

LOWER BOUNDS FOR READ-ONCE AND TROPICAL FORMULAS

By

ANUJ TAWARI

MATH10201205003

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

In partial fulfillment of requirements

for the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE



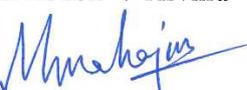
February, 2019

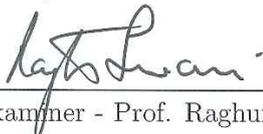
Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by ANUJ TAWARI entitled "LOWER BOUNDS FOR READ-ONCE AND TROPICAL FORMULAS" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.


Chairman - Prof. V Arvind Date: 1-2-2019


Guide/Convenor - Prof. Meena Mahajan Date: 1-2-2019


Examiner - Prof. Raghunath Tewari Date: 1/2/2019


Member 1 - Prof. Venkatesh Raman Date: 1/2/2019


Member 2 - Prof. Vikram Sharma Date: 01/02/2019

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date: 1 Feb 2019

Place: Chennai



Prof. Meena Mahajan (Guide)

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

ANUJ TAWARI

DECLARATION

I hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

ANUJ TAWARI

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Journal

1. **Sums of read-once formulas: How many summands are necessary?**

with Meena Mahajan.

In Theoretical Computer Science, 708:34-45, 2018.

2. **Shortest path length with bounded-alternation (min, +) formulas**

with Meena Mahajan and Prajakta Nimbhorkar.

In International Journal of Advances in Engineering Sciences and Applied Mathematics, <https://doi.org/10.1007/s12572-018-0229-6>, 2018.

Conferences

1. **Computing the maximum using (min, +) formulas**

with Meena Mahajan and Prajakta Nimbhorkar.

Proceedings of 42nd International Symposium on Mathematical Foundations of Computer Science MFCS 2017 (Aalborg, Denmark, 21-25 August).

2. **Sums of read-once formulas: How many summands suffice?**

with Meena Mahajan.

Proceedings of 11th Computer Science Symposium in Russia CSR 2016 (St. Petersburg, Russia, 9-13 June).

Others

1. **Computing the maximum using (min, +) formulas**

with Meena Mahajan and Prajakta Nimbhorkar.

in Electronic Colloquium on Computational Complexity (ECCC TR18-020)

DEDICATIONS

To my parents.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Meena Mahajan for her invaluable guidance during the course of this thesis. Besides her expertise, her patience, clarity of thoughts and systematic way of working have greatly influenced me. I would also like to thank her for all the support and freedom that she gave me. I also thank Prajakta Nimbhorkar for many helpful discussions.

I also thank all the other faculty members at IMSc for their wonderful teaching, guidance and support.

I would like to thank all my friends at IMSc for making my stay at IMSc a very fun-filled and memorable one. Many thanks to my office-mates over the years: Roohani, Pratik, Swaroop, Anantha and Garima. Office was a lot of fun in your company.

Special thanks to Swaroop, Ramnathan, Roohani, Shraddha, Lawqueen and Sanjukta for accompanying me to countless food trips. Thank you all for your help and support as well.

Finally, but most importantly, I would like to thank my parents for their support and encouragement. Thank you for supporting all my decisions. Without their support, taking up a research career would have been impossible.

Contents

Synopsis	i
List of Figures	iii
1 Introduction	1
1.1 Arithmetic circuits	2
1.2 Our main results	7
1.3 Organisation of thesis	11
2 Sums of read-once formulas: How many summands are necessary?	13
2.1 Highlights	13
2.2 Preliminaries	14
2.3 Background	16
2.4 Upper bounds	20
2.5 Existence of hard polynomials	22
2.6 Some useful operators	24
2.7 A proper separation in the $\sum^k \cdot ROP$ hierarchy	27

2.8	A family of 4-variate multilinear polynomials not in $\Sigma^2 \cdot \text{ROP}$	32
2.9	Discussion	41
3	Computation over semirings	43
4	Computing max using $(\min, +)$ formulas	45
4.1	Highlights	45
4.2	Introduction	46
4.2.1	Background	46
4.2.2	Motivation	48
4.2.3	Our results and techniques	49
4.3	Transformations and Upper bounds	51
4.4	Graph entropy	54
4.5	Computing max over \mathbb{N}	55
4.6	Upper bounds	56
4.7	The main lower bound	57
4.8	The Monus operation	64
4.9	Discussion	66
5	Computing shortest paths via bounded depth $(\min, +)$ formulas	67
5.1	Highlights	67
5.2	Introduction	68
5.2.1	The Shortest Path problem	68

5.2.2 Motivation	69
5.2.3 Known upper bounds	71
5.2.4 Lower bounds implied from known work	72
5.3 New Lower Bounds	74
5.4 Conclusion	81
6 Conclusion	83
Bibliography	85

Synopsis

One of the major aims of theoretical computer science is to understand what is the most efficient way to perform a given task with limited computational resources. In this thesis, some absolutely tight lower bounds are shown for certain restricted models of computation. More specifically, this thesis studies the questions of proving tight lower bounds for sums of read-once formulas, and of proving tight lower bounds for tropical formulas.

An arithmetic read-once formula (ROF) is a formula (circuit of fan-out 1) over $+$, \times where each variable labels at most one leaf. Every multilinear polynomial can be expressed as the sum of ROFs. We prove, for certain multilinear polynomials, a tight lower bound on the number of summands in such an expression.

We then proceed to study computation by tropical formulas or formulas over $(\min, +)$. Many dynamic programming algorithms can be modeled using $(\min, +)$ formulas. We consider the computation of $\max(x_1, x_2, \dots, x_n)$ over \mathbb{N} as a difference of $(\min, +)$ formulas, and show that size $n + n \log n$ is sufficient and necessary. Our proof also shows that any $(\min, +)$ formula computing the minimum of all sums of $n - 1$ out of n variables must have $n \log n$ leaves; this too is tight. Our proofs use a complexity measure for $(\min, +)$ functions based on minterm-like behaviour and on the entropy of an associated graph.

Next, we consider the well-studied shortest paths problem SHORTEST-PATH: Given a graph on vertex set $[n] = \{1, 2, \dots, n\}$ with an assignment of non-negative

integer weights to its edges, we want to find a $(\min, +)$ formula which computes the weight of the shortest path from $s = 1$ to $t = n$. We study bounded-depth $(\min, +)$ formulas solving the shortest paths problem. For depth $2d$ with $d \geq 2$, we obtain lower bounds parameterized by certain fan-in restrictions on $+$ gates except those at the bottom level. For the special case of depth four, in two regimes of the parameter, the bounds are tight.

List of Figures

1.1 Arithmetic circuit computing $f(x_1, x_2, x_3, x_4) = x_1x_2x_3 + x_1x_2x_4 +$ $2x_1x_2 + 2x_3x_4 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4$	4
2.1 Normal form for ROFs	15
2.2 Not a multiplicative ROF	25
4.1 $(\min, +)$ formula	52

Chapter 1

Introduction

The goal of theoretical computer science, and in particular, complexity theory, is to understand the power of efficient computation. That is, it asks what is the most efficient way to perform a task with limited computational resources, say, time, space or randomness. To answer this question we need to (1) find a way to perform the given task and (2) show that any other way, no matter how clever it may be, cannot do any better. In other words, we need to give upper and lower bounds on the complexity of the given task. In this thesis, we will mainly be interested in proving lower bounds.

When proving lower bounds, we need to argue about all possible ways of performing the given task. Such arguments are hard to find: we do not seem to know the techniques needed to prove strong lower bounds. Due to this lack of progress, research has focused on proving lower bounds for certain restricted models of computation, in the hope that such results may give us some insight into the general case. The purpose of this thesis is to point out some interesting directions to proving such lower bounds.

Specifically, we study the questions of proving lower bounds for the model of sums of read-once formulas, lower bounds for difference of tropical formulas computing the

maximum function, and lower bounds for bounded depth tropical formulas solving the shortest paths problem. Below, we introduce these problems in greater detail.

1.1 Arithmetic circuits

A large class of problems can be expressed as the task of computing some specific polynomials: Such problems include matrix multiplication, computing the determinant, permanent, Fast Fourier transform (FFT) as well as many other linear algebra problems. Several algorithms have been designed for these problems but we still do not know whether the currently best known algorithm is indeed the best one.

One of the most natural ways to capture many of such algorithms is via an arithmetic circuit. In this model, the circuit is fed as input some input variables and constants from the underlying field. The circuit is allowed to add and multiply two previously computed polynomials. The two main complexity measures associated with such circuits are its size and depth. The size corresponds to the number of processors running in parallel to perform the given computational task while the depth corresponds to the parallel time taken. We will mostly be interested in a special type of circuits called *formulas*, which are circuits whose underlying computation graph is a tree.

We now fix some notation. Unless otherwise specified, we work over a fixed, arbitrary algebraic structure $\mathcal{S} = (S, +, \times, 0, 1)$ where S is a set of elements, $+$ and \times are commutative binary operations acting on S and 0 and 1 are members of S . 0 is the identity for $+$ and 1 is the identity for \times : $a + 0 = a$ and $a \times 1 = a$ holds for all $a \in S$. Our main algebraic structure of interest is that of *fields* which satisfy the following additional properties:

1. $(S, +)$ and $(S \setminus 0, \times)$ are abelian groups.

2. \times distributes over $+$: $a \times (b + c) = (a \times b) + (a \times c)$ for all $a, b, c \in S$.

Specific examples of fields include the set of all rational numbers \mathbb{Q} , the set of all real numbers \mathbb{R} , and the set of all complex numbers \mathbb{C} .

Sometimes, we will be interested in computations over rings where $(S \setminus \{0\}, \times)$ may not be an abelian group, or even semirings where even $(S, +)$ need not be a group.

A special case is that of the boolean semiring, $\mathbf{B} = (\{0, 1\}, \wedge, \vee, 0, 1)$.

Now we formally define arithmetic circuits.

Definition 1. *An arithmetic circuit \mathcal{C} over the algebraic structure \mathcal{S} and set of variables X is a directed acyclic graph $G = (V, E)$. We use u, v, w to denote vertices in G and uv to denote a directed edge in E . The role of any vertex falls in one of the following cases:*

- *If v has in-degree zero, v is called an input of the arithmetic circuit and is labeled with one of the variables from X or an element from \mathcal{S} .*
- *Otherwise v is labeled with either $+$ or \times .*
- *If v has out-degree zero, then v is called an output of the arithmetic circuit.*

Given an arithmetic circuit \mathcal{C} , let the polynomial computed by \mathcal{C} at the vertex $v \in V$ be denoted by \mathcal{C}_v . We define \mathcal{C}_v inductively as follows:

1. *If v is an input, then \mathcal{C}_v is the label of v .*
2. *If v is labeled with a $+$ gate, then,*

$$\mathcal{C}_v = \sum_{u:uv \in E} \mathcal{C}_u$$

3. If v is labeled with a \times gate, then,

$$C_v = \prod_{u:uv \in E} C_u$$

An arithmetic circuit is said to be a formula if the underlying undirected graph is a tree.

Unless otherwise mentioned, we will be working with fan-in 2 circuits: each of the internal nodes has at most two children.

The size of the circuit \mathcal{C} , denoted by $\text{size}(\mathcal{C})$, is the total number of vertices in the underlying graph of \mathcal{C} . The depth of the circuit \mathcal{C} , denoted by $\text{depth}(\mathcal{C})$, is the length of the longest directed path in \mathcal{C} .

Below is an example of an arithmetic circuit which computes the polynomial $f(x_1, x_2, x_3, x_4) = x_1x_2x_3 + x_1x_2x_4 + 2x_1x_2 + 2x_3x_4 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4$:

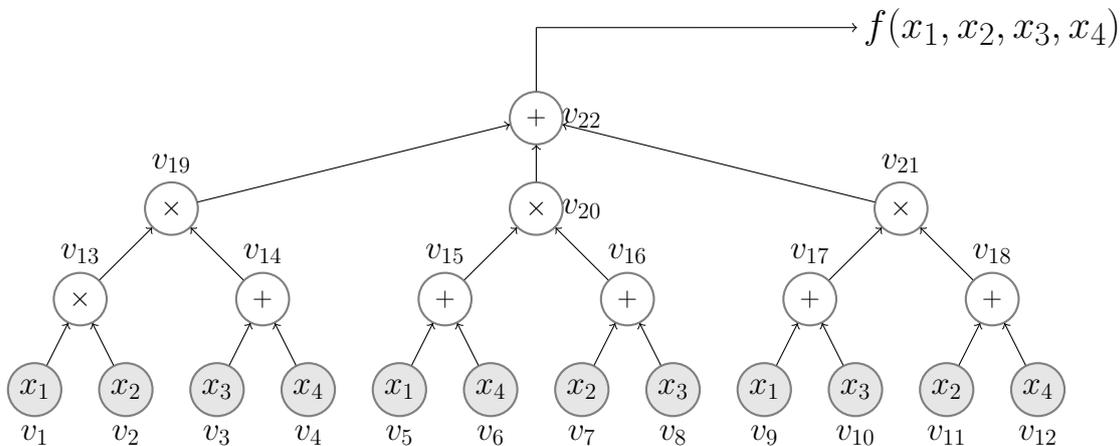


Figure 1.1: Arithmetic circuit computing $f(x_1, x_2, x_3, x_4) = x_1x_2x_3 + x_1x_2x_4 + 2x_1x_2 + 2x_3x_4 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4$

It is easy to see that in the above figure, the arithmetic circuit considered is a formula with size 22 and depth 3.

Research in arithmetic complexity theory is mainly focused on proving explicit

circuit lower bounds. Below, we explain this problem in detail.

Explicit arithmetic circuit lower bounds

The task of proving explicit arithmetic circuit lower bounds is the following problem: Come up with an explicit family of polynomials $F = \{f_n \mid n \in \mathbb{N}\}$ which cannot be computed by any family of circuits of *small* size, say polynomial, or quasi-polynomial in n . This problem is easy if we allow polynomials of arbitrary degree. For instance, consider the polynomial $f = x^{2^d}$. It is easy to verify that any circuit computing f must have at least d gates. Hence we will be interested in polynomials with reasonable bounds on their degree and the size of the smallest circuit computing it. This is captured by the notion of p -bounded polynomials. The following definition is from [44].

Definition 2. *A family of polynomials $\{f_n\}$ is p -bounded if there exists some polynomial $t : \mathbb{N} \rightarrow \mathbb{N}$ such that both the number of variables in f_n and the degree of f_n are bounded by $t(n)$. Also, there is an arithmetic circuit of size at most $t(n)$ computing f_n .*

This class of polynomials is more popularly known as the class VP .

Example 3. *An important family of p -bounded polynomials is the family of determinants, DET_n , defined below:*

$$DET_n(X) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i \in [n]} x_{i, \sigma_i}$$

Here $X = [x_{ij}]$ is a $n \times n$ matrix, S_n is the set of permutations of n elements and $\text{sgn}(\sigma)$ corresponds to the sign of the permutation σ .

We can now formally state the question of proving *explicit* circuit lower bounds: We wish to find an explicit family of polynomials $F = \{f_n(x_1, x_2, \dots, x_{t(n)}) \mid n \in \mathbb{N}\}$

where the number of variables and degree in f_n are bounded by some polynomial in n but any arithmetic circuit computing f_n requires size super-polynomial in n . By an explicit family, we mean the following: there is a deterministic Turing machine which when given an integer n and a monomial $m = x_1^{c_1} \cdot x_2^{c_2} \cdots x_{t(n)}^{c_{t(n)}}$ finds the coefficient of m in time polynomial in n .

We are very far from solving this question: The best explicit lower bound known is only $\Omega(n \log n)$ due to [47] and [4]. It might be interesting to note that the above $\Omega(n \log n)$ lower bound is absolutely tight. This raises the following natural question:

Question 1. *Can one prove a tight lower bound, better than $\Omega(n \log n)$, for some n -variate explicit family of polynomials?*

Even for restricted circuits, although strong (superpolynomial, or even exponential) lower bounds are known, not many are known to be tight. Below, we survey lower bound results for circuits with two types of restrictions: (1) number of times an input variable is read and (2) nature of polynomials computed at individual nodes (for instance, multilinear). First, we define *multilinear* polynomials and circuits.

Definition 4. *A polynomial is said to be multilinear if the individual degree of each variable appearing in it is at most one. For instance, $x_1x_2 + 2x_1x_3 + 5x_2x_4$. Many well-studied polynomials, including the determinant and permanent are multilinear.*

Definition 5. *A circuit (or formula) is said to be multilinear if at every individual gate of the circuit, the polynomial computed is multilinear.*

For the model of multilinear formulas, superpolynomial lower bounds are already known [39]. However, the best known formulas for computing those polynomials (including the permanent and determinant) have exponential size, leaving a big gap between the upper and lower bounds.

1.2 Our main results

A useful model for computing multilinear polynomials is that of sum of *read-once* formulas (ROFs). A formula is said to be read-once if every variable is the label of at most one leaf. Any polynomial that depends on all its variables must read every variable at least once. Therefore, this model computes some of the simplest polynomials. Despite being a very simple and restrictive model of computation, read-once formulas have attracted a lot of attention, both in the boolean as well as the arithmetic world [6,7,17,42,43].

We study the model of sums of read-once formulas in the arithmetic world. Exponential lower bounds on the number of summands are known for this model. For instance, it was shown in [9] that a certain polynomial, defined by Raz and Yehudayoff [40], when expressed as a sum of ROFs, requires $2^{\Omega(n^{1/3}/\log n)}$ summands, while 2^n summands is anyway sufficient. We show that any random multilinear polynomial when expressed as a sum of ROFs requires exponentially many summands. Our lower bound is better than the one proved in [9], though it is *not* for an explicit polynomial.

Result 1. *Fix any field F . There exists a family of multilinear polynomials $(f_n)_{n>0}$ with each $f_n \in F[x_1, \dots, x_n]$ such that any sums-of-ROFs representation for f_n requires $\Omega\left(\frac{2^n}{n^2}\right)$ many summands.*

As noted above, a significant gap exists between the lower bound and upper bound for the model of sums of ROPs. On the other hand, we prove an absolutely tight lower bound for a specific polynomial. Our *target* polynomial is a member of the well-studied family of elementary symmetric polynomials.

Result 2. *For any $n \geq 1$, the n -variate degree $n - 1$ elementary symmetric polynomial S_n^{n-1} cannot be expressed as the sum of fewer than $\lceil n/2 \rceil$ ROPs but it can be written as the sum of $\lceil n/2 \rceil$ ROPs.*

This result implies a strict hierarchy among sums-of-k-ROFs ($\Sigma^k \cdot \text{ROF}$, in short). In other words, for any k , $\Sigma^{k+1} \cdot \text{ROF}$ is a strictly stronger model than $\Sigma^k \cdot \text{ROF}$. This result separates $\Sigma^3 \cdot \text{ROP}$ from $\Sigma^2 \cdot \text{ROP}$ via the polynomials S_5^4 and S_6^5 . Our next main result shows that $\Sigma^3 \cdot \text{ROP}$ is also separated from $\Sigma^2 \cdot \text{ROP}$ by a 4-variate multilinear polynomial. Since every trivariate multilinear polynomial is expressible as the sum of two ROP's, the number of variables cannot be reduced further.

Result 3. *There is an explicit 4-variate multilinear polynomial f which cannot be written as the sum of 2 ROPs over \mathbb{R} .*

We next consider the model of tropical formulas. A tropical formula, or more specifically, a $(\min, +)$ formula, is a formula (tree) in which the leaves are labeled by variables or constants. The internal nodes are gates labeled by either \min or $+$. A \min gate computes the minimum value among its inputs while a $+$ gate simply adds the values computed by its inputs. Such formulas can compute any function expressible as the minimum over several linear polynomials with non-negative integer coefficients.

$(\min, +)$ circuits share an intimate connection with dynamic programming (DP, in short) algorithms. We focus on the shortest path problem, which we denoted by `SHORTEST-PATH`: Let G be an edge-weighted graph with every edge given a non-negative integral weight and two special vertices s and t . The goal is to find the weight of the shortest path from s to t . Note that the weight of a path is the sum of the weight of its edges. The classical dynamic programming algorithm for this problem due to Bellman and Ford [5,15] gives a $(\min, +)$ circuit of $O(n^3)$ size and depth $\Theta(n)$. Whether $\Omega(n^3)$ is necessary is still open. However, the Bellman-Ford algorithm produces skew circuits, and for skew circuits, this bound is shown in [27] to be optimal. A divide-and-conquer approach gives a bounded fan-in circuit of $\text{poly}(n)$ ($O(n^4)$) size and depth $\Theta(\log^2 n)$.

Another interesting problem for us is the graph reachability problem, which we

will denote by REACH : Given a directed graph G and two special vertices s and t , decide whether t can be reached from s . It is known that over the Boolean semiring, any bounded fan-in monotone (\vee, \wedge) circuit for REACH must have depth $\Omega(\log^2 n)$ [29]. Using a natural mapping from $(\min, +)$ semiring to the boolean semiring, this result also implies that any bounded fan-in $(\min, +)$ circuit for SHORTEST-PATH must have $\Omega(\log^2 n)$ depth, no matter what size. The divide-and-conquer approach shows that this depth lower bound is tight.

Many DP algorithms correspond to $(\min, +)$ circuits. Notable examples include the Bellman-Ford-Moore (BFM) algorithm for the single-source-shortest-path problem (SSSP) [5,15,37], the Floyd-Warshall (FW) algorithm for the All-Pairs-Shortest-Path (APSP) problem [13,49], and the Held-Karp (HK) algorithm for the Travelling Salesman Problem (TSP) [19]. All these algorithms are just recursively constructed $(\min, +)$ circuits. For example, both the BFM and the FW algorithms give $O(n^3)$ sized $(\min, +)$ circuits while the HK algorithm gives a $O(n^2 \cdot 2^n)$ sized $(\min, +)$ circuit.

