2016 IEEE 57th Annual Symposium on Foundations of Computer Science

Polynomial Representations of Threshold Functions and Algorithmic Applications

Josh Alman[*], Timothy M. Chan[†] and Ryan Williams[*]
[*]Computer Science Department,
Stanford University,
{jalman,rrw}@cs.stanford.edu.
[†]Cheriton School of Computer Science,
University of Waterloo,
tmchan@uwaterloo.ca.

***Abstract*—We design new polynomials for representing threshold functions in three different regimes: *probabilistic polynomials* of low degree, which need far less randomness than previous constructions, *polynomial threshold functions* (PTFs) with "nice" threshold behavior and degree almost as low as the probabilistic polynomials, and a new notion of *probabilistic PTFs* where we combine the above techniques to achieve even lower degree with similar "nice" threshold behavior. Utilizing these polynomial constructions, we design faster algorithms for a variety of problems:**

- ***Offline Hamming Nearest (and Furthest) Neighbors:* Given n red and n blue points in d-dimensional Hamming space for $d = c\log n$, we can find an (exact) nearest (or furthest) blue neighbor for every red point in randomized time $n^{2-1/O(\sqrt{c}\log^{2/3} c)}$ or deterministic time $n^{2-1/O(c\log^2 c)}$. These improve on a randomized $n^{2-1/O(c\log^2 c)}$ bound by Alman and Williams (FOCS'15), and also lead to faster MAX-SAT algorithms for sparse CNFs.**
- ***Offline Approximate Nearest (and Furthest) Neighbors:* Given n red and n blue points in d-dimensional ℓ_1 or Euclidean space, we can find a $(1+\varepsilon)$-approximate nearest (or furthest) blue neighbor for each red point in randomized time near $dn + n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}$. This improves on an algorithm by Valiant (FOCS'12) with randomized time near $dn + n^{2-\Omega(\sqrt{\varepsilon})}$, which in turn improves previous methods based on locality-sensitive hashing.**
- ***SAT Algorithms and Lower Bounds for Circuits With Linear Threshold Functions:* We give a satisfiability algorithm for $\mathsf{AC}^0[m] \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuits with a subquadratic number of LTF gates on the bottom layer, and a subexponential number of gates on the other layers, that runs in deterministic $2^{n-n^{\varepsilon}}$ time. This strictly generalizes a SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{LTF}$ circuits of subexponential size by Williams (STOC'14) and also implies new circuit lower bounds for threshold circuits, improving a recent gate lower bound of Kane and Williams (STOC'16). We also give a randomized $2^{n-n^{\varepsilon}}$-time SAT algorithm for subexponential-size $\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits, where the top MAJ gate and middle LTF gates have $O(n^{6/5-\delta})$ fan-in.**

I. Introduction

The polynomial method is a powerful tool in circuit complexity. The idea of the method is to transform all circuits of some class into "nice" polynomials which represent the circuit in some way. If the polynomial is always sufficiently nice (e.g. has low degree), and one can prove that a certain Boolean function f cannot be represented so nicely, one concludes that the circuit class is unable to compute f.

Recently, these tools have found surprising uses in algorithm design. If a subproblem of an algorithmic problem can be modeled by a simple circuit, and that circuit can be transformed into a "nice" polynomial (or "nice" distribution of polynomials), then fast algebraic algorithms can be applied to evaluate or manipulate the polynomial quickly. This approach has led to advances on problems such as All-Pairs Shortest Paths [1], Orthogonal Vectors and Constraint Satisfaction [2], [3], [4], All-Nearest Neighbor problems [5], and Stable Matching [6].

In most applications, the key step is to randomly convert simple circuits into so-called *probabilistic* polynomials. If f is a Boolean function on n variables, and R is a ring, a *probabilistic polynomial over R for f with error $1/s$ and degree d* is a distribution $\mathscr{D}$ of degree-d polynomials over R such that for all $x \in \{0,1\}^n$, $\Pr_{p\sim\mathscr{D}}[p(x) = f(x)] \geq 1 - \frac{1}{s}$. Razborov [7] and Smolensky [8] introduced the notion of a probabilistic polynomial, and showed that any low-depth circuit consisting of AND, OR, and PARITY gates can be transformed into a low degree probabilistic polynomial by constructing constant degree probabilistic polynomials for those three gates. Many polynomial method algorithms use this transformation.

In this work, we are interested in polynomial representations of threshold functions. The threshold function TH_θ determines whether at least a θ fraction of its input bits are 1s. Threshold functions are among the simplest

Boolean functions that do not have constant degree probabilistic polynomials: Razborov and Smolensky showed that the MAJORITY function (a special case of a threshold function) requires degree $\Omega(\sqrt{n \log s})$. Nonetheless, as we will see throughout this paper, there are many important problems which can be reduced to evaluating circuits involving threshold gates on many inputs, and so further study of polynomial representations of threshold functions is warranted.

