NC hierarchy*, Combinatorica **19** (1999), no. 3, 403–435, Preliminary version in *FOCS '97*.

[Rob18] Robert Robere, *Unified lower bounds for monotone computation*, Ph.D. thesis, University of Toronto, 2018.

[Ros97] Arnold Rosenbloom, *Monotone real circuits are more powerful than monotone Boolean circuits*, Information Processing Letters **61** (1997), no. 3, 161–164.

[Ros08] Benjamin Rossman, *On the constant-depth complexity of k-clique*, Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08), May 2008, pp. 721–730.

[Ros10] ______, *Average-case complexity of detecting cliques*, Ph.D. thesis, Masschussets Institute of Technology, 2010.

[Ros14] ______, *The monotone complexity of k-clique on random graphs*, SIAM Journal on Computing **43** (2014), no. 1, 256–279, Preliminary version in *FOCS '10*.

[RPRC16] Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook, *Exponential lower bounds for monotone span programs*, Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS '16), October 2016, pp. 406–415.

[RWY02] Alexander A. Razborov, Avi Wigderson, and Andrew Yao, *Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus*, Combinatorica **22** (2002), no. 4, 555–574.

[RY16] Anup Rao and Amir Yehudayoff, *Communication complexity*, Manuscript in preparation, 2016.

[S$^+$17] William A. Stein et al., *Sage Mathematics Software (Version 8.1)*, The Sage Development Team, 2017.

[San01] Rahul Santhanam, *Lower bounds on the complexity of recognizing SAT by Turing machines*, Information Processing Letters **79** (2001), no. 5, 243–247.

[Sav98] John E. Savage, *Models of computation: Exploring the power of computing*, Addison-Wesley, 1998, Available at `http://www.modelsofcomputation.org`.

[Sch83] Georg Schnitger, *On depth-reduction and grates*, Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science (FOCS '83), November 1983, pp. 323–328.

[Set75] Ravi Sethi, *Complete register allocation problems*, SIAM Journal on Computing **4** (1975), no. 3, 226–248.

[She11] Alexander A. Sherstov, *The pattern matrix method*, SIAM Journal on Computing **40** (2011), no. 6, 1969–2000, Preliminary version in *STOC '08*.

[SLB14] Pablo San Segundo, Alvaro Lopez, and Mikhail Batsyn, *Initial sorting of vertices in the maximum clique problem reviewed*, 8th International Conference on Learning and Intelligent Optimization (LION '14), Selected Revised Papers, Lecture Notes in Computer Science, vol. 8426, Springer, 2014, pp. 111–120.

[SLB$^+$16] Pablo San Segundo, Alvaro Lopez, Mikhail Batsyn, Alexey Nikolaev, and Panos M. Pardalos, *Improved initial vertex ordering for exact maximum clique search*, Applied Intelligence **45** (2016), no. 3, 868–880.

[SMRH13] Pablo San Segundo, Fernando Matia, Diego Rodríguez-Losada, and Miguel Hernando, *An improved bit parallel exact maximum clique algorithm*, Optimization Letters **7** (2013), no. 3, 467–479.

[Sok17] Dmitry Sokolov, *Dag-like communication and its applications*, Proceedings of the 12th International Computer Science Symposium in Russia (CSR '17), Lecture Notes in Computer Science, vol. 10304, Springer, June 2017, pp. 294–307.

[SRJ11] Pablo San Segundo, Diego Rodríguez-Losada, and Agustín Jiménez, *An exact bit-parallel algorithm for the maximum clique problem*, Computers and Operations Research **38** (2011), no. 2, 571–581.

[SS77] Sowmitri Swamy and John E. Savage, *Space-time trade-offs on the FFT-algorithm*, Technical Report CS-31, Brown University, 1977.

[SS79] John E. Savage and Sowmitri Swamy, *Space-time tradeoffs for oblivious interger multiplications*, Proceedings of the 6th International Colloquium on Automata, Languages and Programming (ICALP '79), 1979, pp. 498–504.

[SS83] Sowmitri Swamy and John E. Savage, *Space-time tradeoffs for linear recursion*, Mathematical Systems Theory **16** (1983), no. 1, 9–27.