We consider the following problem: Given n input variables x_1, x_2, \dots, x_n with values ranging over natural numbers, compute the maximum value taken by them. We first observe that for $n \geq 2$, no $(\min, +)$ formula over \mathbb{N} can compute $\max(x_1, x_2, \dots, x_n)$. Hence to compute the maximum, we must strengthen this model. An obvious way is by allowing minus gates as well. It turns out that just a single minus gate, at the top, suffices. The maximum can be computed by difference of two $(\min, +)$ formulas with total size $O(n \log n)$. We also give a matching lower bound.

Result 4. *Any difference of $(\min, +)$ formulas which computes $\max(x_1, x_2, \dots, x_n)$ must have size $\Omega(n \log n)$.*

Note that this result is tight even on considering constants, and not just asymptotically tight.

We now move on to results on the complexity of the shortest paths problem. First, we briefly describe some lower bounds which follow easily from known results.

We consider the alternation depth of $(\min, +)$ circuits. This corresponds to allowing unbounded fan-in in some cases. In this setting, exponential lower bounds are easy to prove. One way to do so is to use the reduction (via projections) from parity to REACH, and use known lower bounds for (non-monotone) circuits for parity [18]; see Proposition 58. We are looking for lower bounds better than those obtained this way.

In [10], such small-depth lower bounds are obtained for the decision version of "short distance connectivity": is there a path using at most k edges? These lower bounds can also be transferred to $(\min, +)$ circuits computing the corresponding optimization problem: Compute the weight of the shortest path which uses at most k edges. Now, we describe our results.

For depth $2d$ with $d \geq 2$, we obtain lower bounds parameterized by certain fan-in restrictions on $+$ gates except those at the bottom level. For the special case of depth 4, in two regimes of the parameter, the bounds are tight. The restrictions we study are of two types: (1) all gates have low fan-in or (2) not too many gates have large fan-in.

Let $L(F)$ denote the total number of leaves in the formula F . $L(F)$ is indicative of the size of the formula.

Result 5. *If F is a depth $2d$ formula for SHORTEST-PATH where all $+$ gates except those in the bottom level have fanin at most k , then*

$$L(F) \geq \exp \left(\Omega \left(\frac{n \log n}{k^{d-1}} \right) \right).$$

For the special case of depth 4 formulas,

1. If $k = O(1)$, then $L(F) = 2^{\Omega(n \log n)}$. This size is achievable even with a depth-2 formula and so this bound is tight
2. If $k = O(\sqrt{n})$, then $L(F) = 2^{\Omega(\sqrt{n} \log n)}$. This size is achievable with the depth-4 formula constructed by dynamic programming with all $+$ gates having fanin $O(\sqrt{n})$ and so this bound is tight.

Result 6. *For natural numbers n, r, k , let $L(n, k, r)$ denote the (leaf-)size of the smallest depth-4 formula that solves SHORTEST-PATH on n -vertex graphs, and where at most r of the $+$ gates at the second level have fan-in exceeding k .*

$$L(n, 2, r) \geq \exp \left(\Omega \left(\frac{n}{2^r} \log \frac{n}{2^r} \right) \right).$$

The above result gives a non-trivial size lower bound for depth-4 formulas when at most say, $O(\log \log n)$ of the second level $+$ gates have fanin more than 2.

The results described above have appeared or are to appear in [34–36].

1.3 Organisation of thesis

We present our results in the same order as they are described above. In Chapter 2, we consider sums of read-once formulas and show a strict hierarchy for this model. In Chapter 3, we introduce some basic definitions corresponding to the computational model used in the rest of the thesis. In Chapter 3, we consider the computation of the maximum function using $(\min, +)$ formulas. This is followed, in Chapter 5, by results on the complexity of the shortest path problem in the context of bounded depth $(\min, +)$ formulas. Finally, in Chapter 6, we conclude by stating some open questions.

Chapter 2

Sums of read-once formulas: How many summands are necessary?

2.1 Highlights

In this chapter, we present our results on lower bounds for *sums of read-once formulas*. We start by defining read-once formulas. We then define the problem of establishing a hierarchy among sums-of-ROFs and also provide our own motivation for studying this problem. We then discuss some related work and finally our results on this problem. Our main results are as follows:

- There exists a multilinear polynomial, which when expressed as a sum of read-once polynomials (ROPs, in short), requires exponentially many summands. (Theorem [15](#)).
- The hierarchy among sums of ROPs is proper. That is, for any positive integer k , $\sum^k \cdot \text{ROP}$ is a strictly stronger model than $\sum^{k-1} \cdot \text{ROP}$ (Theorem [9](#)).
- Over certain fields, the separation between $\sum^3 \cdot \text{ROP}$ and $\sum^2 \cdot \text{ROP}$ is witnessed by an explicit 4-variate multilinear polynomial. (Theorem [31](#)).

Organization of chapter

In Section [2.2](#), we introduce some basic preliminaries and definitions useful for us. In Section [2.3](#), we introduce our problem of interest. In Section [2.4](#), we present some upper bounds for expressing any multilinear polynomial as a sum of ROPs. In Section [2.5](#), we show the existence of polynomials which when expressed as a sum of ROPs, require exponentially many summands. In Section [2.6](#), we introduce some useful operators need for the proof of our main theorem. In Section [2.7](#), we establish Theorem [9](#), showing that the hierarchy of k -sums of ROPs is proper. In Section [2.8](#) we establish Theorem [10](#), showing an explicit 4-variate multilinear polynomial that is not expressible as the sum of two ROPs. Finally, we conclude in Section [2.9](#) with some further questions that are still open.

2.2 Preliminaries

Recall the definition of arithmetic formulas from Chapter 1 (Definition [1](#)).

A formula is said to be read- k if each variable appears as a leaf label at most k times.

For read-once formulas, it is more convenient to use the following “normal form” from [43](#).

Definition 6 (Read-once formulas [43](#)). *A read-once arithmetic formula (ROF) over a field F in the variables $\{x_1, x_2, \dots, x_n\}$ is a binary tree as follows. The leaves are labeled by variables and internal nodes by $\{+, \times\}$. In addition, every node is labeled by a pair of field elements $(\alpha, \beta) \in F^2$. Each input variable labels at most once leaf. The computation is performed in the following way. A leaf labeled by x_i and (α, β) computes $\alpha x_i + \beta$. If a node v is labeled by $\star \in \{+, \times\}$ and (α, β) and its children compute the polynomials f_1 and f_2 , then v computes $\alpha(f_1 \star f_2) + \beta$.*

The n -variate degree k elementary symmetric polynomial, denoted S_n^k , is defined as follows:

$$S_n^k(x_1, \dots, x_n) = \sum_{A \subseteq [n], |A|=k} \prod_{i \in A} x_i.$$

2.3 Background

Read-once formulas (ROF) are formulas (circuits of fan-out 1) in which each variable appears at most once. A formula computing a polynomial that depends on all its variables must read each variable at least once. Therefore, ROFs compute some of the simplest possible functions that depend on all of their variables. The polynomials computed by such formulas are known as read-once polynomials (ROPs). Since every variable is read at most once, ROPs are multilinear. (A polynomial is said to be multilinear if the individual degree of each variable is at most one.) But not every multilinear polynomial is a ROP. For example, $x_1x_2 + x_2x_3 + x_1x_3$ [41].

We investigate the following question: Given an n -variate multilinear polynomial, can it be expressed as a sum of at most k ROPs? It is easy to see that every bivariate multilinear polynomial is a ROP. Any tri-variate multilinear polynomial can be expressed as a sum of two ROPs. With a little thought, we can obtain a sum-of-3-ROPs expression for any 4-variate multilinear polynomial. An easy induction on n then shows that any n -variate multilinear polynomial, for $n \geq 4$, can be written as a sum of at most $3 \times 2^{n-4}$ ROPs; see Proposition [11]. Also, the sum of two multilinear monomials is a ROP, so any n -variate multilinear polynomial with M monomials can be written as the sum of $\lceil M/2 \rceil$ ROPs (Proposition [12]). We ask the following question: Does there exist a strict hierarchy among k -sums of ROPs? Formally,

Problem 1. *Consider the family of n -variate multilinear polynomials. For $1 < k \leq 3 \times 2^{n-4}$, is $\sum^k \cdot \text{ROP}$ strictly more powerful than $\sum^{k-1} \cdot \text{ROP}$? If so, what explicit polynomials witness the separations?*

We answer this affirmatively for $k \leq \lceil n/2 \rceil$. In particular, for $k = \lceil n/2 \rceil$, there exists an explicit n -variate multilinear polynomial which cannot be written as a sum of less than k ROPs but it admits a sum-of- k -ROPs representation.

Note that n -variate ROPs are computed by linear sized formulas. Thus if an n -variate polynomial p is in \sum^k -ROP, then p is computed by a formula of size $O(kn)$ where every intermediate node computes a multilinear polynomial. Since superpolynomial lower bounds are already known for the model of multilinear formulas [39], we know that for those polynomials (including the determinant and the permanent), a \sum^k -ROP expression must have k at least quasi-polynomial in n . However the best upper bound on k for these polynomials is only exponential in n , leaving a big gap between the lower and upper bound on k . A lesser but still significant gap also exists in the known exponential lower bound for sums of ROPs; in [9] it is shown that a certain polynomial, explicitly described in [40], requires $2^{\Omega(n^{1/3}/\log n)}$ ROP summands, while 2^n summands is anyway sufficient. On the other hand, our lower bound is provably tight.

A counting argument (see Proposition [14]) shows that there exists a multilinear polynomial which requires exponentially many ROPs; there are multilinear polynomials requiring $k = \Omega(2^n/n^2)$. Our general upper bound on k is $O(2^n)$, leaving a gap between the lower and upper bound. One challenge is to close this gap.

A natural question to ask is whether stronger lower bounds than the above result can be proven. In particular, to separate \sum^{k-1} -ROP from \sum^k -ROP, how many variables are needed? Our hierarchy result says that $2k - 1$ variables suffice, but there may be simpler polynomials (with fewer variables) witnessing this separation. We demonstrate another technique which improves upon the previous result for $k = 3$, showing that 4 variables suffice. In particular, we show that over the field of reals, there exists an explicit multilinear 4-variate multilinear polynomial which cannot be written as a sum of 2 ROPs. This lower bound is again tight, as there is

a sum of 3 ROPs representation for every 4-variate multilinear polynomial.

Our results and techniques

We now formally state our main results.

The first main result establishes the strict hierarchy among k -sums of ROPs.

Theorem 9. *For each $n \geq 1$, the n -variate degree $n - 1$ symmetric polynomial S_n^{n-1} cannot be written as a sum of less than $\lceil n/2 \rceil$ ROPs, but it can be written as a sum of $\lceil n/2 \rceil$ ROPs.*

The idea behind the lower bound is that if $g = S_n^{n-1}$ can be expressed as a sum of less than $\lceil n/2 \rceil$ ROPs, then one of the ROPs can be eliminated by taking partial derivative with respect to one variable and substituting another by a field constant. We then use the inductive hypothesis to arrive at a contradiction. This approach necessitates a stronger hypothesis than the statement of the theorem, and we prove this stronger statement in Lemma 26 as part of Theorem 29.

This result separates $\sum^3 \cdot \text{ROP}$ from $\sum^2 \cdot \text{ROP}$ via the polynomials S_5^4 and S_6^5 . Our second main result shows that $\sum^3 \cdot \text{ROP}$ is also separated from $\sum^2 \cdot \text{ROP}$ by a 4-variate multilinear polynomial.

Theorem 10. *There is an explicit 4-variate multilinear polynomial f which cannot be written as the sum of two ROPs over \mathbb{R} .*

The proof of this theorem mainly relies on a structural lemma (Lemma 34) for sum of two read-once formulas. In particular, we show that if f can be written as a sum of two ROPs then one of the following must be true:

1. Some 2-variate restriction is a linear polynomial.

2. There exist variables $x_i, x_j \in \text{Var}(f)$ such that the polynomials $x_i, x_j, \partial_{x_i}(f), \partial_{x_j}(f), 1$ are linearly dependent.
3. We can represent f as $f = l_1 \cdot l_2 + l_3 \cdot l_4$ where (l_1, l_2) and (l_3, l_4) are variable-disjoint linear forms.

Checking the first two conditions is easy. For the third condition we use the commutator of f , introduced in [42], to find one of the l_i 's. The knowledge of one of the l_i 's suffices to determine all the linear forms. Finally, we construct a 4-variate polynomial which does not satisfy any of the above mentioned conditions. This construction does not work over algebraically closed fields. We do not yet know how to construct an explicit 4-variate multilinear polynomial not expressible as the sum of two ROPs over such fields, or even whether such polynomials exist.

Related work

Despite their simplicity, ROFs have received a lot of attention both in the arithmetic as well as in the Boolean world [6–8, 17, 42, 43]. The most fundamental question that can be asked about polynomials is the polynomial identity testing (PIT) problem: Given an arithmetic circuit \mathcal{C} , is the polynomial computed by \mathcal{C} identically zero or not. PIT has a randomized polynomial time algorithm: Evaluate the polynomial at random points. It is not known whether PIT has a deterministic polynomial time algorithm. In [28], a connection between PIT algorithms and proving general circuit lower bounds was established. Similar results are known for some restricted classes of arithmetic circuits, for instance, constant-depth circuits [1, 12]. However, consider the case of multilinear formulas. Even though strong lower bounds are known for this model, there is no efficient deterministic PIT algorithm. (Notice that multilinear depth 3 circuits are a special case of this model.) For this reason, PIT was studied for the weaker model of sum of read-once formulas.

In [43], a deterministic PIT algorithm for the sum of a small number of ROPs was given. Interestingly, their proof uses a lower bound for a weaker model, that of 0-justified ROFs (setting some variables to zero does not kill any other variables). In particular, they show that the polynomial $\mathcal{M}_n = x_1x_2 \cdots x_n$, consisting of just a single monomial, cannot be represented as a sum of less than $n/3$ weakly justified ROPs. More recently, Kayal showed that if \mathcal{M}_n is represented as a sum of powers of low degree (at most d) polynomials, then the number of summands is at least $\exp(\Omega(n/d))$ [30]. This lower bound, along with the arguments in [43], yields a sub-exponential time PIT algorithm for multilinear polynomials. This can be further extended to arbitrary polynomials written as sum of powers of low degree polynomials, using the ideas in [14]. Our lower bound from Theorem 9 is independent of both these lower bounds (0-justified ROFs from [43], and sums of powers of low-degree polynomials from [30]) and is provably tight. An interesting question is whether it can be used to give a PIT algorithm for sums of k ROPs, when k is linear in n .

Similar to ROPs, one may also study read-restricted formulas. For any number k , RkFs are formulas that read every variable at most k times. For $k \geq 2$, RkFs need not be multilinear, and thus are strictly more powerful than ROPs. However, even when restricted to multilinear polynomials, they are more powerful; in [3], Anderson, Melkebeek and Volkovich show that there is a multilinear n -variate polynomial in R2F requiring $\Omega(n)$ summands when written as a sum of ROPs.

2.4 Upper bounds

Proposition 11. *For every n -variate multilinear polynomial f ,*
 $\text{Summands}_{\text{ROP}}(f) \leq \lceil 3 \times 2^{n-4} \rceil$.

Proof. For $n = 1, 2, 3$ this is easy to see.

For $n = 4$, let $f(X)$ be given by the expression $\sum_{S \subseteq [4]} a_S x_S$, where x_S denotes the monomial $\prod_{i \in S} x_i$. We want to express f as $f_1 + f_2 + f_3$, where each f_i is an ROP. If there are no degree 2 terms, we use the following:

$$\begin{aligned} f_1 &= a_\emptyset + a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \\ f_2 &= x_1 x_2 (a_{123} x_3 + a_{124} x_4) \\ f_3 &= x_3 x_4 (a_{134} x_1 + a_{234} x_2 + a_{1234} x_1 x_2) \end{aligned}$$

Otherwise, assume without loss of generality that $a_{13} \neq 0$. Then define

$$\begin{aligned} f_1 &= \left[\sum_{S \subseteq [2]} a_S \prod_{i \in S} x_i \right] + \left[\sum_{\emptyset \neq S \subseteq \{3,4\}} a_S \prod_{i \in S} x_i \right] \\ f_2 &= (a_{13} x_1 + a_{23} x_2 + a_{123} x_1 x_2) \cdot \left(\frac{a_{14}}{a_{13}} x_4 + x_3 + \frac{a_{134}}{a_{13}} x_3 x_4 \right) \\ f_3 &= x_2 x_4 \left[\left(a_{24} - \frac{a_{14} a_{23}}{a_{13}} \right) + x_1 \left(a_{124} - \frac{a_{14} a_{123}}{a_{13}} \right) \right. \\ &\quad \left. + x_3 \left(a_{234} - \frac{a_{134} a_{23}}{a_{13}} \right) + x_1 x_3 \left(a_{1234} - \frac{a_{134} a_{123}}{a_{13}} \right) \right] \end{aligned}$$

Since any bivariate multilinear polynomial is a ROP, each f_i is indeed an ROP.

For $n > 4$, express f as $x_n g + h$ where $g = f|_{x_n=1} - f|_{x_n=0}$ and $h = f|_{x_n=0}$, and use induction, along with the fact that g does not have variable x_n . \square

Proposition 12. *For every n -variate multilinear polynomial f with M monomials, $\text{SummandsROP}(f) \leq \lceil \frac{M}{2} \rceil$.*

Proof. For $S \subseteq [n]$, let x_S denote the multilinear monomial $\prod_{i \in S} x_i$. For any $S, T \subseteq [n]$, the polynomial $ax_S + bx_T$ equals $x_{S \cap T} (ax_{S \setminus T} + bx_{T \setminus S})$ and hence is an ROP. Pairing up monomials in any way gives the $\lceil \frac{M}{2} \rceil$ bound. \square

2.5 Existence of hard polynomials

First, we show the existence of hard polynomials over finite fields. The proof is via a combinatorial argument. The idea behind the proof is that the number of multilinear polynomials with 0 – 1 coefficients is strictly larger than the number of different possible sums of *small* number of ROFs. Then we conclude that there exists some multilinear polynomial which when expressed as a sum of ROFs requires large number of summands. To this end, we first compute the total number of multilinear polynomials with 0 – 1 coefficients.

Observation 13. *Let \mathcal{M} denote the set of multilinear polynomials in $\mathbb{F}[x_1, \dots, x_n]$ where each coefficient is either zero or one. Then $|\mathcal{M}| = 2^{2^n}$.*

Proposition 14. *Fix any finite field \mathbb{F} . There exists a family of multilinear polynomials $(f_n)_{n>0}$ with each $f_n \in \mathbb{F}[x_1, \dots, x_n]$ such that $\text{SummandsROP}(f_n) = \Omega\left(\frac{2^n}{n \log n}\right)$.*

Proof. A single ROF is a binary tree with at most n leaves, and with labels at each node. A leaf is labeled by a single x variable and a pair of field elements, and an internal node is labeled by a gate type (+ or \times) and a pair of field elements. The number of binary trees with at most n leaves is $2^{O(n)}$. If the field size is q , then the number of labelings per tree is at most $(nq^2)^n(2q^2)^n$. Hence the number of ROFs is no more than $2^{O(n \log n)}$. A $\sum^s \cdot$ ROF formula can be obtained by choosing an ROF for each of the s positions; hence there are at most $2^{O(sn \log n)}$ distinct formulas. This is less than $|\mathcal{M}|$ unless $s = \Omega\left(\frac{2^n}{n \log n}\right)$. \square

Next, we establish the existence of hard polynomials over any field, not necessarily finite.

Theorem 15. *Fix any field \mathbb{F} . There exists a family of multilinear polynomials $(f_n)_{n>0}$ with each $f_n \in \mathbb{F}[x_1, \dots, x_n]$ such that $\text{SummandsROP}(f_n) = \Omega\left(\frac{2^n}{n^2}\right)$.*

Proof. We will show that unless $s \in \Omega\left(\frac{2^n}{n^2}\right)$, the number of polynomials in \mathcal{M} computable by $\sum^s \cdot \text{ROF}$ is strictly less than $|\mathcal{M}|$.

We use the strategy from [20]; a similar strategy was also used in [44]. Using notation from [20], we call a circuit or formula with no field constants a *skeleton*. From any circuit or formula, we can obtain a skeleton by simply replacing each occurrence of a field element by a fresh variable. Our counting proceeds as follows:

Fix any $s \in \mathbb{N}$. Define the following quantities.

N_1 : the number of distinct skeletons arising from $\sum^s \cdot \text{ROF}$ formulas on n variables.

Each skeleton computes a polynomial in the variables $X \cup Z$, where $X = \{x_i \mid i \in [n]\}$ and $Z = \{z_i \mid i \in [t]\}$ for some $t \in O(ns)$.

N_2 : the number of polynomials from \mathcal{M} computable by a single skeleton on appropriate instantiation of the z variables.

Then $\sum^s \cdot \text{ROF}$ expressions can compute at most $N_1 \times N_2$ polynomials in \mathcal{M} .

First, we estimate N_1 . Note that a $\sum^s \cdot \text{ROF}$ formula has at most $3ns$ gates apart from the top $+$ gates. (We implicitly unfold an ROF gate f labeled (\circ, α, β) and with children g, h into a small sub-formula $\alpha \times (g \circ h) + \beta$, and then replace α, β by fresh z variables.) We use a generous over-estimate for N_1 , namely, the number of skeletons of circuits of size $3ns$. We have n variables in X and t variables in Z . Each node in the skeleton can be labeled in at most $n + t + 2$ ways (a variable or a gate type), and its children can be chosen in at most $(3ns)^2$ ways. Hence the number of skeletons is no more than $[(n + t + 2)(3ns)^2]^{3ns}$. Since $t = O(ns)$, we conclude that $N_1 = 2^{O(ns(\log n + \log s))}$.