Threshold functions have been extensively studied in theoretical computer science for many years; there are numerous applications of linear and polynomial threshold functions to complexity and learning theory (a sample includes [9], [10], [11], [12], [13], [14], [15]).

A. Our Results

We consider three different notions of polynomials representing TH_θ. Each achieves different tradeoffs between polynomial degree, the randomness required, and how accurately the polynomial represents TH_θ. Each leads to improved algorithms in our applications.

Less Randomness. First, we revisit probabilistic polynomials. Alman and Williams [5] designed a probabilistic polynomial for TH_θ which already achieves a tight degree bound of $\Theta(\sqrt{n \log s})$. However, their construction uses $\Omega(n)$ random bits, which makes it difficult to apply in deterministic algorithms. We show how their low-degree probabilistic polynomials for threshold functions can use substantially fewer random bits:

Theorem I.1. *For any $0 \le \theta \le 1$, there is a probabilistic polynomial for the function TH_θ of degree $O(\sqrt{n \log s})$ on n bits with error $1/s$ that can be randomly sampled using only $O(\log n \log(ns))$ random bits.*

Polynomial Threshold Function Representations. Second, we consider deterministic Polynomial Threshold Functions (PTFs). A PTF for a Boolean function f is a polynomial (*not* a distribution on polynomials) $p : \{0,1\}^n \to \mathbb{R}$ such that $p(x)$ is smaller than a fixed value when $f(x) = 0$, and $p(x)$ is larger than the value when $f(x) = 1$. In our applications, we seek PTFs with "good threshold behavior", such that $|p(x)| \le 1$ when $f(x) = 0$, and $p(x)$ is very large otherwise. We can achieve almost the same degree for a PTF as for a probabilistic polynomial, and even better degree for an approximate threshold function:

Theorem I.2. *We can construct a polynomial $P_{s,t,\varepsilon} : \mathbb{R} \to \mathbb{R}$ of degree $O(\sqrt{1/\varepsilon} \log s)$, such that*

- *if $x \in \{0,1,\ldots,t\}$, then $|P_{s,t,\varepsilon}(x)| \le 1$;*
- *if $x \in (t,(1+\varepsilon)t)$, then $P_{s,t,\varepsilon}(x) > 1$;*
- *if $x \ge (1+\varepsilon)t$, then $P_{s,t,\varepsilon}(x) \ge s$.*

For the "exact" setting with $\varepsilon = 1/t$, we can alternatively bound the degree by $O(\sqrt{t} \log(st))$.

By summing multiple copies of the polynomial from Theorem I.2, we immediately obtain a PTF with the same degree for the OR of $O(s)$ threshold functions (needed in our applications). This theorem follows directly from known extremal properties of Chebyshev polynomials, as well as the lesser known *discrete* Chebyshev polynomials. Because Theorem I.2 gives a single polynomial instead of a distribution on polynomials, it is especially helpful for designing deterministic algorithms. Chebyshev polynomials are well-known to yield good approximate polynomials for computing certain Boolean functions over the reals [16], [17], [13], [18], [19] (please see the Preliminaries for more background).

Probabilistic PTFs. Third, we introduce a new (natural) notion of a *probabilistic PTF* for a Boolean function f. This is a distribution on PTFs, where for each input x, a PTF drawn from the distribution is highly likely to agree with f on x. Combining the techniques from probabilistic polynomials for TH_θ and the deterministic PTFs in a simple way, we construct a probabilistic PTF with good threshold behavior whose degree is *lower* than both the deterministic PTF and the degree bounds attainable by probabilistic polynomials (surprisingly breaking the "square-root barrier"):

Theorem I.3. *We can construct a probabilistic polynomial $\widetilde{P}_{n,s,t,\varepsilon} : \{0,1\}^n \to \mathbb{R}$ of degree $O((1/\varepsilon)^{1/3} \log s)$, such that*

- *if $\sum_{i=1}^n x_i \le t$, then $|\widetilde{P}_{n,s,t,\varepsilon}(x_1,\ldots,x_n)| \le 1$ with probability at least $1-1/s$;*
- *if $\sum_{i=1}^n x_i \in (t, t+\varepsilon n)$, then $\widetilde{P}_{n,s,t,\varepsilon}(x_1,\ldots,x_n) > 1$ with probability at least $1-1/s$;*
- *if $\sum_{i=1}^n x_i \ge t+\varepsilon n$, then $\widetilde{P}_{n,s,t,\varepsilon}(x_1,\ldots,x_n) \ge s$ with probability at least $1-1/s$.*

For the "exact" setting with $\varepsilon = 1/n$, we can alternatively bound the degree by $O(n^{1/3} \log^{2/3}(ns))$.