[ST10] Pablo San Segundo and Cristóbal Tapia, *A new implicit branching strategy for exact maximum clique*, Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI '10), vol. 1, 2010, pp. 352–357.

[Tha16] Neil Thapen, *A trade-off between length and width in resolution*, Theory of Computing **12** (2016), no. 5, 1–14.

[TK07] Etsuji Tomita and Toshikatsu Kameda, *An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments*, Journal of Global Optimization **37** (2007), no. 1, 95–111.

[Tom78] Martin Tompa, *Time-space tradeoffs for computing functions, using connectivity properties of their circuits*, Proceedings of the 10th annual ACM symposium on Theory of computing (STOC '78), 1978, pp. 196–204.

[TS03] Etsuji Tomita and Tomokazu Seki, *An efficient branch-and-bound algorithm for finding a maximum clique*, Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science (DMTCS '03), vol. 3, 2003, pp. 278–289.

[TSH$^+$10] Etsuji Tomita, Yoichi Sutani, Takanori Higashi, Shinya Takahashi, and Mitsuo Wakatsuki, *A simple and faster branch-and-bound algorithm for finding a maximum clique*, Proceedings of the 4th International Workshop on Algorithms and Computation (WALCOM '10), Lecture Notes in Computer Science, vol. 5942, Springer, 2010, pp. 191–203.

[Tur37] Alan M. Turing, *On computable numbers, with an application to the Entscheidungsproblem*, Proceedings of the London mathematical society **2** (1937), no. 1, 230–265.

[TYH$^+$16] Etsuji Tomita, Kohei Yoshida, Takuro Hatta, Atsuki Nagao, Hiro Ito, and Mitsuo Wakatsuki, *A much faster branch-and-bound algorithm for finding a maximum clique*, Proceedings of the 10th International Workshop on Frontiers in Algorithmics (FAW '16), Lecture Notes in Computer Science, vol. 9711, Springer, June 2016, pp. 215–226.

[Urq87] Alasdair Urquhart, *Hard examples for resolution*, Journal of the ACM **34** (1987), no. 1, 209–219.

[Val75a] Leslie G. Valiant, *On non-linear lower bounds in computational complexity*, Proceedings of the 7th annual ACM symposium on Theory of computing (STOC '75), 1975, pp. 45–53.

[Val75b] ______, *Parallelism in comparison problems*, SIAM Journal on Computing **4** (1975), no. 3, 348–355.

[Val77] ______, *Graph-theoretic arguments in low-level complexity*, Proceedings of the 6th International Symposium on Mathematical Foundations of Computer Science (MFCS '77), Lecture Notes in Computer Science, vol. 53, Springer, September 1977, pp. 162–176.

[Vas09] Virginia Vassilevska, *Efficient algorithms for clique problems*, Information Processing Letters **109** (2009), no. 4, 254–257.

[VH67] Jean Van Heijenoort, *From Frege to Gödel: a source book in mathematical logic, 1879-1932*, Harvard University Press, 1967.

[vM07] Dieter van Melkebeek, *A survey of lower bounds for satisfiability and related problems*, Foundations and Trends in Theoretical Computer Science **2** (2007), no. 3, 197–303.

[Wil00] Ryan Williams, *Space-efficient reversible simulations*, Tech. report, Cornell University, 2000, Available at `http://web.stanford.edu/~rrwill/spacesim9_22.pdf`.

[Wil08] ______, *Time-space tradeoffs for counting NP solutions modulo integers*, Computational Complexity **17** (2008), no. 2, 179–219, Preliminary version in *CCC '07*.

[Woo97] David R. Wood, *An algorithm for finding a maximum clique in a graph*, Operations Research Letters **21** (1997), no. 5, 211–217.

[Yao79] Andrew Yao, *Some complexity questions related to distributive computing (preliminary report)*, Proceedings of the 11th Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '79, ACM, 1979, pp. 209–213.

[Zuc07] David Zuckerman, *Linear degree extractors and the inapproximability of max clique and chromatic number*, Theory of Computing **3** (2007), no. 6, 103–128, Preliminary version in *STOC '06*.