Estimating N_2 is trickier because the field may not be finite, and thus a single skeleton can give rise to infinitely many polynomials. However, we are interested only in polynomials from the finite set \mathcal{M} . This can be bounded using a dimension

argument as used in [20]. In particular, we use the following result proved in [20]:

Lemma 16 (Lemma 3.5 in [20]). *Let F be a field. Let $F : F^n \rightarrow F^m$ be a polynomial map of degree $d > 0$, that is, $F = (F_1, \dots, F_m)$, each F_i is a n -variate polynomial of degree d . Then $|F(F^n) \cap \{0, 1\}^m| \leq (2d)^n$*

We have a given fixed skeleton corresponding to some $\sum^s \cdot \text{ROF}$. It computes some polynomial $\psi(X, Z)$, with $|X| = n$, $|Z| = t$, $t = O(ns)$. By the nature of ROF, ψ is multilinear, and hence can be written in the form

$$\psi(X, Z) = \sum_{S \subseteq [n]} \left(c_S(Z) \prod_{i \in S} x_i \right)$$

where each coefficient $c_S(Z)$ is a multilinear polynomial. These 2^n coefficient polynomials form our polynomial map $F : F^t \rightarrow F^{2^n}$. Since each coefficient polynomial is multilinear, it has total degree at most t . Hence, from Lemma [16], we conclude that at most $(2t)^t$ 0-1 tuples are produced by this map. Thus the given skeleton can compute at most $(2t)^t$ polynomials from \mathcal{M} . Since $t = O(ns)$, we obtain $N_2 = 2^{O(ns(\log n + \log s))}$.

Now that we have estimated N_1 and N_2 , we can bound SummandsROP. Assume that for all polynomials $f \in \mathcal{M}$, $\text{SummandsROP}(f) \leq s$. Then $\sum^s \cdot \text{ROF}$ contains all of \mathcal{M} . Hence $N_1 \times N_2 \geq |\mathcal{M}|$, implying $s \geq \Omega\left(\frac{2^n}{n^2}\right)$. \square

2.6 Some useful operators

The partial derivative of a polynomial is defined naturally over continuous domains. The definition can be extended in more than one way over finite fields. However, for multilinear polynomials, these definitions coincide. We consider only multilinear polynomials, and the following formulation is most useful for us: The partial derivative of a polynomial $p \in F[x_1, x_2, \dots, x_n]$ with respect to a variable x_i , for $i \in [n]$, is

given by $\partial_{x_i}(p) \triangleq p|_{x_i=1} - p|_{x_i=0}$. For multilinear polynomials, the sum, product, and chain rules continue to hold.

Fact 17 (Useful Fact about ROPs [43]). *The partial derivatives of ROPs are also ROPs.*

Proposition 18 (3-variate ROPs). *Let $f \in \mathbb{F}[x_1, x_2, x_3]$ be a 3-variate ROP. Then there exists $i \in [3]$ and $a \in \mathbb{F}$ such that $\deg(f|_{x_i=a}) \leq 1$.*

Proof. Assume without loss of generality that $f = f_1(x_1) \star f_2(x_2, x_3) + c$ where $\star \in \{+, \times\}$ and $c \in \mathbb{F}$. Here f_1 is linear while f_2 is multilinear. If $\star = +$, then for all $a \in \mathbb{F}$, $\deg(f|_{x_2=a}) \leq 1$. If $\star = \times$, $\deg(f|_{f_1=0}) \leq 1$. \square

We will also be dealing with a special case of ROFs called multiplicative ROFs defined below:

Definition 19 (Multiplicative Read-once formulas). *A ROF is said to be a multiplicative ROF if it does not contain any addition gates. We say that f is a multiplicative ROP if it can be computed by a multiplicative ROF.*

For instance, in Figure 2.1, the formula is a multiplicative ROF because the equivalent normal form has no $+$ gates. Below, we give an example of a ROF which is not multiplicative.

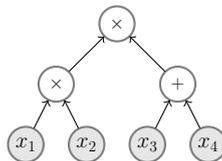


Figure 2.2: Not a multiplicative ROF

Fact 20 ([43] (Lemma 3.10)). *A ROP p is a multiplicative ROP if and only if for any two variables $x_i, x_j \in \text{Var}(p)$, $\partial_{x_i}\partial_{x_j}(p) \neq 0$.*

Multiplicative ROPs have the following useful property, observed in [43]. (See Lemma 3.13 in [43]. For completeness, and since we refer to the proof later, we include a proof sketch here.)

Lemma 21 ([43]). *Let g be a multiplicative ROP with $|\text{Var}(g)| \geq 2$. For every $x_i \in \text{Var}(g)$, there exists $x_j \in \text{Var}(g) \setminus \{x_i\}$ and $\gamma \in \mathbb{F}$ such that $\partial_{x_j}(g) |_{x_i=\gamma} = 0$.*

Proof. Let φ be a multiplicative ROF computing g . Pick any $x_i \in \text{Var}(g)$. As $|\text{Var}(\varphi)| = |\text{Var}(g)| \geq 2$, φ has at least one gate. Let v be the unique neighbour (parent) of the leaf labeled by x_i , and let w be the other child of v . We denote by $P_v(\bar{x})$ and $P_w(\bar{x})$ the ROPs computed by v and w . Since v is a \times gate and we use the normal form from Definition 6, P_v is of the form $(\alpha x_i + \beta) \times P_w$ for some $\alpha \neq 0$. Replacing the output from v by a new variable y , we obtain from φ another multiplicative ROF ψ in the variables $\{y\} \cup \text{Var}(g) \setminus \text{Var}(P_v)$. Let ψ compute the polynomial Q ; then $g = Q |_{y=P_v}$.

Note that the sets $\text{Var}(Q), \{x_i\}, \text{Var}(P_w)$ are non-empty and disjoint, and form a partition of $\{y, x_1, \dots, x_n\}$.

By the chain rule, for every variable $x_j \in \text{Var}(P_w)$ we have:

$$\partial_{x_j}(g) = \partial_y(Q) \cdot \partial_{x_j}(P_v) = \partial_y(Q) \cdot (\alpha x_i + \beta) \cdot \partial_{x_j}(P_w)$$

It follows that for $\gamma = -\beta/\alpha$, $\partial_{x_j}(g) |_{x_i=\gamma} = 0$. □

Along with partial derivatives, another operator that we will find useful is the commutator of a polynomial. The commutator of a polynomial has previously been used for polynomial factorization and in reconstruction algorithms for read-once formulas, see [42].

Definition 22 (Commutator [42]). *Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a multilinear polynomial and let $i, j \in [n]$. The commutator between x_i and x_j , denoted $\Delta_{ij}P$, is*

defined as follows.

$$\Delta_{ij}P = (P|_{x_i=0, x_j=0}) \cdot (P|_{x_i=1, x_j=1}) - (P|_{x_i=0, x_j=1}) \cdot (P|_{x_i=1, x_j=0})$$

The following property of the commutator will be useful to us.

Lemma 23. *Let $f = l_1(x_1, x_2) \cdot l_2(x_3, x_4) + l_3(x_1, x_3) \cdot l_4(x_2, x_4)$ where the l_i 's are linear polynomials. Then l_2 divides $\Delta_{12}(f)$.*

Proof. First, we show that $\Delta_{12}(l_3 \cdot l_4) = 0$. Assume $l_3 = Cx_1 + m$ and $l_4 = Dx_2 + n$ where $C, D \in \mathbb{F}$ and m, n are linear polynomials in x_3, x_4 respectively. By definition, $\Delta_{12}(l_3 \cdot l_4) = mn(C + m)(D + n) - m(D + n)(C + m)n = 0$.

Now we write $\Delta_{12}f$ explicitly. Let $l_1 = ax_1 + bx_2 + c$. By definition,

$$\begin{aligned} \Delta_{12}f &= \Delta_{12}(l_1l_2 + l_3l_4) \\ &= (cl_2 + mn)((a + b + c)l_2 + (C + m)(D + n)) - \\ &\quad ((b + c)l_2 + m(D + n)) \cdot ((a + c)l_2 + n(C + m)) \\ &= l_2^2(c(a + b + c) - (a + c)(b + c)) \\ &\quad + l_2(c(C + m)(D + n) + mn(a + b + c) - n(b + c)(C + m) - m(a + c)(D + n)) \end{aligned}$$

It follows that l_2 divides $\Delta_{12}f$. □

2.7 A proper separation in the $\sum^k \cdot ROP$ hierarchy

This section is devoted to proving Theorem [9](#).

We prove the lower bound for S_n^{n-1} by induction. This necessitates a stronger induction hypothesis, so we will actually prove the lower bound for a larger class of polynomials. For any $\alpha, \beta \in \mathbb{F}$, we define the polynomial $\mathcal{M}_n^{\alpha, \beta} = \alpha S_n^\alpha + \beta S_n^{\alpha-1}$.

Proposition 24. $\mathcal{M}_n^{\alpha,\beta}$ has the following recursive structure:

$$\begin{aligned} (\mathcal{M}_n^{\alpha,\beta})|_{x_n=\gamma} &= \mathcal{M}_{n-1}^{\alpha\gamma+\beta,\beta\gamma} . \\ \partial_{x_n}(\mathcal{M}_n^{\alpha,\beta}) &= \mathcal{M}_{n-1}^{\alpha,\beta} . \end{aligned}$$

Proof.

$$\begin{aligned} \mathcal{M}_n^{\alpha,\beta} &= \alpha S_n^n + \beta S_n^{n-1} = \alpha \left(\prod_{j \in [n]} x_j \right) + \beta \left(\sum_{i \in [n]} \left[\prod_{j \in [n] \setminus \{i\}} x_j \right] \right) \\ &= \alpha x_n \left(\prod_{j \in [n-1]} x_j \right) + \beta \left(\sum_{i \in [n-1]} x_n \left[\prod_{j \in [n-1] \setminus \{i\}} x_j \right] \right) + \beta \left(\prod_{j \in [n-1]} x_j \right) \\ &= \alpha x_n S_{n-1}^{n-1} + \beta x_n S_{n-1}^{n-2} + \beta S_{n-1}^{n-1}. \end{aligned}$$

$$\text{Hence } (\mathcal{M}_n^{\alpha,\beta})|_{x_n=\gamma} = (\alpha\gamma + \beta) S_{n-1}^{n-1} + \beta\gamma S_{n-1}^{n-2} = \mathcal{M}_{n-1}^{\alpha\gamma+\beta,\beta\gamma}$$

$$\text{and } \partial_{x_n}(\mathcal{M}_n^{\alpha,\beta}) = \alpha S_{n-1}^{n-1} + \beta S_{n-1}^{n-2} = \mathcal{M}_{n-1}^{\alpha,\beta}$$

□

We show below that each $\mathcal{M}_n^{\alpha,\beta}$ is expressible as the sum of $\lceil n/2 \rceil$ ROPs (Lemma [25](#)); however, for any non-zero $\beta \in \mathbb{F}$, $\mathcal{M}_n^{\alpha,\beta}$ cannot be written as the sum of fewer than $\lceil n/2 \rceil$ ROPs (Lemma [26](#)). At $\alpha = 0$, $\beta = 1$, we get S_n^{n-1} , the simplest such polynomials, establishing Theorem [9](#).

First we establish the upper bound.

Lemma 25. For any field \mathbb{F} and $\alpha, \beta \in \mathbb{F}$, the polynomial $f = \alpha S_n^n + \beta S_n^{n-1}$ can be written as a sum of at most $\lceil n/2 \rceil$ ROPs.

Proof. For n odd, this follows immediately from Proposition [12](#).

If n is even, say $n = 2k$, then define the following polynomials:

$$\text{for } i \in [k-1], \quad f_i = (x_{2i-1} + x_{2i}) \cdot \left(\prod_{\substack{k \in [n] \\ k \neq 2i, 2i-1}} x_k \right)$$

$$f_k = (\beta x_{2k-1} + \beta x_{2k} + \alpha x_{2k-1} x_{2k}) \cdot \left(\prod_{\substack{m \in [n] \\ k \neq 2k, 2k-1}} x_m \right).$$

Then we have $f = \beta(f_1 + f_2 + \dots + f_{k-1}) + f_k$.

Note that each f_i is an ROP; for $i < k$ this is immediate, and for $i = k$, the factor involving x_{2k-1} and x_{2k} is bivariate multilinear and hence an ROP. Thus we have a representation of f as a sum of $k = \lceil n/2 \rceil$ ROPs. \square

The following lemma shows that the above upper bound is indeed optimal.

Lemma 26. *Let F be a field. For every $\alpha \in F$ and $\beta \in F \setminus \{0\}$, the polynomial $\mathcal{M}_n^{\alpha, \beta} = \alpha S_n^n + \beta S_n^{n-1}$ cannot be written as a sum of $k < n/2$ ROPs.*

Proof. The proof is by induction on n . The cases $n = 1, 2$ are easy to see. We now assume that $k \geq 1$ and $n > 2k$. Assume to the contrary that there are ROPs f_1, f_2, \dots, f_k over $F[x_1, x_2, \dots, x_n]$ such that $f \triangleq \sum_{m \in [k]} f_m = \mathcal{M}_n^{\alpha, \beta}$. The main steps in the proof are as follows:

1. Show using the inductive hypothesis that for all $m \in [k]$ and $a, b \in [n]$, $\partial_{x_a} \partial_{x_b} (f_m) \neq 0$.
2. Conclude that for all $m \in [k]$, f_m must be a multiplicative ROP. That is, the ROF computing f_m does not contain any addition gate.
3. Use the multiplicative property of f_k to show that f_k can be eliminated by taking partial derivative with respect to one variable and substituting another by a field constant. If this constant is non-zero, we contradict the inductive hypothesis.

4. Otherwise, use the sum of (multiplicative) ROPs representation of $\mathcal{M}_n^{\alpha,\beta}$ to show that the degree of f can be made at most $(n - 2)$ by setting one of the variables to zero. This contradicts our choice of f since $\beta \neq 0$.

We now proceed with the proof.

Proposition 27. *For all $m \in [k]$ and $a, b \in [n]$, $\partial_{x_a} \partial_{x_b}(f_m) \neq 0$.*

Proof. Suppose to the contrary that $\partial_{x_a} \partial_{x_b}(f_m) = 0$. Assume without loss of generality that $a = n$, $b = n - 1$, $m = k$, so $\partial_{x_n} \partial_{x_{n-1}}(f_k) = 0$. Then,

$$\begin{aligned} \mathcal{M}_n^{\alpha,\beta} = f &= \sum_{m=0}^k f_m && \text{(by assumption)} \\ \partial_{x_n} \partial_{x_{n-1}}(\mathcal{M}_n^{\alpha,\beta}) &= \sum_{m=0}^k \partial_{x_n} \partial_{x_{n-1}}(f_m) && \text{(by additivity of partial derivative)} \\ \mathcal{M}_{n-2}^{\alpha,\beta} &= \sum_{m=0}^{k-1} \partial_{x_n} \partial_{x_{n-1}}(f_m) && \text{(recursive structure of } \mathcal{M}_n \text{ from Proposition 24,} \\ &&& \text{and since } \partial_{x_n} \partial_{x_{n-1}}(f_k) = 0) \end{aligned}$$

Thus $\mathcal{M}_{n-2}^{\alpha,\beta}$ can be written as the sum of $k - 1$ polynomials, each of which is a ROP (by Fact 17). By the inductive hypothesis, $2(k - 1) \geq (n - 2)$. Therefore, $k \geq n/2$ contradicting our assumption. \square

From Proposition 27 and Fact 20, we can conclude:

Observation 28. *For all $m \in [k]$, f_m is a multiplicative ROP.*

Observation 28 and Lemma 21 together imply that for each $m \in [k]$ and $a \in [n]$, there exist $b \neq a \in [n]$ and $\gamma \in \mathbb{F}$ such that $\partial_{x_b}(f_m) |_{x_a=\gamma} = 0$. There are two cases to consider.

First, consider the case when for some m, a and the corresponding b, γ , it turns out that $\gamma \neq 0$. Assume without loss of generality that $m = k$, $a = n - 1$, $b = n$, so that

$\partial_{x_n}(f_k) |_{x_{n-1}=\gamma} = 0$. (For other indices the argument is symmetric.) Then

$$\mathcal{M}_n^{\alpha,\beta} = \sum_{i \in [k]} f_i \quad (\text{by assumption})$$

$$\partial_{x_n}(\mathcal{M}_n^{\alpha,\beta}) |_{x_{n-1}=\gamma} = \sum_{i \in [k]} \partial_{x_n}(f_i) |_{x_{n-1}=\gamma} \quad (\text{by additivity of partial derivative})$$

$$\mathcal{M}_{n-1}^{\alpha,\beta} |_{x_{n-1}=\gamma} = \sum_{i \in [k]} \partial_{x_n}(f_i) |_{x_{n-1}=\gamma} \quad (\text{recursive structure of } \mathcal{M}_n \text{ from Proposition } \boxed{24})$$

$$\mathcal{M}_{n-1}^{\alpha,\beta} |_{x_{n-1}=\gamma} = \sum_{i \in [k-1]} \partial_{x_n}(f_i) |_{x_{n-1}=\gamma} \quad (\text{since } \gamma \text{ is chosen as per Lemma } \boxed{21})$$

$$\mathcal{M}_{n-2}^{\alpha\gamma+\beta,\beta\gamma} = \sum_{i \in [k-1]} \partial_{x_n}(f_i) |_{x_{n-1}=\gamma} \quad (\text{recursive structure of } \mathcal{M}_n \text{ from Proposition } \boxed{24})$$

Therefore, $\mathcal{M}_{n-2}^{\alpha\gamma+\beta,\beta\gamma}$ can be written as a sum of at most $k - 1$ polynomials, each of which is a ROP (Fact [17](#)). By the inductive hypothesis, $2(k - 1) \geq n - 2$ implying that $k \geq n/2$ contradicting our assumption.

(Note: the term $\mathcal{M}_{n-2}^{\alpha\gamma+\beta,\beta\gamma}$ is what necessitates a stronger induction hypothesis than working with just $\alpha = 0, \beta = 1$.)

It remains to handle the case when for all $m \in [k]$ and $a \in [n]$, the corresponding value of γ to some x_b (as guaranteed by Lemma [21](#)) is 0. Examining the proof of Lemma [21](#), this implies that each leaf node in any of the ROFs can be made zero only by setting the corresponding variable to zero. That is, the linear forms at all leaves are of the form $a_i x_i$.

Since each φ_m is a multiplicative ROP, setting $x_n = 0$ makes the variables in the polynomial computed at the sibling of the leaf node $a_n x_n$ redundant. Hence setting $x_n = 0$ reduces the degree of each f_m by at least 2. That is, $\deg(f |_{x_n=0}) \leq n - 2$. But $\mathcal{M}_n^{\alpha,\beta} |_{x_n=0}$ equals $\mathcal{M}_{n-1}^{\beta,0} = \beta S_{n-1}^{n-1}$, which has degree $n - 1$, contradicting the assumption that $f = \mathcal{M}_n^{\alpha,\beta}$. \square

Combining the results of Lemma [26](#) and Lemma [25](#), we obtain the following theorem.

At $\alpha = 0, \beta = 1$, it yields Theorem [9](#).

Theorem 29. For each $n \geq 1$, any $\alpha \in F$ and any $\beta \in F \setminus \{0\}$, the polynomial $\alpha S_n^n + \beta S_n^{n-1}$ is in $\sum^k \cdot \text{ROP}$ but not in $\sum^{k-1} \cdot \text{ROP}$, where $k = \lceil n/2 \rceil$.

2.8 A family of 4-variate multilinear polynomials not in $\sum^2 \cdot \text{ROP}$

This section is devoted to proving Theorem [10](#). We want to find an explicit 4-variate multilinear polynomial that is not expressible as the sum of 2 ROPs.

Note that the proof of Theorem [9](#) does not help here, since the polynomials separating $\sum^2 \cdot \text{ROP}$ from $\sum^3 \cdot \text{ROP}$ have 5 or 6 variables. One obvious approach is to consider other combinations of the symmetric polynomials. This fails too; we can show that all such combinations are in $\sum^2 \cdot \text{ROP}$.

Proposition 30. For every choice of field constants a_i for each $i \in \{0, 1, 2, 3, 4\}$, the polynomial $\sum_{i=0}^4 a_i S_4^i$ can be expressed as the sum of two ROPs.

Proof. Let $g = \sum_i a_i S_4^i$. We obtain the expression for g in different ways in 4 different cases.

Case	Expression
$a_2 = a_3 = 0$	$g = a_0 + a_1 S_4^1 + a_4 S_4^4$
$a_2 = 0;$ $a_3 \neq 0$	$g = \left(a_1 + a_3 x_1 x_2 \right) \left(x_3 + x_4 + \frac{a_4}{a_3} x_3 x_4 \right)$ $+ \left((a_1 + a_3 x_3 x_4) \left(x_1 + x_2 - \frac{a_1 a_4}{a_3} \right) \right) + c$
$a_2 \neq 0;$ $a_2 a_4 = a_3^2$	$a_2 g = (a_1 + a_2(x_1 + x_2) + a_3 x_1 x_2) (a_1 + a_2(x_3 + x_4) + a_3 x_3 x_4)$ $+ (a_2^2 - a_1 a_3) (x_1 x_2 + x_3 x_4) + c$
$a_2 \neq 0;$ $a_2 a_4 \neq a_3^2$	$a_2 g = (a_1 + a_2(x_1 + x_2) + a_3 x_1 x_2) (a_1 + a_2(x_3 + x_4) + a_3 x_3 x_4)$ $+ \left(x_1 x_2 + \frac{a_2^2 - a_1 a_3}{a_2 a_4 - a_3^2} \right) ((a_2 a_4 - a_3^2) x_3 x_4 + a_2^2 - a_1 a_3) + c$

In the above, c is an appropriate field constant, and can be added to any ROP.