The PTFs of Theorem I.3 can be sampled using only $O(\log(n) \cdot \log(ns))$ random bits as well; their lower degree will allow us to design faster randomized algorithms for a variety of problems. For emphasis, we will sometimes refer to PTFs as *deterministic PTFs* to distinguish them from probabilistic PTFs.

These polynomials for TH_θ can be applied to many different problems:

Offline Hamming Nearest Neighbor Search. In the Hamming Nearest Neighbor problem, we wish to preprocess a set D of n points in $\{0,1\}^d$ such that, for a query $q \in \{0,1\}^d$, we can quickly find the $p \in D$ with smallest Hamming distance to q. This problem is central to many problems throughout Computer Science, especially in search and error correction [20]. However, it suffers from the *curse of dimensionality* phenomenon, where known algorithms achieve the nearly trivial runtimes of either $2^{\Omega(d)}$ or $\Omega(n/\text{poly}(\log n))$, with matching lower bounds in many data structure models (see e.g. [21]). Using our PTFs, we instead design a new algorithm for the natural offline version of this problem:

Theorem I.4. *Given n red and n blue points in $\{0,1\}^d$ for $d = c\log n \ll \log^3 n/\log^5\log n$, we can find an (exact) Hamming nearest/farthest blue neighbor for every red point in randomized time $n^{2-1/O(\sqrt{c}\log^{3/2} c)}$.*

Using the same ideas, we are also able to derandomize our algorithm, to achieve *deterministic* time $n^{2-1/O(c\log^2 c)}$ (see Remark 3 in Section V). When $d = c\log n$ for constant c, these algorithms both have "truly subquadratic" runtimes. These both improve on Alman and Williams' algorithm [5] which runs in randomized time $n^{2-1/O(c\log^2 c)}$, and only gives a nontrivial algorithm for $d \ll \log^2 n/\log^3\log n$. Applying reductions from [5], we can achieve similar runtimes for finding closest pairs in ℓ_1 for vectors with small integer entries, and pairs with maximum inner product or Jaccard coefficient.

It is worth noting that there may be a serious limit to solving this problem much faster. Theorem I.4 (and [5]) shows for all c there is a $\delta > 0$ such that Offline Hamming Nearest Neighbor search in dimension $d = c\log n$ takes $O(n^{2-\delta})$ time. Showing that there is a universal $\delta > 0$ that works for all c would disprove the Strong Exponential Time Hypothesis [5, Theorem 1.4].

Offline Approximate Nearest Neighbor Search. The problem of finding high-dimensional *approximate* nearest neighbors has received even more attention. Locality-sensitive hashing yields data structures that can find $(1+\varepsilon)$-factor approximate nearest neighbors to any query point in $\tilde{O}(dn^{1-\Omega(\varepsilon)})$ (randomized) time after preprocessing in $\tilde{O}(dn + n^{2-\Omega(\varepsilon)})$ time and space,[1] for not only Hamming space but also ℓ_1 and ℓ_2 space [22], [23]. Thus, a batch of n queries can be answered in $\tilde{O}(dn^{2-\Omega(\varepsilon)})$ randomized time. Exciting recent work on locality-sensitive hashing [24], [25] has improved the constant factor in the $\Omega(\varepsilon)$ bound, but not the growth rate in ε. In 2012, G. Valiant [19] reported a surprising algorithm running in $\tilde{O}(dn + n^{2-\Omega(\sqrt{\varepsilon})})$ randomized time for the offline version of the problem in ℓ_2. We obtain a still faster algorithm for the offline problem, with $\sqrt{\varepsilon}$ improved to about $\varepsilon^{1/3}$:

Theorem I.5. *Given n red and n blue points in $[U]^d$ and $\varepsilon \gg \frac{\log^6\log n}{\log^3 n}$, we can find a $(1+\varepsilon)$-approximate ℓ_1 or ℓ_2 nearest/farthest blue neighbor for each red point in $(dn + n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}) \cdot poly(\log(nU))$ randomized time.*

Valiant's algorithm, like Alman and Williams' [5], relied on fast matrix multiplication, and it also used Chebyshev polynomials but in a seemingly more complicated way. Our new probabilistic PTF construction is inspired by our attempt to unify Valiant's approach with Alman and Williams', which leads to not only a simplification but also an improvement of Valiant's algorithm. (We also almost succeed in derandomizing Valiant's $n^{2-\tilde{\Omega}(\sqrt{\varepsilon})}$ result in the Hamming case, except for an initial dimension reduction step; see Remark 3 in Section V.)

Numerous applications to high-dimensional computational geometry follow; for example, we can approximate the diameter or Euclidean minimum spanning tree in roughly the same running time.