Notice that the first expression is a sum of two ROPs since it is the sum of a linear

polynomial and a single monomial. All the other expressions have two summands, each of which is a product of variable-disjoint bivariate polynomials (ignoring constant terms). Since every bivariate polynomial is a ROP, these representations are also sums of 2 ROPs. \square

Instead, we define a polynomial that gives carefully chosen weights to the monomials of S_4^2 . Let $f^{\alpha,\beta,\gamma}$ denote the following polynomial:

$$f^{\alpha,\beta,\gamma} = \alpha \cdot (x_1x_2 + x_3x_4) + \beta \cdot (x_1x_3 + x_2x_4) + \gamma \cdot (x_1x_4 + x_2x_3).$$

To keep notation simple, we will omit the superscript when it is clear from the context. In the theorem below, we obtain necessary and sufficient conditions on α, β, γ under which f can be expressed as a sum of two ROPs.

Theorem 31 (Hardness of representation for sum of 2 ROPs). *Let f be the polynomial $f^{\alpha,\beta,\gamma} = \alpha \cdot (x_1x_2 + x_3x_4) + \beta \cdot (x_1x_3 + x_2x_4) + \gamma \cdot (x_1x_4 + x_2x_3)$. The following are equivalent:*

1. f is not expressible as the sum of two ROPs over \mathbb{F} .
2. α, β, γ satisfy all the three conditions C1, C2, C3 listed below.

C1: $\alpha\beta\gamma \neq 0$.

C2: $(\alpha^2 - \beta^2)(\beta^2 - \gamma^2)(\gamma^2 - \alpha^2) \neq 0$.

C3: None of the equations $X^2 - d_i = 0$, $i \in [3]$, has a root in \mathbb{F} , where

$$d_1 = (+\alpha^2 - \beta^2 - \gamma^2)^2 - (2\beta\gamma)^2$$

$$d_2 = (-\alpha^2 + \beta^2 - \gamma^2)^2 - (2\alpha\gamma)^2$$

$$d_3 = (-\alpha^2 - \beta^2 + \gamma^2)^2 - (2\alpha\beta)^2$$

- Remark 32.** 1. It follows, for instance, that $2(x_1x_2 + x_3x_4) + 4(x_1x_3 + x_2x_4) + 5(x_1x_4 + x_2x_3)$ cannot be written as a sum of 2 ROPs over reals, yielding Theorem [10](#).
2. If F is an algebraically closed field, then for every α, β, γ , condition C3 fails, and so every $f^{\alpha, \beta, \gamma}$ can be written as a sum of 2 ROPs. However we do not know if there are other examples, or whether all multilinear 4-variate polynomials are expressible as the sum of two ROPs.
3. Even if F is not algebraically closed, condition C3 fails if for each $a \in F$, the equation $X^2 = a$ has a root.

Our strategy for proving Theorem [31](#) is a generalization of an idea used in [48](#). While Volkovich showed that 3-variate ROPs have a nice structural property in terms of their partial derivatives and commutators, we show that the sums of two 4-variate ROPs have at least one nice structural property in terms of their bivariate restrictions, partial derivatives, and commutators. Then we show that provided α, β, γ are chosen carefully, the polynomial $f^{\alpha, \beta, \gamma}$ will not satisfy any of these properties and hence cannot be a sum of two ROPs.

To prove Theorem [31](#), we first consider the easier direction, $1 \Rightarrow 2$, and prove the contrapositive.

Lemma 33. *If α, β, γ do not satisfy all of C1, C2, C3, then the polynomial f can be written as a sum of 2 ROPs.*

Proof. **C1 false:** If any of α, β, γ is zero, then by definition f is the the sum of at most two ROPs.

C2 false: Without loss of generality, assume $\alpha^2 = \beta^2$, so $\alpha = \pm\beta$. Then f is computed by $f = \alpha \cdot (x_1 \pm x_4)(x_2 \pm x_3) + \gamma \cdot (x_1x_4 + x_2x_3)$.

C1 true; C3 false: Without loss of generality, the equation $X^2 - d_1 = 0$ has a root

τ . We try to express f as

$$\alpha(x_1 - ax_3)(x_2 - bx_4) + \beta(x_1 - cx_2)(x_3 - dx_4).$$

The coefficients for x_3x_4 and x_2x_4 force $ab = 1$, $cd = 1$, giving the form

$$\alpha(x_1 - ax_3)(x_2 - \frac{1}{a}x_4) + \beta(x_1 - cx_2)(x_3 - \frac{1}{c}x_4).$$

Comparing the coefficients for x_1x_4 and x_2x_3 , we obtain the constraints

$$-\frac{\alpha}{a} - \frac{\beta}{c} = \gamma; \quad -\alpha a - \beta c = \gamma$$

Expressing a as $\frac{-\gamma - \beta c}{\alpha}$, we get a quadratic constraint on c ; it must be a root of the equation

$$Z^2 + \frac{-\alpha^2 + \beta^2 + \gamma^2}{\beta\gamma}Z + 1 = 0.$$

Using the fact that $\tau^2 = d_1 = (-\alpha^2 + \beta^2 + \gamma^2)^2 - (2\beta\gamma)^2$, we see that indeed this equation does have roots. The left-hand side splits into linear factors, giving

$$(Z - \delta)(Z - \frac{1}{\delta}) = 0 \quad \text{where} \quad \delta = \frac{\alpha^2 - \beta^2 - \gamma^2 + \tau}{2\beta\gamma}.$$

It is easy to verify that $\delta \neq 0$ and $\delta \neq -\frac{\gamma}{\beta}$ (since $\alpha \neq 0$). Further, define $\mu = \frac{-(\gamma + \beta\delta)}{\alpha}$. Then μ is well-defined (because $\alpha \neq 0$) and is also non-zero. Now setting $c = \delta$ and $a = \mu$, we have satisfied all the constraints and so we can write f as the sum of 2 ROPs as follows:

$$f = \alpha(x_1 - \mu x_3)(x_2 - \frac{1}{\mu}x_4) + \beta(x_1 - \delta x_2)(x_3 - \frac{1}{\delta}x_4).$$

□

Now we consider the harder direction: $2 \Rightarrow 1$. Again, we consider the contrapositive.

We first show (Lemma [34](#)) a structural property satisfied by every polynomial in $\Sigma^2 \cdot \text{ROP}$: it must satisfy at least one of the three properties $C1', C2', C3'$ described in the lemma. We then show (Lemma [35](#)) that under the conditions $C1, C2, C3$ from the theorem statement, f does not satisfy any of $C1', C2', C3'$; it follows that f is not expressible as the sum of 2 ROPs.

Lemma 34. *Let g be a 4-variate multilinear polynomial over the field F which can be expressed as a sum of 2 ROPs. Then at least one of the following conditions is true:*

C1': *There exist $i, j \in [4]$ and $a, b \in F$ such that $g|_{x_i=a, x_j=b}$ is linear.*

C2': *There exist $i, j \in [4]$ such that $x_i, x_j, \partial_{x_i}(g), \partial_{x_j}(g), 1$ are linearly dependent.*

C3': *$g = l_1 \cdot l_2 + l_3 \cdot l_4$ where l_i s are linear forms, l_1 and l_2 are variable-disjoint, and l_3 and l_4 are variable-disjoint.*

Proof. Let φ be a sum of 2 ROPs computing g . Let v_1 and v_2 be the children of the topmost $+$ gate. The proof is in two steps. First, we reduce to the case when $|\text{Var}(v_1)| = |\text{Var}(v_2)| = 4$. Then we use a case analysis to show that at least one of the aforementioned conditions hold true. In both steps, we will repeatedly use Proposition [18](#), which showed that any 3-variate ROP can be reduced to a linear polynomial by substituting a single variable with a field constant. We now proceed with the proof.

Suppose $|\text{Var}(v_1)| \leq 3$. Applying Proposition [18](#) first to v_1 and then to the resulting restriction of v_2 , one can see that there exist $i, j \in [4]$ and $a, b \in F$ such that $g|_{x_i=a, x_j=b}$ is a linear polynomial. So condition $C1'$ is satisfied.

Now assume that $|\text{Var}(v_1)| = |\text{Var}(v_2)| = 4$. Depending on the type of gates of v_1 and v_2 , we consider 3 cases.

Case 1: Both v_1 and v_2 are \times gates. Then g can be represented as $M_1 \cdot M_2 + M_3 \cdot M_4$ where (M_1, M_2) and (M_3, M_4) are variable-disjoint ROPs.

Suppose that for some i , $|\text{Var}(M_i)| = 1$. Then, $g|_{M_i \rightarrow 0}$ is a 3-variate restriction of f and is clearly an ROP. Applying Proposition [18](#) to this restriction, we see that condition $C1'$ holds.

Otherwise each M_i has $|\text{Var}(M_i)| = 2$.

Suppose (M_1, M_2) and (M_3, M_4) define distinct partitions of the variable set. Assume without loss of generality that $g = M_1(x_1, x_2) \cdot M_2(x_3, x_4) + M_3(x_1, x_3) \cdot M_4(x_2, x_4)$. If all M_i s are linear forms, it is clear that condition $C3'$ holds. If not, assume that M_1 is of the form $l_1(x_1) \cdot m_1(x_2) + c_1$ where l_1, m_1 are linear forms and $c_1 \in \mathbb{F}$. Now $g|_{l_1 \rightarrow 0} = c_1 \cdot M_2(x_3, x_4) + M'_3(x_3) \cdot M_4(x_2, x_4)$. Either set x_3 to make M'_3 zero, or, if that is not possible because M'_3 is a non-zero field constant, then set $x_4 \rightarrow b$ where $b \in \mathbb{F}$. In both cases, by setting at most 2 variables, we obtain a linear polynomial, so $C1'$ holds.

Otherwise, (M_1, M_2) and (M_3, M_4) define the same partition of the variable set. Assume without loss of generality that $g = M_1(x_1, x_2) \cdot M_2(x_3, x_4) + M_3(x_1, x_2) \cdot M_4(x_3, x_4)$. If one of the M_i s is linear, say without loss of generality that M_1 is a linear form, then $g|_{M_4 \rightarrow 0}$ is a 2-variate restriction which is also a linear form, so $C1'$ holds. Otherwise, none of the M_i s is a linear form. Then each M_i can be represented as $l_i \cdot m_i + c_i$ where l_i, m_i are univariate linear forms and $c_i \in \mathbb{F}$. We consider a 2-variate restriction which sets l_1 and m_4 to 0. (Note that $\text{Var}(l_1) \cap \text{Var}(m_4) = \emptyset$.) Then the resulting polynomial is a linear form, so $C1'$ holds.

Case 2: Both v_1 and v_2 are + gates. Then g can be written as $f = M_1 + M_2 + M_3 + M_4$ where (M_1, M_2) and (M_3, M_4) are variable-disjoint ROPs.

Suppose (M_1, M_2) and (M_3, M_4) define distinct partitions of the variable set.

Suppose further that there exists M_i such that $|\text{Var}(M_i)| = 1$. Without loss of generality, $\text{Var}(M_1) = \{x_1\}$, $\{x_1, x_2\} \subseteq \text{Var}(M_3)$, and $x_3 \in \text{Var}(M_4)$. Any setting to x_2 and x_4 results in a linear polynomial, so $C1'$ holds.

So assume without loss of generality that $g = M_1(x_1, x_2) + M_2(x_3, x_4) + M_3(x_1, x_3) + M_4(x_2, x_4)$. Then for $a, b \in \mathbb{F}$, $g|_{x_1=a, x_4=b}$ is a linear polynomial, so $C1'$ holds.

Otherwise, (M_1, M_2) and (M_3, M_4) define the same partition of the variable set. Again, if say $|\text{Var}(M_1)| = 1$, then setting two variables from M_2 shows that $C1'$ holds. So assume without loss of generality that $g = M_1(x_1, x_2) + M_2(x_3, x_4) + M_3(x_1, x_2) + M_4(x_3, x_4)$. Then for $a, b \in \mathbb{F}$, $g|_{x_1=a, x_3=b}$ is a linear polynomial, so again $C1'$ holds.

Case 3: One of v_1, v_2 is a $+$ gate and the other is a \times gate. Then g can be written as $g = M_1 + M_2 + M_3 \cdot M_4$ where (M_1, M_2) and (M_3, M_4) are variable-disjoint ROPs. Suppose that $|\text{Var}(M_3)| = 1$. Then $g|_{M_3 \rightarrow 0}$ is a 3-variate restriction which is a ROP. Using Proposition [18](#), we get a 2-variate restriction of g which is also linear, so $C1'$ holds. The same argument works when $|\text{Var}(M_4)| = 1$. So assume that M_3 and M_4 are bivariate polynomials.

Suppose that (M_1, M_2) and (M_3, M_4) define distinct partitions of the variable set. Assume without loss of generality that $g = M_1 + M_2 + M_3(x_1, x_2) \cdot M_4(x_3, x_4)$, and x_3, x_4 are separated by M_1, M_2 . Then $g|_{M_3 \rightarrow 0}$ is a 2-variate restriction which is also linear, so $C1'$ holds.

Otherwise (M_1, M_2) and (M_3, M_4) define the same partition of the variable set. Assume without loss of generality that $g = M_1(x_1, x_2) + M_2(x_3, x_4) + M_3(x_1, x_2) \cdot M_4(x_3, x_4)$. If M_1 (or M_2) is a linear form, then consider a 2-variate restriction of g which sets M_4 (or M_3) to 0. The resulting polynomial is a linear form. Similarly if M_3 (or M_4) is of the form $l \cdot m + c$ where l, m are univariate linear forms, then we consider a 2-variate restriction which sets l to 0 and some $x_i \in \text{Var}(M_4)$ to a field constant. The resulting polynomial again is a linear form. In all these cases, $C1'$ holds.

The only case that remains is that M_3 and M_4 are linear forms while M_1 and M_2 are not. Assume that $M_1 = (a_1x_1 + b_1)(a_2x_2 + b_2) + c$ and $M_3 = a_3x_1 + b_3x_2 + c_3$. Then $\partial_{x_1}(g) = a_1(a_2x_2 + b_2) + a_3M_4$ and $\partial_{x_2}(g) = (a_1x_1 + b_1)a_2 + b_3M_4$. It follows that $b_3 \cdot \partial_{x_1}(g) - a_3 \cdot \partial_{x_2}(g) + a_1a_2a_3x_1 - a_1a_2b_3x_2 = a_1b_2b_3 - b_1a_2a_3 \in \mathbb{F}$, and hence the polynomials $x_1, x_2, \partial_{x_1}(g), \partial_{x_2}(g)$ and 1 are linearly dependent. Therefore, condition

$C2'$ of the lemma is satisfied. □

Lemma 35. *If α, β, γ satisfy conditions $C1, C2, C3$ from the statement of Theorem [31](#), then the polynomial $f^{\alpha, \beta, \gamma}$ does not satisfy any of the properties $C1', C2', C3'$ from Lemma [34](#).*

Proof. **C1** \Rightarrow \neg **C1'**: Since $\alpha\beta\gamma \neq 0$, f contains all possible degree 2 monomials. Hence after setting $x_i = a$ and $x_j = b$, the monomial $x_k x_l$ where $k, l \in [4] \setminus \{i, j\}$ still survives.

C2 \Rightarrow \neg **C2'**: The proof is by contradiction. Assume to the contrary that for some i, j , without loss of generality say for $i = 1$ and $j = 2$, the polynomials $x_1, x_2, \partial_{x_1}(f), \partial_{x_2}(f), 1$ are linearly dependent. Note that $\partial_{x_1}(f) = \alpha x_2 + \beta x_3 + \gamma x_4$ and $\partial_{x_2}(f) = \alpha x_1 + \gamma x_3 + \beta x_4$. This implies that the vectors $(1, 0, 0, 0, 0), (0, 1, 0, 0, 0), (0, \alpha, \beta, \gamma, 0), (\alpha, 0, \gamma, \beta, 0)$ and $(0, 0, 0, 0, 1)$ are linearly dependent. This further implies that the vectors (β, γ) and (γ, β) are linearly dependent. Therefore, $\beta = \pm\gamma$, contradicting **C2**.

C1 \wedge **C2** \wedge **C3** \Rightarrow \neg **C3'**: Suppose, to the contrary, that $C3'$ holds. That is, f can be written as $f = l_1 \cdot l_2 + l_3 \cdot l_4$ where (l_1, l_2) and (l_3, l_4) are variable-disjoint linear forms. By the preceding arguments, we know that f does not satisfy $C1'$ or $C2'$.

First consider the case when (l_1, l_2) and (l_3, l_4) define the same partition of the variable set. Assume without loss of generality that $\text{Var}(l_1) = \text{Var}(l_3), \text{Var}(l_2) = \text{Var}(l_4)$, and $|\text{Var}(l_1)| \leq 2$. Setting the variables in l_1 to any field constants yields a linear form, so f satisfies **C1'**, a contradiction.

Hence it must be the case that (l_1, l_2) and (l_3, l_4) define different partitions of the variable set. Since all degree-2 monomials are present in f , each pair x_i, x_j must be separated by at least one of the two partitions. This implies that both partitions have exactly 2 variables in each part. Assume without loss of generality that $f = l_1(x_1, x_2) \cdot l_2(x_3, x_4) + l_3(x_1, x_3) \cdot l_4(x_2, x_4)$.

At this point, we use properties of the commutator of f ; recall Definition [22](#). By

Lemma [23](#), we know that l_2 divides $\Delta_{12}f$. We compute $\Delta_{12}f$ explicitly for our candidate polynomial:

$$\begin{aligned}\Delta_{12}f &= (\alpha x_3 x_4)(\alpha + (\beta + \gamma)(x_3 + x_4) + \alpha x_3 x_4) \\ &\quad - (\beta x_4 + \gamma x_3 + \alpha x_3 x_4)(\beta x_3 + \gamma x_4 + \alpha x_3 x_4) \\ &= -\beta\gamma(x_3^2 + x_4^2) + (\alpha^2 - \beta^2 - \gamma^2)x_3 x_4\end{aligned}$$

Since l_2 divides $\Delta_{12}f$, $\Delta_{12}f$ is not irreducible but is the product of two linear factors. Since $\Delta_{12}f(0, 0) = 0$, at least one of the linear factors of $\Delta_{12}f$ must vanish at $(0, 0)$. Let $x_3 - \delta x_4$ be such a factor. Then $\Delta_{12}(f)$ vanishes not only at $(0, 0)$, but whenever $x_3 = \delta x_4$. Substituting $x_3 = \delta x_4$ in $\Delta_{12}f$, we get

$$-\delta^2\beta\gamma - \beta\gamma + \delta(\alpha^2 - \beta^2 - \gamma^2) = 0$$

Hence δ is of the form

$$\delta = \frac{-(\alpha^2 - \beta^2 - \gamma^2) \pm \sqrt{(\alpha^2 - \beta^2 - \gamma^2)^2 - 4\beta^2\gamma^2}}{-2\beta\gamma}$$

Hence $2\beta\gamma\delta - (\alpha^2 - \beta^2 - \gamma^2)$ is a root of the equation $X^2 - d_1 = 0$, contradicting the assumption that C3 holds.

Hence it must be the case that C3' does not hold. □

With this, the proof of Theorem [31](#) is complete.

The conditions imposed on α, β, γ in Theorem [31](#) are tight and irredundant. Below we give some explicit examples over the field of reals.

1. $f = 2(x_1x_2 + x_3x_4) + 2(x_1x_3 + x_2x_4) + 3(x_1x_4 + x_2x_3)$ satisfies conditions C1 and C3 from the Theorem but not C2; $\alpha = \beta$. A \sum^2 -ROP representation for f is $f = 2(x_1 + x_4)(x_2 + x_3) + 3(x_1x_4 + x_2x_3)$.

2. $f = 2(x_1x_2 + x_3x_4) - 2(x_1x_3 + x_2x_4) + 3(x_1x_4 + x_2x_3)$ satisfies conditions C1 and C3 but not C2; $\alpha = -\beta$. A \sum^2 -ROP representation for f is $f = 2(x_1 - x_4)(x_2 - x_3) + 3(x_1x_4 + x_2x_3)$.
3. $f = (x_1x_2 + x_3x_4) + 2(x_1x_3 + x_2x_4) + 3(x_1x_4 + x_2x_3)$ satisfies conditions C1 and C2 but not C3. Note that in this case, $d_1 = (1^2 - 2^2 - 3^2)^2 - (2 \cdot 2 \cdot 3)^2 = 0$. Clearly $X^2 - d_1 = X^2 = 0$ has a real root. A \sum^2 -ROP representation for f is $f = (x_1 + x_3)(x_2 + x_4) + 2(x_1 + x_2)(x_3 + x_4)$.

2.9 Discussion

1. We have seen in Proposition [11](#) that every n -variate multilinear polynomial ($n \geq 4$) can be written as the sum of $3 \times 2^{n-4}$ ROPs. The counting argument from Proposition [14](#) shows that there exist multilinear polynomials f requiring exponentially many ROPs summands; if $f \in \sum^k$ -ROP then $k = \Omega(2^n/n^2)$. Our general upper bound on k is $O(2^n)$, leaving a small gap between the lower and upper bound. What is the true tight bound? Can we find explicit polynomials where exponentially large k is necessary and sufficient in any \sum^k -ROP expression? One such example is the polynomial defined by Raz and Yehudayoff in [40](#); as shown in [9](#), k must be exponential in $\Omega(n^{1/3}/\log n)$. But we do not know whether this value of k is asymptotically tight.
2. We have shown in Theorem [9](#) that for each k , \sum^k -ROP can be separated from \sum^{k-1} -ROP by a polynomial on $2k - 1$ variables. Can we separate these classes with fewer variables? Note that any separating polynomial must have $\Omega(\log k)$ variables.
3. In particular, can 4-variate multilinear polynomials separate sums of 3 ROPs from sums of 2 ROPs over every field? If not, what is an explicit example?
4. We now understand ROPs and ROFs very well, [48](#). However, our understand-

ing of sums of ROPs is not so good. Can we at least characterise \sum^2 -ROPs?

Chapter 3

Computation over semirings

In this chapter, we give some preliminaries needed for the rest of this thesis.

Recall the definition of circuits from Chapter [1](#). In the rest of this thesis, we will be interested in circuits over a particular *semiring* instead of fields. We will mainly consider circuits over the tropical semiring.

A circuit C syntactically *produces* a polynomial p_C over the semiring $\mathcal{S} = (S, \oplus, \otimes)$ in a natural way; at a leaf, the polynomial produced is the leaf label, and at intermediate nodes, the polynomial produced is obtained by combining polynomials produced at the children using the operation labeling the gate. Using the distributivity of \otimes over \oplus , the polynomial produced at the output gate can be represented as a \oplus sum of monomials, where within each monomial we use the \otimes product.

A circuit C *computes* a polynomial p if the polynomial p_C produced by C agrees with the polynomial p at all input settings. Over the arithmetic semiring $A = (\mathbb{N}, +, \times)$, computing and producing are equivalent in terms of the size of the circuit required. However, over other semirings, there can be significant differences. In particular, this is the case for the tropical semiring Min , the focus of this paper. We use the notation $\mathcal{S}(p)$ to denote the size of the smallest circuit computing (not producing) p over the semiring \mathcal{S} .

The tropical semiring Min is the semiring $\text{Min} = (\mathbb{N}, \min, +)$, with 0 being the iden-

tity for $+$ and ∞ the identity for \min . A circuit over Min , with variables x_1, \dots, x_n , produces a polynomial of the form $\min\{\ell_1, \ell_2, \dots, \ell_t\}$ where each monomial ℓ_r is of the form $c_0 + c_1x_1 + \dots + c_nx_n$, for non-negative integers c_i .

For a polynomial $p(X)$, $\text{Mon}(p)$ denotes the set of monomials of p . Let \preceq be the following (partial) ordering amongst monomials over the variable set $X = \{x_1, \dots, x_n\}$: $c_0 + c_1x_1 + \dots + c_nx_n \preceq d_0 + d_1x_1 + \dots + d_nx_n$ if $c_i \leq d_i$ for all i . Then $\text{Mon}_{le}(p)$, the *lower envelope* of monomials of p , denotes the set of those monomials in $\text{Mon}(p)$ that are minimal with respect to \preceq , and the lower envelope of p , denoted p_{le} , is the minimum (the \oplus sum) of these monomials. For instance consider the polynomial $p(X) = \min(x_1, x_2, x_1 + x_2, 2x_1 + 3x_2)$ over the Min semiring. In this case, $\text{Mon}(p) = \{x_1, x_2, x_1 + x_2, 2x_1 + 3x_2\}$, $\text{Mon}_{le}(p) = \{x_1, x_2\}$ and $p_{le} = \min(x_1, x_2)$.

Over the semiring Min , if polynomials p and q have the same lower envelope, then they compute the same values everywhere. Thus for a polynomial p , $\text{Min}(p)$ is the size of the smallest $(\min, +)$ circuit producing a polynomial whose lower envelope is p_{le} .

Computation over the semiring Min lies somewhere in between monotone Boolean computation and monotone arithmetic computation. For any polynomial p described as an \oplus sum of \otimes monomials, the following relation holds: $B(p) \leq \text{Min}(p) \leq A(p)$. Here B and A denote the Boolean and arithmetic semirings respectively.

For convenience in describing the upper bounds, we may use the values $0, \infty$ at the leaves, but these can be propagated upwards without increasing the size.

Chapter 4

Computing max using $(\min, +)$ formulas

4.1 Highlights

In this chapter, we present our results on lower bounds for *difference of $(\min, +)$ formulas*. We start by defining $(\min, +)$ formulas. We then define the problem of computing the maximum value among n variables using $(\min, +)$ formulas and also provide our own motivation for studying this problem. We then discuss some related work and finally our results on this problem. Our main results are as follows:

- No $(\min, +)$ formula over \mathbb{N} or \mathbb{Z} can compute the maximum value among n variables.
- Any difference of $(\min, +)$ formulas that computes the maximum of n variables must have size at least $\Theta(n \log n)$.

Organization of chapter

The rest of this chapter is organized as follows. In Section [4.2](#), we introduce our problem of interest. In Section [4.3](#), we discuss some useful transformations and

easy upper bounds. Then in Section 4.4, we introduce our main technical tool for proving lower bounds, that is, graph entropy. In Section 4.5 and Section 4.6, we further discuss some upper bounds for computing the maximum. In Section 4.7, we discuss the proof of our main lower bound. In Section 4.8, we discuss the case when the model of $(\min, +)$ formulas is strengthened using a monus operation. Finally, in Section 4.9, we conclude with stating some still open questions.

4.2 Introduction

4.2.1 Background

Let X denote the set of variables $\{x_1, \dots, x_n\}$. We use \tilde{x} to denote $(x_1, x_2, \dots, x_n, 1)$. We use e_i to denote the $(n+1)$ -dimensional vector with a 1 in the i th coordinate and zeroes elsewhere. For $i \in [n]$, we also use e_i to denote an assignment to the variables x_1, x_2, \dots, x_n where x_i is set to 1 and all other variables are set to 0.

Definition 36. For $0 \leq r \leq n$, the n -variate functions Sum_n , MinSum_n^r and MaxSum_n^r are as defined below.

$$\begin{aligned} \text{Sum}_n &= \sum_{i=1}^n x_i \\ \text{MinSum}_n^r &= \min \left\{ \sum_{i \in S} x_i \mid S \subseteq n, |S| = r \right\} \\ \text{MaxSum}_n^r &= \max \left\{ \sum_{i \in S} x_i \mid S \subseteq n, |S| = r \right\} \end{aligned}$$

Note that MinSum_n^0 and MaxSum_n^0 are the constant function 0, and MinSum_n^1 and MaxSum_n^1 are just the min and max functions respectively.

Observation 37. For $1 \leq r < n$, $\text{MinSum}_n^n = \text{MaxSum}_n^n = \text{Sum}_n = \text{MinSum}_n^r + \text{MaxSum}_n^{n-r}$.

We consider the following setting. Suppose we are given n input variables

x_1, x_2, \dots, x_n and we want to find a formula which computes the maximum value taken by these variables, $\max(x_1, x_2, \dots, x_n)$. If variables are restricted to take non-negative integer values, it is easy to show that no $(\min, +)$ formula can compute \max . (See the proof of Theorem [45](#) for details.) Hence to compute the maximum, we must strengthen this model – an obvious way is by allowing minus gates as well. Now we have a very small linear sized formula: $\max(x_1, x_2, \dots, x_n) = 0 - \min(0 - x_1, 0 - x_2, \dots, 0 - x_n)$. However, this solution is not satisfactory for two reasons: firstly, it uses many minus gates, and secondly, intermediate gates in this formula can compute negative integer values even though we are working over natural numbers. Is this necessary? Addressing the first question, it turns out that just a single minus gate, appearing at the top, suffices. That is, we can compute the maximum as the *difference* of two $(\min, +)$ expressions. Here is one such computation:

$$(\text{Sum of all variables}) - \min_i (\text{Sum of all variables except } x_i).$$

The second expression can be computed by a linear-size $(\min, +)$ circuit or by a $(\min, +)$ formula of size $n \log n$ (see Lemma [48](#)). And this computation addresses the second question as well; all intermediate values are non-negative. Can we do any better? We show that this simple difference formula is indeed the best we can achieve in this model.

The main motivation behind studying this question is the following question: Does there exist a naturally occurring function f for which $(\min, +)$ circuits are super-polynomially weaker than $(\max, +)$ circuits? There are two possibilities:

1. Show that \max can be implemented using a small $(\min, +)$ circuit.
2. Come up with an explicit function f which has *small* $(\max, +)$ circuits but requires *large* $(\min, +)$ circuits.

Since we show that no $(\min, +)$ formula (or circuit) can compute \max , option [1](#) is ruled out. In the weaker model of formulas instead of circuits, we show that

any difference of $(\min, +)$ formulas computing \max should have size at least $n \log n$. This yields us a slight, super-linear, separation between $(\max, +)$ formulas and difference of $(\min, +)$ formulas. Note that a similar question was also asked in [24]: Does there exist a naturally occurring *polynomial* for which the $(\min, +)$ semiring is super-polynomially weaker than the $(\max, +)$ semiring? Note that the same polynomial can have different interpretations over different semirings. For instance, consider the polynomial $f(x_1, x_2) = x_1^2 + x_1x_2 + x_2^3$. Over the $(\min, +)$ semiring, it is interpreted as $\min\{2x_1, x_1 + x_2, 3x_2\}$ while over the $(\max, +)$ semiring, it is interpreted as $\max\{2x_1, x_1 + x_2, 3x_2\}$.

4.2.2 Motivation

Many dynamic programming algorithms correspond to $(\min, +)$ circuits over an appropriate semiring. Notable examples include the Bellman-Ford-Moore (BFM) algorithm for the single-source-shortest-path problem (SSSP) [5, 15, 37], the Floyd-Warshall (FW) algorithm for the All-Pairs-Shortest-Path (APSP) problem [13, 49], and the Held-Karp (HK) algorithm for the Travelling Salesman Problem (TSP) [19]. All these algorithms are just recursively constructed $(\min, +)$ circuits. For example, both the BFM and the FW algorithms give $O(n^3)$ sized $(\min, +)$ circuits while the HK algorithm gives a $O(n^2 \cdot 2^n)$ sized $(\min, +)$ circuit. Matching lower bounds were proved for TSP in [22], for APSP in [24], and for SSSP in [27]. So, proving tight lower bounds for circuits over $(\min, +)$ can help us understand the power and limitations of dynamic programming. We refer the reader to [24, 25] for more results on $(\min, +)$ circuit lower bounds.

Apart from the many natural settings where the tropical semiring $\text{Min} = (\min, +, \mathbb{N} \cup \{\infty\}, 0, \infty)$ crops up, it is also interesting because it can simulate the Boolean semiring for monotone computation. The mapping is straightforward: $0, 1, \vee, \wedge$ in the Boolean semiring are replaced by $\infty, 0, \min, +$ respectively in the tropical semiring. Proving lower bounds for $(\min, +)$ formulas could be easier than for monotone

Boolean formulas because the $(\min, +)$ formula has to compute a function correctly at all values, not just at $0, \infty$. In particular, we draw attention to the following observation in [24]: “The power of tropical circuits lies somewhere between that of monotone Boolean circuits and monotone arithmetic circuits, and the gaps may even be exponential.” Thus, over the tropical semiring Min , lower bounds can be inherited from the monotone Boolean setting, and upper bounds from the monotone arithmetic setting.

Note that algorithms for problems like computing the diameter of a graph are naturally expressed using $(\min, \max, +)$ circuits. This makes the cost of converting a \max gate to a $(\min, +)$ circuit or formula an interesting measure.

A related question arises in the setting of counting classes defined by arithmetic circuits and formulas. Circuits over \mathbb{N} , with specific resource bounds, count accepting computation paths or proof-trees in a related resource-bounded Turing machine model defining a class \mathcal{C} . The counting function class is denoted $\#\mathcal{C}$. The difference of two such functions in a class $\#\mathcal{C}$ is a function in the class $\text{Diff}\mathcal{C}$. On the other hand, circuits with the same resource bounds, but over \mathbb{Z} , or equivalently, with subtraction gates, describe the function class $\text{Gap}\mathcal{C}$. For most complexity classes \mathcal{C} , a straightforward argument shows that that $\text{Diff}\mathcal{C}$ and $\text{Gap}\mathcal{C}$ coincide upto polynomial factors. See [2] for further discussion on this. In this framework, we restrict attention to computation over \mathbb{N} and see that as a member of a Gap class over $(\min, +)$, \max has linear-size formulas, whereas as a member of a Diff class, it requires $\Omega(n \log n)$ size.

4.2.3 Our results and techniques

We now formally state our results and briefly comment on the techniques used to prove them.

1. For $n \geq 2$, no $(\min, +)$ formula over \mathbb{N} can compute $\max(x_1, x_2, \dots, x_n)$. (Theorem [45])

The proof is simple: apply a carefully chosen restriction to the variables and show that the $(\min, +)$ formula does not output the correct value of \max on this restriction.

2. $\max(x_1, x_2, \dots, x_n)$ can be computed by a difference of two $(\min, +)$ formulas with total size $n + n \lceil \log n \rceil$. (Theorem [47](#))

One of the formulas computes just the sum of all n variables and is clearly of linear size. The other formula computes the minimum sum of $n - 1$ distinct variables; using recursion, we obtain the claimed size bound.

3. Let F_1, F_2 be $(\min, +)$ formulas over \mathbb{N} such that $F_1 - F_2 = \max(x_1, x_2, \dots, x_n)$. Then F_1 must have at least n leaves and F_2 at least $n \log n$ leaves. (Theorem [50](#))

A major ingredient in our proof is the definition of a measure for functions computable by constant-free $(\min, +)$ formulas, and relating this measure to formula size. The measure involves terms analogous to minterms of a monotone Boolean function, and uses the entropy of an associated graph under the uniform distribution on its vertices. In the setting of monotone Boolean functions, this technique was used in [38](#) to give formula size lower bounds. We adapt that technique to the $(\min, +)$ setting.

The same technique also yields the following lower bound: Any $(\min, +)$ formula computing the minimum over the sums of $n - 1$ variables must have at least $n \log n$ leaves. This is tight. (Lemma [48](#) and Corollary [55](#)) Note that for the corresponding Boolean function Th_n^{n-1} , a lower bound of $n \log n$ is known for monotone Boolean formulas [21](#), and hence by [24](#), this lower bound automatically carries over to the $(\min, +)$ semiring. However, transferring the lower bound seems to require the use of ∞ . Our argument shows that the lower bound holds even if we are interested in computation over \mathbb{N} without the element ∞ .

4. Arguably, over totally ordered monoids that are not groups, a *monus* operation is a more appropriate version of the inverse than minus. In Section [4.8](#), we briefly discuss augmenting $(\min, +)$ formulas with gates computing monus as opposed to minus.

4.3 Transformations and Upper bounds

We consider the computation of $\max\{x_1, \dots, x_n\}$ over \mathbb{N} using $(\min, +)$ formulas. To start with, we describe some properties of $(\min, +)$ formulas that we use repeatedly. The first property, Proposition [39](#) below, is expressing the function computed by a formula as a depth-2 polynomial where $+$ plays the role of multiplication and \min plays the role of addition. The next properties, Proposition [40](#) and [41](#) below, deal with removing redundant sub-expressions created by the constant zero or moving common parts aside.

Definition 38. *Let F be a $(\min, +)$ formula with leaves labeled from $X \cup \mathbb{N}$. For each gate $v \in F$, we construct a set $S_v \subseteq \mathbb{N}^{n+1}$ as described below.*

We construct the sets inductively based on the depth of v .

1. *Case 1: v is a leaf labeled α for some $\alpha \in \mathbb{N}$. Then $S_v = \{\alpha \cdot e_{n+1}\}$. (Recall, e_i is the unit vector with 1 at the i th coordinate and zero elsewhere).*
2. *Case 2: v is a leaf labeled x_i for some $i \in [n]$. Then $S_v = \{e_i\}$.*
3. *Case 3: $v = \min\{u, w\}$. Then $S_v = S_u \cup S_w$.*
4. *Case 4: $v = u + w$. Then $S_v = \{\tilde{a} + \tilde{b} \mid \tilde{a} \in S_u, \tilde{b} \in S_w\}$ (coordinate-wise addition).*

Let r be the output gate of F . We denote by $S(F)$ the set S_r so constructed.

Below we give an example of a $(\min, +)$ formula along with the sets corresponding to each gate.

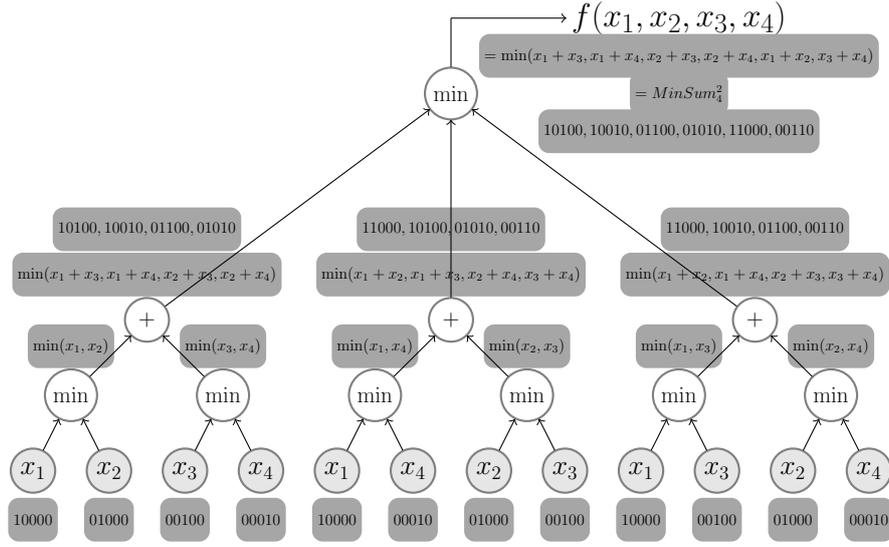


Figure 4.1: $(\min, +)$ formula

It is straightforward to see that if F has no constants (so Case 1 is never invoked), then a_{n+1} remains 0 throughout the construction of the sets S_v . Hence if F is constant-free, then for each $\tilde{a} \in S(F)$, $a_{n+1} = 0$.

By construction, the set $S(F)$ describes the function computed by F . (In the language of [24], it represents the unique polynomial “produced” by the formula.) Thus we have the following:

Proposition 39. *Let F be a formula with \min and $+$ gates, with leaves labeled by elements of $\{x_1, \dots, x_n\} \cup \mathbb{N}$. For each gate $v \in F$, let f_v denote the function computed at v .*

$$\text{Then } f_v = \min\{\langle \tilde{a} \cdot \tilde{x} \rangle \mid \tilde{a} \in S_v\}.$$

One can easily verify the above proposition in Figure 4.1.

The following proposition is an easy consequence of the construction in Definition 38.

Proposition 40. *Let F be a $(\min, +)$ formula over \mathbb{N} . Let G be the formula obtained from F by replacing all constants by the constant 0. Let H be the constant-free formula obtained from G by eliminating 0s from G through repeated replacements of $0 + A$ by A , $\min\{0, A\}$ by 0. Then*

1. $L(H) \leq L(G) = L(F)$,
2. $S(G) = \{\tilde{b} \mid b_{n+1} = 0, \exists \tilde{a} \in S(F), \forall i \in [n], a_i = b_i\}$, and
3. G and H compute the same function $\min\{\langle \tilde{b} \cdot \tilde{x} \rangle \mid \tilde{b} \in S(G)\}$.

(Note: It is not the claim that $S(G) = S(H)$. Indeed, this may not be the case. For instance, let $F = x + \min\{1, x + y\}$. Then $S(F) = \{101, 210\}$, $S(G) = \{100, 210\}$, $S(H) = \{100\}$. However, the functions computed are the same.)

The next proposition shows how to remove “common” contributors to $S(F)$ without increasing the formula size.

Proposition 41. *Let F be a $(\min, +)$ formula computing a function f .*

If, for some $i \in [n+1]$, $a_i > 0$ for every $\tilde{a} \in S(F)$, then $f - \langle e_i \cdot \tilde{x} \rangle$ can be computed by a $(\min, +)$ formula F' of size at most $\text{size}(F)$. Furthermore, $S(F') = \{\tilde{b} \mid \exists \tilde{a} \in S(F), \tilde{b} = \tilde{a} - e_i\}$.

Proof. First consider $i \in [n]$. Let X be the subset of nodes in F defined as follows:

$$X = \{v \in F \mid \forall \tilde{a} \in S_v : a_i > 0\}$$

Clearly, the output gate r of F belongs to X . By the construction of the sets S_v , whenever a min node v belongs to X , both its children belong to X , and whenever a $+$ node belongs to X , at least one of its children belongs to X . We pick a set $T \subseteq X$ as follows. Include r in T . For each min node in T , include both its children in T . For each $+$ node in T , include in T one child that belongs to X (if both children are in X , choose any one arbitrarily). This gives a sub-formula of F where all leaves are labeled x_i . Replace these occurrences of x_i in F by 0 to get formula F' . It is easy to see that $S(F') = \{\tilde{a} - e_i \mid \tilde{a} \in S\}$. Hence F' computes $f - x_i$.

For $i = a_{n+1}$, the same process as above yields a subformula where each leaf is labeled by a positive constant. Subtracting 1 from the constant at each leaf in T gives the formula computing $f - 1$. \square

4.4 Graph entropy

The notion of the entropy of a graph or hypergraph, with respect to a probability distribution on its vertices, was first defined by Körner in [31]. In that and subsequent works (e.g. [11, 32, 33, 38]), equivalent characterizations of graph entropy were established and are often used now as the definition itself, see for instance [45, 46]. In this paper, we use graph entropy only with respect to the uniform distribution, and simply call it graph entropy. We use the following definition, which is exactly the definition from [46] specialised to the uniform distribution.

Definition 42. *Let G be a graph with vertex set $V(G) = \{1, \dots, n\}$. The vertex packing polytope $VP(G)$ of the graph G is the convex hull of the characteristic vectors of independent sets of G . The entropy of G is defined as*

$$H(G) = \min_{\vec{a} \in VP(G)} \sum_{i=1}^n \frac{1}{n} \log \frac{1}{a_i} .$$

It can easily be seen that $H(G)$ is a non-negative real number, and moreover, $H(G) = 0$ if and only if G has no edges. We list non-trivial properties of graph entropy that we use.

Lemma 43 ([32, 33]). *Let $F = (V, E(F))$ and $G = (V, E(G))$ be two graphs on the same vertex set. The following hold:*

1. **Monotonicity.** *If $E(F) \subseteq E(G)$, then $H(F) \leq H(G)$*
2. **Subadditivity.** *Let Q be the graph with vertex set V and edge set $E(F) \cup E(G)$. Then $H(Q) \leq H(F) + H(G)$.*

Lemma 44 (see for instance [45, 46]). *The following hold:*

1. *Let K_n be the complete graph on n vertices. Then $H(K_n) = \log n$.*
2. *Let G be a graph on n vertices, whose edges induce a bipartite graph on m (out of n) vertices. Then $H(G) \leq \frac{m}{n}$.*

4.5 Computing max over \mathbb{N}

It is intuitively clear that no $(\min, +)$ formula can compute max. A formal proof using Proposition [39](#) appears below.

Theorem 45. *For $n \geq 2$, no $(\min, +)$ formula over \mathbb{N} can compute $\max\{x_1, \dots, x_n\}$.*

Proof. This theorem can be proved in multiple ways. One way to do so is by applying a carefully chosen restriction to the variables, and by showing that the $(\min, +)$ formula does not compute the correct value of max on this restriction. We now give the details.

Suppose, to the contrary, some formula C computes max. Then its restriction D to $x_1 = X, x_2 = Y, x_3 = x_4 = \dots = x_n = 0$, correctly computes $\max\{X, Y\}$. Since all leaves of D are labeled from $\{x_1, x_2\} \cup \mathbb{N}$, the set $S(D)$ is a set of triples. Let $S \subseteq \mathbb{N}^3$ be this set. For all $X, Y \in \mathbb{N}$, $\max\{X, Y\}$ equals $E(X, Y) = \min\{AX + BY + C \mid (A, B, C) \in S\}$.

Let K denote the maximum value taken by C in any triple in S . If for some $B, C \in \mathbb{N}$, the triple $(0, B, C)$ belongs to S , then $E(K + 1, 0) \leq C \leq K < K + 1 = \max\{0, K + 1\}$. So for all $(A, B, C) \in S$, $A \neq 0$, so $A \geq 1$. Similarly, for all $(A, B, C) \in S$, $B \geq 1$. Hence for all $(A, B, C) \in S$, $A + B \geq 2$.

Now $E(1, 1) = \min\{A + B + C \mid (A, B, C) \in S\} \geq 2 > 1 = \max\{1, 1\}$. So $E(X, Y)$ does not compute $\max(X, Y)$ correctly. \square

Remark 46. Another way to prove this theorem was given in [26](#): The idea is to use a general property of $(\min, +)$ circuits: functions $f : \mathbb{N}^n \rightarrow \mathbb{N}$ computable by $(\min, +)$ circuits follow midpoint concavity. : $f(2u + 2v) \geq 2f(u) + 2f(v)$ holds for all $u, v \in \mathbb{N}^n$, but max can be shown to *not* follow midpoint concavity by choosing an appropriate substitution to the variables. For instance, say $u = (0, 1)$ and $v = (1, 0)$. Then $f(2u + 2v) = \max(2, 2) = 2$. But $2f(u) + 2f(v) = 2 + 2 = 4$.

However, if we also allow the subtraction operation at internal nodes, it

is very easy to compute the maximum in linear size; $\max(x_1, \dots, x_n) = -\min\{-x_1, -x_2, \dots, -x_n\}$. Here $-a$ is implemented as $0 - a$, and if we allow only variables, not constants, at leaves, we can compute $-a$ as $(x_1 - x_1) - a$.

Thus the subtraction operation adds significant power. How much? Can we compute the maximum with very few subtraction gates? As mentioned before in Section 4.2, the max function can be computed as the difference of two $(\min, +)$ formulas. Equivalently, there is a $(\min, +, -)$ formula with a single $-$ gate at the root, that computes the max function. This formula is not linear in size, but it is not too big either; we show that it has size $O(n \log n)$.

4.6 Upper bounds

Theorem 47. *The function $\max\{x_1, \dots, x_n\}$ can be computed by a difference of two $(\min, +)$ formulas with total size $n + n\lceil\log n\rceil$.*

Proof. Note that $\max\{x_1, \dots, x_n\} = \text{Sum}_n - \text{MinSum}_n^{n-1}$. Lemma 48 below shows that MinSum_n^{n-1} can be computed by a formula of size $n(\lceil\log n\rceil)$. Since Sum_n can be computed by a formula of size n , the claimed upper bound for max follows. \square

Lemma 48. *The function MinSum_n^{n-1} can be computed by a $(\min, +)$ formula of size $n(\lceil\log n\rceil)$.*

Proof. Let $m' = \lfloor n/2 \rfloor$, $m'' = \lceil n/2 \rceil$, Let X, X_l, X_r denote the sets of variables $\{x_1, \dots, x_n\}, \{x_1, \dots, x_{m'}\}, \{x_{m'+1}, \dots, x_n\}$. Note that $|X_l| = m', |X_r| = m'', m' + m'' = n$. Let p denote $\lceil\log n\rceil$. Note that $\lceil\log m'\rceil = \lceil\log m''\rceil = p - 1$.

To compute MinSum_n^{n-1} on X , we recursively compute $\text{Sum}_{m'}$ and $\text{MinSum}_{m'}^{m'-1}$ on X_l and $\text{Sum}_{m''}$ and $\text{MinSum}_{m''}^{m''-1}$ on X_r . Now $\text{MinSum}_n^{n-1}(X)$ can be computed as

$$\min \left\{ \text{Sum}_{m'}(X_l) + \text{MinSum}_{m''}^{m''-1}(X_r), \quad \text{MinSum}_{m'}^{m'-1}(X_l) + \text{Sum}_{m''}(X_r) \right\}$$

For the sub-expressions appearing above, the sizes are as claimed by induction. Thus

the number of leaves in the resulting formula is given by $m' + m''(p - 1) + m'(p - 1) + m'' = np$ as claimed. \square

Remark 49. A straightforward generalisation of this approach allows us to compute MinSum_n^{n-k} by formulas of size $n(\lceil \log n \rceil)^k$ for $1 \leq k < n$, and MinSum_n^k by formulas of size $n(\lceil \log n \rceil)^{k-1}$ for $1 \leq k < n$. But these are not the right bounds in general. For instance, for $k \in O(1)$, it is known from constructions in [16] that MinSum_n^k has $(\min, +)$ formulas of size $O(n \log n)$. (The constructions there are for monotone boolean formulas but hold for $(\min, +)$ and monotone arithmetic computations too because all they use are set schemes.) For $k = 2$, our formula above has the same size as that of [16], and is essentially the same formula, presented differently.

Similarly, for $k = n/2$, the recursive construction described above seems to need exponential size $((\log n)^{n/2})$. But this is because we count inefficiently. If we instead consider the depth of the constructed formula, we see that it is $O(\log^2 n)$, and hence the formula has at most quasi-polynomial $2^{O(\log^2 n)}$ size.

In the rest of this chapter, our goal is to prove a matching lower bound for the max function. Note that the constructions in Theorem 47 and Lemma 48 yield formulas that do not use constants at any leaves. Intuitively, it is clear that if a formula computes the maximum correctly for all natural numbers, then constants cannot help. So the lower bound should hold even in the presence of constants, and indeed our lower bound does hold even if constants are allowed.

4.7 The main lower bound

In this section, we prove the following theorem:

Theorem 50. *Let F_1, F_2 be $(\min, +)$ formulas over \mathbb{N} such that $F_1 - F_2 = \max(x_1, \dots, x_n)$. Then $L(F_1) \geq n$, and $L(F_2) \geq n \log n$.*

If F_1 and F_2 actually compute Sum_n and MinSum_n^{n-1} , then the lower bound on $L(F_1)$ is obvious, and the lower bound on $L(F_2)$ too seems “morally” clear since it holds

for monotone Boolean formulas computing the threshold function Th_n^{n-1} (see [21]).

However, the given F_1, F_2 may not be of this form at all. Also if constants are allowed at leaves of the formula, reasoning becomes a bit messy.

Our proof proceeds as follows: we first transform F_1 and F_2 over a series of steps to formulas G_1 and G_2 no larger than F_1 and F_2 , such that $G_1 - G_2$ equals $F_1 - F_2$ and hence still computes \max , and G_1 and G_2 have some nice properties. These properties immediately imply that $L(F_1) \geq L(G_1) \geq n$. We further transform G_2 to a constant-free formula H no larger than G_2 . We then define a measure for functions computable by constant-free $(\min, +)$ formulas, relate this measure to formula size, and use the properties of G_2 and H to show that the function h computed by H has large measure and large formula size.

Transformation 1: For $b \in \{1, 2\}$, let S_b denote the set $S(F_b)$. For $i \in [n + 1]$, let A_i be the minimum value appearing in the i th coordinate in any tuple in $S_1 \cup S_2$. Let \tilde{A} denote the tuple $(A_1, \dots, A_n, A_{n+1})$. By repeatedly invoking Proposition 41, we obtain formulas G_b computing $F_b - \langle \tilde{A} \cdot \tilde{x} \rangle$, with $L(G_b) \leq L(F_b)$. For $b \in \{1, 2\}$, let T_b denote the set $S(G_b)$.

We now establish the following properties of G_1 and G_2 .

Lemma 51. *Let F_1, F_2 be $(\min, +)$ formulas such that $F_1 - F_2$ computes \max over \mathbb{N} . Let G_1, G_2 be obtained as described above. Then*

1. $L(G_1) \leq L(F_1), L(G_2) \leq L(F_2)$,
2. $\max(X) = F_1 - F_2 = G_1 - G_2$,
3. For every $i \in [n]$, for every $\tilde{a} \in T_1$, $a_i > 0$. Hence $L(G_1) \geq n$.
4. For every $i \in [n]$, there exists $\tilde{a} \in T_2$, $a_i = 0$.
5. There exist $\tilde{a} \in T_1, \tilde{b} \in T_2$, $a_{n+1} = b_{n+1} = 0$.
6. For every $i, j \in [n]$ with $i \neq j$, for every $\tilde{a} \in T_2$, $a_i + a_j > 0$.

Proof. 1. This follows from Proposition [41](#).

2. Obvious; we decrease F_1 and F_2 by the same amount to get G_1 and G_2 respectively, so the difference remains the same.
3. Suppose for some $\tilde{a} \in T_1$ and for some $i \in [n]$, $a_i = 0$. Consider the input assignment \tilde{d} where $d_i = 1 + a_{n+1}$ and $d_j = 0$ for $j \in [n] \setminus \{i\}$. Then $\max\{d_1, \dots, d_n\} = 1 + a_{n+1}$. However, $\langle \tilde{a} \cdot \tilde{d} \rangle = a_{n+1}$. Therefore on input \tilde{d} , $G_1(\tilde{d}) \leq a_{n+1}$. Since $G_2 \geq 0$ on all assignments, we get $G_1(\tilde{d}) - G_2(\tilde{d}) \leq a_{n+1} < \max(\tilde{d})$, contradicting the assumption that $G_1 - G_2$ computes \max .
4. This follows from the previous point and the choice of A_i for each i .
5. From the choice of A_{n+1} , we know that there is an \tilde{a} in $T_1 \cup T_2$ with $a_{n+1} = 0$. Suppose there is such a tuple in exactly one of the sets T_1, T_2 . Then exactly one of $G_1(\tilde{0}), G_2(\tilde{0})$ equals 0, and so $G_1 - G_2$ does not compute $\max(\tilde{0})$.
6. Suppose to the contrary, some $\tilde{a} \in T_2$ has $a_i = a_j = 0$. Consider the input assignment \tilde{d} where $d_i = d_j = 1 + a_{n+1}$ and $d_k = 0$ for $k \in [n] \setminus \{i, j\}$. Then $\max\{d_1, \dots, d_n\} = 1 + a_{n+1}$. Since every x_k figures in every tuple of T_1 , $G_1(\tilde{d}) \geq d_i + d_j = 2a_{n+1} + 2$. But $G_2(\tilde{d}) \leq a_{n+1}$. Hence $G_1(\tilde{d}) - G_2(\tilde{d})$ does not compute $\max(\tilde{d})$.

□

We have already shown above that $L(F_1) \geq L(G_1) \geq n$. Now the more tricky part: we need to lower bound $L(G_2)$. Note that property 3 shows that F_1 and G_1 must be computing something of the form $\text{Sum}_n + F'_1$, or $\text{Sum}_n + G'_1$, respectively. If G'_1 were to be 0, we know that G_2 must be MinSum_n^{n-1} . However, G'_1 may have been carefully chosen to make the computation of $\text{MinSum}_n^{n-1} + G'_1$ easy. We need to rule out this possibility.

Transformation 2: Let H' be the formula obtained by simply replacing every constant in G_2 by 0. Let H be the constant-free formula obtained from H' by eliminating the zeroes, repeatedly replacing $0 + A$ by A , $\min\{0, A\}$ by 0. Let h be the function computed by H . Then, $L_{cf}(h) \leq L(H) \leq L(H') = L(G_2) \leq L(F_2)$. It thus suffices to show that $L_{cf}(h) \geq n \log n$. To this end, we define a complexity measure μ , relate it to constant-free formula size, and show that it is large for the function h .

Definition 52. For an n -variate function f computable by a constant-free $(\min, +)$ formula, we define

$$(f)_1 = \{i \mid f(e_i) \geq 1, f(0) = 0\}.$$

$$(f)_2 = \{(i, j) \mid f(e_i + e_j) \geq 1, f(e_i) = 0, f(e_j) = 0\}.$$

We define $G(f)$ to be the graph whose vertex set is $[n]$ and edge set is $(f)_2$.

The measure μ for function f is defined as follows:

$$\mu = \frac{|(f)_1|}{n} + H(G(f))$$

The following lemma relates $\mu(f)$ with $L(f)$. This relation has been used before, see for instance [\[38\]](#) for applications to monotone Boolean circuits. Since we have not seen an application in the setting of $(\min, +)$ formulas, we (re-)prove this in detail here; however, it is really the same proof.

Lemma 53. Let f be an n -variate function computable by a constant-free $(\min, +)$ formula. Then $L_{cf}(f) \geq n \cdot \mu(f)$.

Proof. The proof is by induction on the depth of a witnessing formula F that computes f and has $L_{cf}(F) = L_{cf}(f)$.

Base case: F is an input variable, say x_i . Then $(f)_1 = \{x_i\}$, and $G(f)$ is the empty graph, so $\mu(f) = \frac{1}{n}$. Hence $1 = L_{cf}(f) = n\mu(f)$.

Inductive step: F is either $F' + F''$ or $\min\{F', F''\}$ for some formulas F', F'' computing functions f', f'' respectively. Since F is an optimal-size formula for f , F' and F'' are optimal-size formulas for f' and f'' as well. So $L_{cf}(f) = L(F) = L(F') + L(F'') = L_{cf}(f') + L_{cf}(f'')$.

Case a. $F = F' + F''$. Then $(f)_1 = (f')_1 \cup (f'')_1$ and $G(f) \subseteq G(f') \cup G(f'')$. Hence,

$$\begin{aligned}
\mu(f) &\leq \frac{|(f')_1 \cup (f'')_1|}{n} + H(G(f') \cup G(f'')) && \text{(Lemma 43)} \\
&\leq \frac{|(f')_1|}{n} + \frac{|(f'')_1|}{n} + H(G(f')) + H(G(f'')) && \text{(Lemma 43)} \\
&= \mu(f') + \mu(f'') \\
&\leq \frac{1}{n} \cdot L_{cf}(f') + \frac{1}{n} \cdot L_{cf}(f'') && \text{(Induction)} \\
&= \frac{1}{n} \cdot L_{cf}(f) && (L_{cf}(f) = L_{cf}(f') + L_{cf}(f''))
\end{aligned}$$

Case b. $F = \min(F', F'')$. Let $(f')_1 = A$ and $(f'')_1 = B$. Then $(f)_1 = A \cap B$ and $G(f) \subseteq G(f') \cup G(f'') \cup G(A \setminus B, B \setminus A)$. Here, $G(P, Q)$ denotes the bipartite graph G with parts P and Q . Hence,

$$\begin{aligned}
\mu(f) &\leq \frac{1}{n}(|A \cap B|) + H(G(f') \cup G(f'') \cup G(A \setminus B, B \setminus A)) && \text{(Lemma 43)} \\
&\leq \frac{1}{n}(|A \cap B|) + H(G(f')) + H(G(f'')) + H(G(A \setminus B, B \setminus A)) && \text{(Lemma 43)} \\
&\leq \frac{1}{n}(|A \cap B|) + H(G(f')) + H(G(f'')) + \frac{1}{n}(|A \setminus B| + |B \setminus A|) && \text{(Lemma 44)} \\
&\leq \frac{1}{n}(|A| + |B|) + H(G(f')) + H(G(f'')) \\
&= \mu(f') + \mu(f'') \\
&\leq \frac{1}{n} \cdot L_{cf}(f') + \frac{1}{n} \cdot L_{cf}(f'') && \text{(Induction)} \\
&= \frac{1}{n} \cdot L_{cf}(f) && (L_{cf}(f) = L_{cf}(f') + L_{cf}(f''))
\end{aligned}$$

Hence, $\mu(f) \leq \frac{1}{n} \cdot L_{cf}(f)$. □

Using this measure, we can now show the required lower bound.

Lemma 54. *For the function h obtained after Transformation 2, $\mu(h) \geq \log n$.*

Proof. Recall that we replaced constants in G_2 by 0 to get H' , then eliminated the 0s to get constant-free H computing h . By Proposition [40](#), we know that $S(H') = \{\tilde{b} \mid b_{n+1} = 0, \exists \tilde{a} \in T_2, a_i = b_i \forall i \in [n]\}$ and that $h = \min\{\tilde{x} \cdot \tilde{b} \mid \tilde{b} \in S(H')\}$.

From item 4 in Lemma [51](#), it follows that $(h)_1 = \emptyset$. (For every i , there is a $\tilde{b} \in S(H')$ with $b_i = 0$. So $h(e_i) \leq \langle e_i \cdot \tilde{b} \rangle = 0$.)

Since $(h)_1$ is empty, $(i, j) \in G(h)$ exactly when $h(e_i + e_j) \geq 1$. From item 6 in Lemma [51](#), it follows that every pair (i, j) is in $G(h)$. Thus $G(h)$ is the complete graph K_n .

From Lemma [44](#) we conclude that $\mu(h) = \log n$. □

Lemmas [53](#) and [54](#) imply that $L_{cf}(h) \geq n \log n$. Since $L_{cf}(h) \leq L(H) \leq L(H') = L(G_2) \leq L(F_2)$, we conclude that $L(F_2) \geq n \log n$.

This completes the proof of Theorem [50](#).

A major ingredient in this proof is using the measure μ . This yields lower bounds for constant-free formulas. For functions computable in a constant-free manner, it is hard to see how constants can help. However, to transfer a lower bound on $L_{cf}(f)$ to a lower bound on $L(f)$, this idea of “constants cannot help” needs to be formalized. The transformations described before we define μ do precisely this.

For the MinSum_n^{n-1} function, applying the measure technique immediately yields the lower bound $L_{cf}(\text{MinSum}_n^{n-1}) \geq n \log n$. Transferring this lower bound to formulas with constants is a corollary of our main result, and with it we see that the upper bound from Lemma [48](#) is tight for MinSum_n^{n-1} .

Corollary 55. *Any $(\min, +)$ formula computing MinSum_n^{n-1} over \mathbb{N} must have size at least $n \log n$.*

Proof. Let F be any formula computing MinSum_n^{n-1} . Applying Theorem [50](#) to $F_1 = x_1 + \dots + x_n$ and $F_2 = F$, we obtain $L(F) \geq n \log n$. □

It is worth noting that this lower bound for $(\min, +)$ formulas computing MinSum_n^{n-1}

holds in the presence of ∞ , and also holds over integers, that is over the semiring Min^- .

Corollary 56. *Any $(\min, +)$ formula computing MinSum_n^{n-1} over $\mathbb{Z} \cup \{\infty\}$ must have size at least $n \log n$.*

Proof. Consider any formula F computing MinSum_n^{n-1} . If all constants appearing at any of the leaves are finite and non-negative, then Corollary 55 already tells us that F must have size at least $n \log n$, otherwise it will err on some inputs with no negative values or ∞ . If some leaf is labeled by the constant ∞ , we can remove such constants through repeated applications of the rules $\min(\infty, A) = A$, $\infty + A = \infty$. It remains to show that negative constants cannot help.

Consider the set $S(F)$ as in Definition 38. Here is an easy-to-see property: For any $\tilde{a} \in S(F)$, $a_{n+1} \geq 0$. This is because $F(\tilde{0}) = \text{MinSum}_n^{n-1}(0, 0, \dots, 0) = 0$. But $F(\tilde{0}) = \min\{a_{n+1} \mid \tilde{a} \in S(F)\}$, so this minimum is 0. This also shows that for at least one $\tilde{a} \in S(F)$, $a_{n+1} = 0$.

Apply Proposition 40 to get formulas G and H . (Replace all constants at leaves of F by 0 to get G , then eliminate the 0s to get H .) Let g, h be the function computed by G, H respectively. Then $g = h$. Also $L_{cf}(h) \leq L(H) \leq L(G) = L(F)$. So it suffices to lower-bound $L_{cf}(h)$.

By the property of $S(F)$ described above, for every $\tilde{x} \in \mathbb{N}^n$, $0 \leq G(\tilde{x}) \leq F(\tilde{x})$.

Now note that for all $i \in [n]$, $F(e_i) = 0$, and hence $G(e_i) = 0$. For all $i, j \in [n]$ with $i \neq j$, $F(e_i + e_j) = 1$, and hence $0 \leq G(e_i + e_j) \leq 1$. We can rule out 0 as follows. Suppose $G(e_i + e_j) = 0$. Then there exists an $\tilde{a} \in S(F)$ with $a_i = a_j = 0$; let this a_{n+1} be $c \geq 0$. Now $F((c+1)(e_i + e_j)) \leq (c+1)(e_i + e_j) \cdot \tilde{a} = c$, but $\text{MinSum}_n^{n-1}((c+1)(e_i + e_j)) = c+1$, a contradiction. So for all $i \neq j$, $G(e_i + e_j) = 1$. It now follows from Lemma 44 that $\mu(g) = \log n$. Since $h = g$, Lemma 53 implies that $L_{cf}(h) \geq n \log n$. \square

A natural question to ask is whether the main lower bound can be proved without

using graph entropy. Note that a structural property of $(\min, +)$ circuits (and formulas) implies that lower bounds on the monotone boolean circuit/formula size remain valid also for $(\min, +)$ circuits/formulas, even when only non-negative integer weights are allowed. So, the lower bound for MinSum_n^{n-1} can be alternatively derived from lower bounds on the monotone formula complexity of the threshold-2 function. The details of this argument can be found in [26]. However, the proof of the lower bound for the threshold-2 function also makes use of graph entropy.

4.8 The Monus operation

In general, over a monoid $(S, +)$, the operation of minus is not defined. If the set of monoid elements is totosed under this operation. This is why we considered the setting above where it is “required” that whenever the minus operation is used, it indeed yields a non-negative value. This is a semantic condition on a formula with minus gates. However, there is also a syntactic way of defining subtraction in totally ordered monoids, via the monus operation, denoted \ominus . For all a, b , $a \ominus b$ is simply the smallest c such that $a \leq b + c$. Over the monoid $(\mathbb{N}, +)$ (this monoid sits inside the Min semiring), it amounts to this: for all $a, b \in \mathbb{N}$, $a \ominus b$ equals $a - b$ if $a \geq b$, otherwise it equals 0. That is, $a \ominus b = \max\{0, a - b\}$. As another example, over the monoid (\mathbb{N}^+, \times) , the above definition of the monus operation as an inverse of \times gives $a \ominus b = \lceil \frac{a}{b} \rceil$.

Since \max cannot be computed within the Min semiring, we augmented it with minus. We could also have augmented it with monus instead of minus. Notice that both \min and \max are easily expressible using monus:

$$\min(a, b) = a \ominus (a \ominus b); \quad \max(a, b) = (a \ominus b) + b$$

Thus any circuit with \min , \max and $+$ gates can be transformed to a $(\ominus, +)$ circuit with just a doubling of size. However, for formulas, the cost of replacing \min and

max by \ominus could be more.

Let us consider just the maximum of n variables, as before. Again, with no restriction on monus gates, max can be computed by a linear-sized formula using the identity $\max(a, b) = (a \ominus b) + b$ recursively: $\max(x_1, x_2, \dots, x_n) = (\max(x_1, x_2, \dots, x_{n-1}) \ominus x_n) + x_n$. Unfolding this recursive construction yields a formula of size $2n - 1$. But it uses many monus gates. If we allow only one monus gate, at the top, then there is no difference between monus and minus; thus we have linear-sized circuits, $n + n \log n$ size formulas, and our lower bound for the difference of $(\min, +)$ formulas (Theorem [50](#)) continues to hold.

We show that in general, one \ominus gate always suffices; any $(\min, +, \ominus)$ formula can be equivalently computed by a $(\min, +, \ominus)$ formula with a single \ominus gate at the top. However, this transformation comes at some expense in size. The blow-up is linear for circuits but can be substantial for formulas.

Proposition 57. *Let F be any $(\min, +, \ominus)$ formula, computing a function f . Then there are $(\min, +)$ formulas F_1, F_2 such that $f = F_1 \ominus F_2$.*

Proof. We prove this by induction on the depth of F .

For the base case at depth 0, we just set $F_1 = F$ and $F_2 = 0$.

Let $F = G_1 \circ G_2$ where $\circ \in \{\min, +, \ominus\}$. Inductively, assume that G_1 and G_2 are already in the desired form; i.e. each of them has a single \ominus gate at the top. Let $G_1 = x \ominus y$ and $G_2 = z \ominus w$, where x, y, z, w are all $(\min, +)$ formulae. The expression $F_1 \ominus F_2$ in each of the three cases below can be verified to be equivalent to F .

F	F_1	F_2
$G_1 + G_2$	$x + z$	$\min(x, y) + \min(z, w)$
$\min(G_1, G_2)$	$\min(y + z, x + \min(z, w))$	$y + \min(z, w)$
$G_1 \ominus G_2$	$x + \min(z, w)$	$y + z$

Note that in the last case, although F already has a \ominus gate at the top in this case, a transformation is still needed since G_1 and G_2 also have \ominus gates at the top and

we want a single \ominus gate. □

4.9 Discussion

Our results hold when variables take values from \mathbb{N} . In the standard $(\min, +)$ semiring, the value ∞ is also allowed, since it serves as the identity for the min operation. The proof of our main result Theorem [50](#) does not carry over to this setting. The main stumbling block is the removal of the “common” part of $S(F)$. However, if we allow ∞ as a value that a variable can take, but not as a constant appearing at a leaf, then the lower bound proof goes through. However, the upper bound no longer works; while taking a difference, what is $\infty - \infty$? So the question remains: how can we compute \max over $\mathbb{N} \cup \{\infty\}$ as the difference of $(\min, +)$ formulas? Note that the monus formulation for \max still works, since $\infty \ominus a = \infty$ for any $a < \infty$.

The lower bound method uses graph entropy which is always bounded above by $\log n$. Thus this method cannot give a lower bound larger than $n \log n$. It would be interesting to obtain a modified technique that can show that all the upper bounds in Theorem [47](#) and Lemma [48](#) and Remark [49](#) are tight. It would also be interesting to find a direct combinatorial proof of our lower bound result, without using graph entropy. Note that [26](#) shows how to get this lower bound by transferring the lower bound for monotone boolean formulas for a related function. The proof of the corresponding lower bound for monotone boolean formulas again uses graph entropy.

Chapter 5

Computing shortest paths via bounded depth $(\min, +)$ formulas

5.1 Highlights

In this chapter, we present our results on lower bounds for *bounded depth $(\min, +)$ formulas*. We first define the problem of proving lower bounds for computing shortest paths in the setting of bounded depth $(\min, +)$ formulas and also provide motivation for studying this problem. We then discuss some related work and finally our results on this problem. Our main results are the following:

1. Any depth 3 circuit(or formula) for computing shortest path must have size at least $2^{\Omega(n \log n)}$ (Theorem [65](#)).
2. If F is a depth $2d$ formula for SHORTEST-PATH, where all $+$ gates except those in the bottom level have fanin at most k , then F must have size at least

$$\exp\left(\Omega\left(\frac{n \log n}{k^{d-1}}\right)\right).$$

(Theorem [67](#)).

3. For natural numbers n, r, k , let F be the smallest depth-4 formula that solves

SHORTEST-PATH on n -vertex graphs, and where at most r of the $+$ gates at the second level have fan-in exceeding k . Then F must have size at least

$$\exp\left(\Omega\left(\frac{n}{2^r} \log \frac{n}{2^r}\right)\right).$$

(Theorem [70](#)).

Organization of chapter

The rest of this chapter is organized as follows. In Section [5.2.1](#), we define our problem of interest, that is, the shortest paths problem. In Section [5.2.2](#), we provide some motivation. This is followed by upper bounds in Section [5.2.3](#) and some lower bounds which follow easily from previous work in Section [5.2.4](#). Then in Section [5.3](#), we discuss some new lower bounds and finally conclude in Section [5.4](#).

5.2 Introduction

5.2.1 The Shortest Path problem

Let G be a directed graph on a set of n vertices. Edges are labeled with costs that are non-negative and integer-valued. The cost of a path is the total cost of all edges in the path. For designated source vertex s and target vertex t , the SHORTEST-PATH $_n$ problem is to find the minimum cost of a path from s to t . (The subscript n indicates the graph size; we drop it when implicit from context.)

To compute SHORTEST-PATH by $(\min, +)$ circuits or formulas, we assume that G is the complete directed graph with vertex set $[n]$, and for each $i, j \in [n]$ with $i \neq j$, the variable $x_{i,j}$ is the cost of the edge directed from vertex i to vertex j . All variables take values in the set $\mathbb{N} \cup \{\infty\}$. While solving SHORTEST-PATH on any input graph, we will set $x_{i,j}$ to be the actual cost of the edge from i to j . We will set $x_{i,j}$ to ∞ for edges absent in G . We may also assume that there are variables $x_{i,i}$ for $i \in [n]$;

these variables are all set to 0. With these conventions, the following expression is the desired SHORTEST-PATH value.

$$\text{SHORTEST-PATH} = \min \{ \text{cost}(\rho) \mid \rho \text{ is a simple } s\text{-to-}t \text{ path} \},$$

where

$$\text{cost}(\rho) = \sum_{\langle i,j \rangle \in \rho} x_{i,j}.$$

We denote by REACH_n the decision problem of deciding whether an n -vertex graph has an s -to- t path.

5.2.2 Motivation

Recall the definition of $(\min, +)$ circuits from Chapter 3. This model captures the complexity of “pure” dynamic programming algorithms; see for instance [23–25].

A naive approach towards solving SHORTEST-PATH using $(\min, +)$ circuits would be as follows: Compute the weights of all possible paths from 1 to n and take the minimum of these weights. This yields a $(\min, +)$ circuit of size $2^{O(n \log n)}$ size. For depth 2 $(\min, +)$ circuits, it is easy to see that this naive approach is indeed the best possible. We show that this is also the case for depth 3 circuits (Theorem [65]). We would expect the lower bounds to degrade as we allow more depth, and the question we are interested in is, how fast do they degrade? We provide partial answers to this question, exploring restricted cases of $(\min, +)$ formulas (circuits of fan-out 1). We study restrictions of two types. In the first restriction, except at the bottom-most level, $+$ gates do not have very large fan-in. (Note that since paths in an n -vertex graph have at most $n - 1$ edges, the fanin of useful $+$ gates will not exceed n .) Our lower bound is parameterized by the depth d and the permitted $+$ fanin k (Theorem [67]). For the depth-4 case, the lower bound is tight when $k = O(1)$ and also when $k = O(\sqrt{n})$. In the second restriction, which applies only to depth-4 formulas, most $+$ gates just below the top gate have fan-in 2. Our lower bound here

is parameterized by the the number of $+$ gates with fan-in exceeding 2 (Theorem [70](#)). Note that any constant-depth circuit can be simulated by a formula of the same depth, with at most a polynomial blow-up in size. Therefore, our result also implies an exponential lower bound for the corresponding subclass of constant depth $(\min, +)$ circuits.

Background

Many known algorithms for solving SHORTEST-PATH are essentially recursively constructed $(\min, +)$ circuits. For instance, the classical dynamic programming algorithm by Bellman and Ford [5, 15](#) gives a bounded fan-in circuit of $O(n^3)$ size and depth $\Theta(n)$. Whether $\Omega(n^3)$ is necessary is still open. However the Bellman-Ford algorithms produce skew circuits, and for skew circuits, this bound is shown in [27](#) to be optimal. A divide-and-conquer approach gives a bounded fan-in circuit of $\text{poly}(n)$ ($O(n^4)$) size and depth $\Theta(\log^2 n)$.

A natural question to ask is whether one can prove strong size lower bounds for bounded depth $(\min, +)$ circuits or formulas.

It is known that over the Boolean semiring, any bounded fan-in monotone (\vee, \wedge) circuit for REACH must have depth $\Omega(\log^2 n)$ [29](#). Using a natural mapping from $(\min, +)$ semiring to the boolean semiring, this result also implies that any bounded fan-in $(\min, +)$ circuit for STCON must have $\Omega(\log^2 n)$ depth, no matter what size. The divide-and-conquer approach shows that this depth lower bound is tight.

In this work, we consider the alternation depth of $(\min, +)$ circuits. This corresponds to allowing semi-unbounded fan-in and even unbounded fan-in in some cases. In this setting, exponential lower bounds are easy to prove. One way to do so is to use the reduction (via projections) from parity to REACH, and use known lower bounds for (non-monotone) circuits for parity [18](#); see Proposition [58](#). We are looking for lower bounds better than those obtained this way.

In [10](#), such small-depth lower bounds are obtained for the decision version of "short

distance connectivity”: is there a path using at most k edges? These lower bounds can also be transferred to $(\min, +)$ circuits computing the corresponding optimisation problem: Compute the weight of the shortest path which uses at most k edges.

5.2.3 Known upper bounds

Viewed as a polynomial over the semiring Min , the SHORTEST-PATH polynomial has the set of monomials

$$\text{Mon}(\text{SHORTEST-PATH}) = \{\text{cost}(\rho) \mid \rho \text{ is a simple } s\text{-to-}t \text{ path}\}.$$

It is known that any circuit producing the SHORTEST-PATH polynomial must be exponentially large [24]. However, to compute this polynomial, it suffices to design a circuit producing a polynomial whose lower envelope has exactly the monomials in $\text{Mon}(\text{SHORTEST-PATH})$, and this is a considerably easier task.

Incremental dynamic programming, extending sub-paths by a single edge at a time, gives a bounded fanin circuit of $O(n^3)$ size and depth $\Theta(n)$.

Dynamic programming, merging roughly equal-length sub-paths (equivalently, dividing each path roughly mid-way), gives a bounded fanin circuit of $\text{poly}(n)$ ($O(n^4)$) size and depth $\Theta(\log^2 n)$.

Here we are concerned with alternation depth, or equivalently, unbounded fanin circuits and formulas. Dynamic programming, where r sub-paths are merged in each merge step for some parameter r , gives a recurrence as follows. Let $f(i, j, \ell)$ denote the minimum cost of a path from i to j amongst all paths with at most ℓ edges. Our goal is to compute $f(s, t, n - 1)$, we have $f(i, i, \ell) = 0$ for all ℓ , and we

have the following expressions for $i \neq j$:

$$\begin{aligned} f(i, j, 0) &= \infty \\ f(i, j, 1) &= x_{i,j} \\ f(i, j, \ell) &= \min_{k_1, \dots, k_{r-1}} \left\{ f\left(i, k_1, \frac{\ell}{r}\right) + f\left(k_1, k_2, \frac{\ell}{r}\right) + \dots + f\left(k_{r-1}, j, \frac{\ell}{r}\right) \right\} \end{aligned}$$

The depth of recursion is given by $p = \frac{\log n}{\log r}$. Each level of recursion corresponds to a layer of min gates followed by a layer of + gates. Thus the corresponding circuit has depth $d = 2p$. Conversely, to get a (min, +) circuit of depth $d = 2p$, it suffices to take $r = n^{\frac{2}{d}}$. The fanin of the min gates is at most n^{r-1} whereas the fanin of + gates is at most r . It is easily verified that this gives rise to a circuit of size $\exp(O(n^{\frac{2}{d}} \log n))$, or a formula of size $\exp(O(dn^{\frac{2}{d}} \log n))$. In particular, the depth-2 formula is of size $\exp(O(n \log n))$.

5.2.4 Lower bounds implied from known work

The SHORTEST-PATH polynomial, interpreted over the Boolean ring B , decides REACH. Hence $B(\text{SHORTEST-PATH}) \leq \text{Min}(\text{SHORTEST-PATH})$; any monotone Boolean circuit size lower bound for REACH is also a lower bound for $\text{Min}(\text{SHORTEST-PATH})$.

For bounded (alternation) depth, one lower bound for Boolean circuits for REACH is derived from the lower bound for circuits for parity [18]. Although this is folklore, for completeness we include a full proof here.

Proposition 58 (folklore). *Depth d Boolean circuits (and hence also monotone Boolean circuits) for REACH_{2n} must be of size $\exp(\Omega(n^{\frac{1}{d-1}}))$. Hence any depth d (min, +) circuit computing $\text{SHORTEST-PATH}_{2n}$ must have size $\exp(\Omega(n^{\frac{1}{d-1}}))$.*

Proof. Given n bits y_1, \dots, y_n , the PARITY_n function outputs 1 if an odd number of y_i 's are set to 1, and 0 otherwise. In [18] it is shown that Boolean circuits for PARITY_n with alternation depth d must have size $\exp(\Omega(n^{\frac{1}{d-1}}))$.

The PARITY_n function reduces to REACH_{2n} by projections as follows. The REACH_{2n} instance is a graph G with $2n$ vertices, and it is convenient to think of the vertex set as $\{(i, b) \mid i \in [n-1], b \in \{0, 1\}\} \cup \{(0, 0), (n, 1)\}$, with source vertex $s = (0, 0)$ and sink vertex $t = (n, 1)$. The edges of the graph are determined as follows: There is an edge from a vertex $(i-1, b)$ to (i, b) for $b \in \{0, 1\}$ if and only if $y_i = 0$. Similarly, there is an edge from $(i-1, b)$ to $(i, 1-b)$ for $b \in \{0, 1\}$ if and only if $y_i = 1$. That is, $x_{(i-1,b),(i,b)} = \bar{y}_i$, and $x_{(i-1,b),(i,1-b)} = y_i$. If $i \neq j-1$, then $x_{(i,b),(j,b')} = 0$. It is easy to see that there is a path from $(0, 0)$ to $(n, 1)$ in G if and only if $y_1 + \dots + y_n \equiv 1 \pmod{2}$. Hence any Boolean circuit for REACH_{2n} , with alternation depth d , must also have size $\exp(\Omega(n^{\frac{1}{d-1}}))$. \square

In [10], the restriction of REACH to short path lengths is studied. We denote by $\text{SHORTEST-PATH}_{n,k}$ the restriction of the SHORTEST-PATH_n polynomial to the monomial set

$$\{\text{cost}(\rho) \mid \rho \text{ is a simple } s\text{-to-}t \text{ path of length at most } k\},$$

and let $\text{REACH}_{n,k}$ denote the corresponding decision version (decide whether the graph has an s -to- t path of length at most k). In [10], the following result is shown:

Proposition 59 (Theorem 1 in [10]). *1. For any $k(n) \leq n^{1/5}$ and any $d = d(n)$,*

any depth- d circuit computing $\text{REACH}_{n,k}$ must have size $n^{\Omega(k^{1/d}/d)}$.

2. For any $k(n) \leq n$ and any $d = d(n)$, any depth- d circuit computing $\text{REACH}_{n,k}$

must have size $n^{\Omega(k^{1/5d}/d)}$.

Note that this bound applies for any Boolean circuit, not just monotone circuits. At $k = n$, it gives a lower bound of $\exp(\Omega(\frac{n^{1/5d} \log n}{d}))$ for depth- d circuits computing REACH_n . Hence

Corollary 60. *Any depth- d (min, +) circuit for SHORTEST-PATH must have size $\exp(\Omega(\frac{n^{1/5d} \log n}{d}))$.*

5.3 New Lower Bounds

The following fact is easy to verify.

Fact 61. *Let $P(n)$ denote the number of distinct st paths in the complete n -vertex directed graph. Then $P(n) = 2^{\Theta(n \log n)}$. More specifically,*

$$2((n-2)!) < P(n) < e((n-2)!).$$

For any formula F computing SHORTEST-PATH, the polynomial produced by F must have exactly the monomials of SHORTEST-PATH in its lower envelope. A direct graph-theoretic way to see this is given in the following proposition. In its proof, as well as later in this thesis, we use the notation G_ρ to denote the graph with only the edges of ρ , for any simple st path ρ .

Proposition 62. *Let F be a formula computing SHORTEST-PATH. Let p be the polynomial syntactically produced by F , and $\text{Mon}_{le}(p)$ be the set of minimal monomials of this polynomial (the lower envelope). Then $\text{Mon}_{le}(p)$ equals the set of monomials of SHORTEST-PATH, $\{\text{cost}(\rho) \mid \rho \text{ is a simple } s\text{-to-}t \text{ path}\}$.*

Proof. Let ρ be any simple st path, and let G_ρ be the graph with only the edges of ρ . On setting $x_{i,j}$ to 1 for $(i,j) \in \rho$ and to ∞ for all other edges, F should evaluate to $|\rho|$. So at least one linear form (recall, in the semiring Min, monomials are linear forms) should use only the variables from ρ (otherwise it evaluates to ∞). However, if for any such linear form, ℓ , $\text{var}(\ell)$ is a proper subset of $\text{var}(\rho)$, then some variable x_{uv} with value 1 does not appear in ℓ . Deleting edge uv from G_ρ (changing the value of x_{uv} to ∞) disconnects s and t in the resulting graph, so F should now evaluate to ∞ . But ℓ is still finite on this modified graph, a contradiction. Hence every linear form using only variables from ρ must use all variables from ρ . Since the correct value on G_ρ is ρ , at least one such linear form must use all variables from ρ exactly once, producing the monomial $\text{cost}(\rho)$. By the above argument, this linear form is

minimal, and hence in $\text{Mon}_{l_e}(p)$.

To show the other direction, let m be a monomial in $\text{Mon}_{l_e}(p)$. Consider the setting where variables in m are set to 1, and all other variables are set to ∞ ; let this be the graph H . On H , F evaluates to a finite value, so H must have an s -to- t path. Let ρ be a shortest such path. By construction, the variables on edges of ρ are all in m . Hence for the monomial $\text{cost}(\rho)$ we have the order $\text{cost}(\rho) \preceq m$. We have already proved that $\text{cost}(\rho) \in \text{Mon}_{l_e}(p)$. Since m is also in $\text{Mon}_{l_e}(p)$ ie minimal, it follows that $m = \text{cost}(\rho)$. \square

This gives us the following useful property.

Property 63. *Let F be a minimal $(\min, +)$ formula computing SHORTEST-PATH. The top gate of F must be a min gate.*

Proof. By Proposition [62](#), the polynomial produced by F must have exactly the monomials of SHORTEST-PATH in its lower envelope. One of the monomials is the single variable x_{st} , which cannot be further split by addition. If the top gate of F is a $+$ gate with more than one child (since F is minimal, it has no gates with fanin 1), then to produce this monomial, all but one of the children must return the value 0, making them redundant. \square

We start off with some simple lower bounds in the very special case when the depth is 2 or 3.

Proposition 64. *Any depth-2 formula computing SHORTEST-PATH must have size $2^{\Omega(n \log n)}$.*

Proof. Let F be a depth-2 formula computing SHORTEST-PATH. By Property [63](#), the top gate of F must be a min gate. Let this gate have ℓ children. Then the polynomial produced by F has at most ℓ monomials. Hence, by Proposition [62](#) and Fact [61](#), $\ell \geq P(n)$. \square

Theorem 65. *Any depth-3 circuit computing SHORTEST-PATH must have size $2^{\Omega(n \log n)}$.*

Proof. Let F be a depth-3 formula for SHORTEST-PATH. Let p be the polynomial p produced by F . By Proposition 62, $\text{Mon}_{\ell e}(p)$ equals $\text{Mon}(\text{SHORTEST-PATH})$, which by Fact 61 has size $P(n)$. Further, by Property 63, the top gate of F must be a min gate. Let this gate have ℓ children. We prove below that each $+$ gate can produce at most one monomial from $\text{Mon}_{\ell e}(p)$. Hence $\ell \geq P(n)$.

Consider a $+$ gate g with fanin k . Every monomial produced by g has degree k (i.e. k summands). Hence g cannot produce monomials corresponding to paths of length greater than k . In fact, it cannot even produce monomials corresponding to simple paths of length less than k – a shorter path has fewer than k variables while the monomial has exactly k summands, so at least one variable will have to appear with coefficient greater than 1, whereas the monomials $\text{cost}(\rho)$ for simple paths have 0-1 coefficients.

Suppose g produces monomials corresponding to two distinct paths $\eta \neq \rho$, both of length k . We will consider the two graphs G_ρ and G_η .

Let g_1, \dots, g_k be the children of g and let S_i be the set of children of g_i , $1 \leq i \leq k$. Since the circuit has depth 3, each element of S_i is a variable. Let $\rho = \langle i_0 = s, i_1, \dots, i_k = t \rangle$, and without loss of generality, let the variable $x_{i_{p-1}} x_{i_p} \in S_p$ for $1 \leq p \leq k$.

First, we show that for each variable x_{ab} in S_p , it must be the case that $a \in \{i_0, \dots, i_{p-1}\}$ and $b \in \{i_p, \dots, i_k\}$. To see this, consider the graph $G' = G_\rho \setminus \{(i_{p-1}, i_p)\} \cup (a, b)$. Each g_i still evaluates to 1, and hence g evaluates to k on G' . Therefore G' must be connected, which implies that $a \in \{i_0, \dots, i_{p-1}\}$ and $b \in \{i_p, \dots, i_k\}$.

Since the path η is constructed using the variables in the sets S_i , this implies that η cannot have a vertex that is not present in ρ . However, η has the same length as ρ , by assumption, so it uses all the vertices of ρ . Then it must use them in a different

order.

Let p be the smallest index where η and ρ differ. Thus the sub-path $\langle i_0, \dots, i_{p-1} \rangle$ of ρ is also a sub-path of η , and the edge $(i_{p-1}, i_p) \in \rho$, and $(i_{p-1}, i_p) \notin \eta$. Let the edge in η from S_p be (i_q, i_r) . By the argument above, $q \in \{0, \dots, p-1\}$ and $r \in \{p, \dots, k\}$, and furthermore, $r \neq q+1$. There are two cases to consider. One possibility is that $q < p-1$. Then η has two edges out of i_q , contradicting the assumption that η is a simple path. The other possibility is that $q = p-1$ but $r > p$. Then, to eventually visit vertex i_p , η must use an edge that is a “back-edge” with respect to ρ . But we have shown above that the variable sets S_i prohibit such back-edges.

Therefore such a path η does not exist. This completes the proof. \square

We now consider formulas of depth $2d$. By Property [63](#), the top gate is a min gate, and hence the gates at the lowest level are + gates. Without loss of generality, we assume that all paths from the root to the leaves are of length exactly $2d$. (If necessary, add dummy gates with fanin 1; this at most doubles the formula size.)

Let \mathcal{G} denote the set of all + gates in C except those at the leaf level. That is, a + gate is in \mathcal{G} if and only if it has as a child another gate of the formula. Let \mathcal{G}_k denote the set of gates in \mathcal{G} with fanin bounded by k .

Lemma 66. *Let F be an alternating formula, with a min gate on top, and of depth $2d$ for some $d \geq 1$. Let the polynomial syntactically produced by F have M monomials. If for some $k \in \mathbb{N}$, all + gates except those at the leaves have fanin at most k (that is, $\mathcal{G} \subseteq \mathcal{G}_k$), then $M \leq (L(F))^{k^{d-1}}$.*

Proof. The proof is by induction on d .

Base Case: $d = 1$. In this case, to syntactically produce M monomials, the top gate of F must have fanin M , and so $L(F) \geq M$.

Inductive Step: $d > 1$.

Let g_i be the + gates just below the output gate, and let $h_{i,j}$ be the min gates feeding into g_i . (Note that $j \leq k$, by assumption.) Let s_i and $s_{i,j}$ denote the leaf-sizes of

the formulas rooted at g_i and $h_{i,j}$ respectively.

$$\begin{aligned}
M &= \text{number of monomials produced by } F \\
&\leq \sum_i (\text{number of monomials produced by } g_i) \\
&\leq \sum_i \prod_j (\text{number of monomials produced by } h_{i,j}) \\
&\leq \sum_i \prod_j (s_{i,j})^{k^{d-2}} \quad (\text{by induction}) \\
&\leq \sum_i \prod_j (s_i)^{k^{d-2}} \leq \sum_i (s_i)^{k^{d-1}} \leq \left(\sum_i s_i \right)^{k^{d-1}} = (L(F))^{k^{d-1}}.
\end{aligned}$$

□

Combining Proposition [62](#) and Fact [61](#) with Lemma [66](#), we obtain the following lower bound.

Theorem 67. *If F is a depth $2d$ formula for SHORTEST-PATH, where all $+$ gates except those in the bottom level have fanin at most k , then*

$$L(F) \geq \exp \left(\Omega \left(\frac{n \log n}{k^{d-1}} \right) \right).$$

Proof. By Proposition [62](#), the polynomial $p(F)$ produced by F must have the monomials of SHORTEST-PATH as its lower envelope. By Fact [61](#), SHORTEST-PATH has $P(n)$ monomials. Lemma [66](#) bounds the number of monomials of $p(F)$ from above. Hence

$$(L(F))^{k^{d-1}} \geq \text{number of monomials of } p(F) \geq P(n) = 2^{\Omega(n \log n)},$$

giving the claimed bound on formula size.

□

Corollary 68. *Let F be a depth 4 formula for SHORTEST-PATH, where all $+$ gates*

below the top min gate have fanin at most k . Then

$$L(F) \geq \exp\left(\Omega\left(\frac{n \log n}{k}\right)\right).$$

Remark 69. 1. If $k = O(1)$, then $L(F) = 2^{\Omega(n \log n)}$. This size is achievable even with a depth-2 formula and so this bound is tight

2. If $k = O(\sqrt{n})$, then $L(F) = 2^{\Omega(\sqrt{n} \log n)}$. This size is achievable with the depth-4 formula constructed by dynamic programming with all $+$ gates having fanin $O(\sqrt{n})$ and so this bound is tight.

A special case of Corollary [68](#) is when $k = 2$. That is, each path (monomial of SHORTEST-PATH) of length more than 1 is broken at the second level into just two parts. In this case, Corollary [68](#) tells us that $2^{\Omega(n \log n)}$ size is necessary; by the remark following it, this is also sufficient. What if we relax the condition slightly, and allow a few second-level $+$ gates to have fanin more than 2? Can we get non-trivial savings in size? We explore this question next.

For natural numbers n, r, k , let $L(n, k, r)$ denote the (leaf-)size of the smallest depth-4 formula that solves SHORTEST-PATH on n -vertex graphs, and where at most r of the $+$ gates at the second level have fan-in exceeding k .

Theorem 70.

$$L(n, 2, r) \geq \exp\left(\Omega\left(\frac{n}{2^r} \log \frac{n}{2^r}\right)\right).$$

To prove this theorem, we gradually decrease r while reducing the size of the graphs handled, in Lemma [71](#) below. This works for any value of k . Finally when $k = 2$ and r has been brought down to 0, we use the bound given by Corollary [68](#), namely

$$L(n, 2, 0) = \exp(\Omega(n \log n)).$$

Lemma 71. For $r \geq 1$, $L(n, k, r) \geq L(n(\frac{k-1}{k}), k, r-1)$. In particular,

$$L(n, 2, r) \geq L\left(\frac{n}{2}, 2, r-1\right).$$

Proof. Let F be the smallest depth-4 formula solving SHORTEST-PATH on n -vertex graphs, where the number of $+$ gates at the second level with fan-in exceeding k is at most r . Let g be the $+$ gate at level 2 that has the largest fan-in. Without loss of generality, assume fan-in of g to be $q \geq k+1$, otherwise the lemma statement trivially holds. Let g_1, \dots, g_q be the children of g . Let L_i be the set of monomials produced by g_i .

For a monomial ℓ , let $E(\ell) = \{(u, v) \mid x_{uv} \in \ell\}$. By minimality of F , and Proposition [62](#), g must produce at least one monomial from $\text{Mon}(\text{SHORTEST-PATH})$, say corresponding to an st path ρ . Then there exist $\ell_i \in L_i$ such that $\cup_{i=1}^q E(\ell_i)$ gives exactly the edges of G_ρ .

Define $S_i = \{u \mid \exists v, (u, v) \in E(\ell_i)\}$. Without loss of generality, the vertex s is in the set S_i .

Claim 72. For every path η such that g produces the monomial $\text{cost}(\eta)$, and for every $i \in [q]$, the path η visits at least one vertex in S_i .

Proof. For $i = 1$, this is trivially true because the path η starts at vertex s which is in S_1 .

For some $i > 1$, suppose η avoids the set S_i . Suppose g constructs $\text{cost}(\eta)$ by using, for each $j \in [q]$, the monomial $\ell'_j \in L_j$; hence $\cup_{i=j}^q E(\ell'_j) = G_\eta$. Consider the graph $H = G_\eta \setminus E(\ell'_i) \cup E(\ell_i)$. Clearly, $g(H) < \infty$. However, since $E(\ell_i)$ is vertex-disjoint from η , H cannot have an st path, contradicting the correctness of F . \square

The above claim implies that for each $i > 1$, g does not produce any monomial corresponding to a path avoiding S_i (i.e. a path on $[n] \setminus S_i$). Thus, if we remove g from F to get F' , then F' still correctly computes SHORTEST-PATH on the vertex set $[n] \setminus S_i$. We now show that there is $i > 1$ such that $|S_i| \leq (n-2)/(q-1)$.

The set $S = \cup_{i=1}^q S_i$ contains all the vertices of ρ except t , so $|S| \leq n - 1$ and $|S \setminus \{s\}| \leq n - 2$. Further, the sets S_i are disjoint, thereby partitioning S into q parts. Among the $q - 1$ parts that do not contain s , by an averaging argument, the smallest part contains no more than $(n - 2)/(q - 1)$ vertices. Thus F' correctly computes SHORTEST-PATH on $m \geq n - (n - 2)/(q - 1)$ vertices. Since $q > k$, $m \geq n(k - 1)/k$.

□

Proof. (of Theorem 70) Using Lemma 71 r times and then Corollary 68, we get

$$L(n, 2, r) \geq L\left(\frac{n}{2}, 2, r - 1\right) \geq \dots \geq L\left(\frac{n}{2^r}, 2, 0\right) = \exp\left(\Omega\left(\frac{n}{2^r} \log \frac{n}{2^r}\right)\right).$$

□

Theorem 70 gives a non-trivial size lower bound for depth-4 formulas when at most say, $O(\log \log n)$ of the second level + gates have fanin more than 2.

5.4 Conclusion

Understanding the limits of dynamic programming is an interesting and challenging exercise. In particular, it is surprising that we do not yet know whether with bounded fan-in ($\min, +$) circuits, $\Omega(n^3)$ is necessary to compute SHORTEST-PATH. In this paper we have focussed on unbounded fan-in ($\min, +$) formulas. We still do not have a complete understanding of how to optimally exploit additional depth but we have obtained some partial results. A complete characterisation of the exact complexity of SHORTEST-PATH in this setting, parameterised by depth, as is known for Boolean circuits computing the parity function [18] remains open.

Chapter 6

Conclusion

As we mentioned in the introduction, the main aim of this thesis has been to give new techniques for proving lower bounds on restricted models of circuits, in the hope that it sheds light on more general lower bound questions. Below, we state some questions that still remain open. These questions have been mentioned at the end of each individual chapter. We state only the more important ones below.

In Chapter [2](#), we considered the model of sums of read-once formulas (ROFs). We were able to prove a strict hierarchy among sums-of-ROFs. One open question is to obtain separating polynomials with fewer variables. Also, lower bounds similar to the one we proved have turned out to be useful in designing efficient Polynomial Identity Testing (PIT) algorithms. It is an interesting question whether our lower bound can also be turned into a PIT algorithm.

In Chapter [4](#), we considered the model of difference of $(\min, +)$ formulas. We were able to prove tight lower bounds for computing the maximum among n variables using difference of $(\min, +)$ formulas. A natural question to ask is whether one can obtain tight lower bounds for computing similar functions such as the k^{th} largest element or the sum of topmost k elements. We used the technique of graph entropy which cannot yield lower bounds better than $n \log n$. As a first step, it would be interesting to obtain a direct combinatorial proof of our result which avoids the use

of graph entropy.

In Chapter [5](#), we considered bounded depth $(\min, +)$ formulas. We were able to prove tight lower bounds for bounded depth $(\min, +)$ formulas with certain fan-in restrictions for the shortest paths problem. It would be nice to obtain lower bounds for bounded depth $(\min, +)$ formulas without any fan-in restrictions.

Bibliography

- [1] Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 67–75, 2008.
- [2] Eric Allender. Arithmetic circuits and counting complexity classes. In Jan Krajíček, editor, *Complexity of Computations and Proofs*, Quaderni di Matematica Vol. 13, pages 33–72. Seconda Università di Napoli, 2004. An earlier version appeared in the Complexity Theory Column, SIGACT News 28, 4 (Dec. 1997) pp. 2-15.
- [3] Matthew Anderson, Dieter van Melkebeek, and Ilya Volkovich. Deterministic polynomial identity tests for multilinear bounded-read formulae. *Computational Complexity*, 24(4):695–776, 2015.
- [4] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.
- [5] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1956.
- [6] Daoud Bshouty and Nader H. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *Journal of Computer and System Sciences*, 56(1):112–124, 1998.

- [7] Nader H. Bshouty and Richard Cleve. Interpolating arithmetic read-once formulas in parallel. *SIAM Journal on Computing*, 27(2):401–413, 1998.
- [8] Nader H. Bshouty, Thomas R. Hancock, and Lisa Hellerstein. Learning Boolean read-once formulas over generalized bases. *Journal of Computer and System Sciences*, 50(3):521–542, 1995.
- [9] Ramya C. and B. V. Raghavendra Rao. Sum of products of read-once formulas. *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016*, pages 39:1–39:15, 2016.
- [10] Xi Chen, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Near-optimal small-depth lower bounds for small distance connectivity. *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 612–625, 2016.
- [11] Imre Csiszár, János Körner, László Lovász, Katalin Marton, and Gábor Simonyi. Entropy splitting for antiblocking corners and perfect graphs. *Combinatorica*, 10(1):27–40, 1990.
- [12] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM Journal on Computing*, 39(4):1279–1293, 2009.
- [13] Robert W Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.
- [14] Michael A. Forbes. Deterministic divisibility testing via shifted partial derivatives. *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 451–465, 2015.
- [15] Lester R Ford Jr. Network flow theory. Technical Report P-923, Rand Corporation, 1956.

- [16] J. Friedman. Constructing $o(n \log n)$ size monotone formulae for the k -th elementary symmetric polynomial of n boolean variables. *Proceedings, 25th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 506–515, 1984.
- [17] Thomas R. Hancock and Lisa Hellerstein. Learning read-once formulas over fields and extended bases. *Proceedings of the Fourth Annual Workshop on Computational Learning Theory, COLT 1991*, pages 326–336, 1991.
- [18] Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, Cambridge, MA, USA, 1987.
- [19] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.
- [20] Pavel Hrubes and Amir Yehudayoff. Homogeneous formulas and symmetric polynomials. *Computational Complexity*, 20(3):559–578, 2011.
- [21] Radhakrishnan J. Better lower bounds for monotone threshold formulas. *Journal of Computer and System Sciences*, 54:221–226, 1997.
- [22] Mark Jerrum and Marc Snir. Some exact complexity results for straight-line computations over semirings. *Journal of the ACM (JACM)*, 29(3):874–897, 1982.
- [23] Stasys Jukna. Limitations of incremental dynamic programming. *Algorithmica*, 69(2):461–492, 2014.
- [24] Stasys Jukna. Lower bounds for tropical circuits and dynamic programs. *Theory of Computing Systems*, 57(1):160–194, 2015.
- [25] Stasys Jukna. Tropical complexity, Sidon sets, and dynamic programming. *SIAM Journal on Discrete Mathematics*, 30(4):2064–2085, 2016.

- [26] Stasys Jukna. (Min, Plus) is not stronger than (And, Or). *Comment 1 on Electronic Colloquium on Computational Complexity (ECCC)*, 25:20, 2018.
- [27] Stasys Jukna and Georg Schnitger. On the optimality of Bellman-Ford-Moore shortest path algorithm. *Theoretical Computer Science*, 628:101–109, 2016.
- [28] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.
- [29] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990.
- [30] Neeraj Kayal, Pascal Koiran, Timothée Pecatte, and Chandan Saha. Lower bounds for sums of powers of low degree univariates. *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015*, pages 810–821, 2015.
- [31] János Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. *Transactions of 6th Prague Conference on Information Theory*, pages 411–425, 1973.
- [32] János Körner. Fredman-Komlós bounds and information theory. *SIAM. J. on Algebraic and Discrete Methods*, 7(4):560–570, 1986.
- [33] János Körner and Katalin Marton. New bounds for perfect hashing via information theory. *European Journal of Combinatorics*, 9(6):523–530, 1988.
- [34] Meena Mahajan, Prajakta Nimbhorkar, and Anuj Tawari. Computing the maximum using (min, +) formulas. *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017*, pages 74:1–74:11, 2017.

- [35] Meena Mahajan, Prajakta Nimbhorkar, and Anuj Tawari. Shortest path length with bounded-alternation (min, +) formulas. *International Journal of Advances in Engineering Sciences and Applied Mathematics*, 2018.
- [36] Meena Mahajan and Anuj Tawari. Sums of read-once formulas: How many summands are necessary? *Theoretical Computer Science*, 708:34–45, 2018.
- [37] Edward F Moore. *The shortest path through a maze*. Bell Telephone System., 1959.
- [38] Ilan Newman and Avi Wigderson. Lower bounds on formula size of boolean functions using hypergraph entropy. *SIAM Journal on Discrete Mathematics*, 8(4):536–542, 1995.
- [39] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *Journal of the ACM (JACM)*, 56(2), 2009.
- [40] Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Computational Complexity*, 18(2):171–207, 2009.
- [41] Amir Shpilka and Ilya Volkovich. Improved polynomial identity testing for read-once formulas. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009,*, pages 700–713, 2009.
- [42] Amir Shpilka and Ilya Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014.
- [43] Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015.
- [44] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

- [45] Gábor Simonyi. Graph entropy: A survey. *Combinatorial Optimization*, 20:399–441, 1995.
- [46] Gábor Simonyi. Perfect graphs and graph entropy: An updated survey. In *Perfect Graphs*, chapter 13, pages 293–328. John Wiley and Sons, 2001.
- [47] Volker Strassen. Berechnungen in partiellen algebren endlichen typs. *Computing*, 11(3):181–196, 1973.
- [48] Ilya Volkovich. Characterizing arithmetic read-once formulae. *ACM Transactions on Computation Theory*, 8(1):2:1–2:19, February 2016.
- [49] Stephen Warshall. A theorem on Boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12, 1962.