MAX-SAT. Another application is MAX-SAT: finding an assignment that satisfies the maximum number of clauses in a given CNF formula with n variables. In the sparse case when the number of clauses is cn, a series of papers have given faster exact algorithms, for example, achieving $2^{n-n/O(c\log c)}$ time by Dantsin and Wolpert [26], $2^{n-n/O(c\log c)^{2/3}}$ time by Sakai et al. [27], and $2^{n-n/O(\sqrt{c})}$ time by Chen and Santhanam [28]. Using the polynomial method and our new probabilistic PTF construction, we obtain the following improved result:

Theorem I.6. *Given a CNF formula with n variables and $cn \ll n^4/\log^{10} n$ clauses, we can find an assignment that satisfies the maximum number of clauses in randomized $2^{n-n/O(c^{1/3}\log^{7/3} c)}$ time.*

For general dense instances, the problem becomes tougher. Williams [29] gave an $O(2^{0.792n})$-time algorithm for MAX-2-SAT, but an $O(2^{(1-\delta)n})$-time algorithm for MAX-3-SAT (for a universal $\delta > 0$) has remained open; currently the best reported time bound [30] is $2^{n-\Omega(n/\log n)^{1/3}}$, which can be slightly improved to $2^{n-\Omega(\sqrt{n/\log n})}$ with more care. We make new progress on not only MAX-3-SAT but also MAX-4-SAT:

Theorem I.7. *Given a weighted 4-CNF formula F with n variables with positive integer weights bounded by poly(n), we can find an assignment that maximizes the total weight of clauses satisfied in F, in randomized $2^{n-n/O(\log^2 n\log^2\log n)}$ time. In the sparse case when the clauses have total weight cn, the time bound improves to $2^{n-n/O(\log^2 c\log^2\log c)}$.*

The algorithms for MAX-SAT are described in the full version of this paper.

LTF-LTF Circuit SAT Algorithms and Lower

[1]Throughout the paper, the $\tilde{O}$ notation hides polylogarithmic factors, $[U]$ denotes $\{0,1,\ldots,U-1\}$, and poly(n) denotes a fixed polynomial in n.

Bounds. Using our small sample space for probabilistic MAJORITY polynomials (Theorem I.1), we construct a new circuit satisfiability algorithm for circuits with linear threshold functions (LTFs) which improves over several prior results. Let $\mathsf{AC}^0[d,m] \circ \mathsf{LTF} \circ \mathsf{LTF}[S_1,S_2,S_3]$ be the class of circuits with a layer of S_3 LTFs at the bottom layer (nearest the inputs), a layer of S_2 LTFs above the bottom layer, and a size-S_1 $\mathsf{AC}^0[m]$ circuit of depth d above the two LTF layers.[2]

Theorem I.8. *For every integer $d > 0$, $m > 1$, and $\delta > 0$, there is an $\varepsilon > 0$ and an algorithm for satisfiability of $\mathsf{AC}^0[d,m] \circ \mathsf{LTF} \circ \mathsf{LTF}[2^{n^{\varepsilon}}, 2^{n^{\varepsilon}}, n^{2-\delta}]$ circuits that runs in deterministic $2^{n-n^{\varepsilon}}$ time.*

Williams [31] gave a comparable SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{LTF}$ circuits of $2^{n^{\varepsilon}}$ size, where $\varepsilon > 0$ is sufficiently small.[3] Theorem I.8 strictly generalizes the previous algorithm, allowing another layer of $n^{2-\varepsilon}$ linear threshold functions below the existing LTF layer. Theorem I.8 also trivially implies deterministic SAT algorithms for $\mathsf{LTF} \circ \mathsf{LTF}$ circuits of up to $n^{2-o(1)}$ gates, improving over the recent SAT algorithms of Chen, Santhanam, and Srinivasan [32] which only work for $n^{1+\varepsilon}$-wire circuits for $\varepsilon \ll 1$, and the SAT algorithms of Impagliazzo, Paturi, and Schneider [33].

The SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{LTF} \circ \mathsf{LTF}$ is given in the full version of this paper; here we sketch the ideas in it. Similar to the SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{LTF}$ circuits [31], the bottom layer of LTFs can be replaced by a layer of DNFs, via a weight reduction trick. We replace LTFs in the middle layer with $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits (modifying a construction of Maciel and Thérien [34] to keep the fan-in of MAJ gates low), then replace these MAJ gates of $n^{2-\Theta(\delta)}$ fan-in with probabilistic $\mathbb{F}_2$-polynomials of degree $n^{1-\Theta(\delta)+\Theta(\varepsilon)}$ over a small sample space, provided by Theorem I.1. Taking a majority vote over all samples, and observing that an $\mathbb{F}_2$-polynomial is a $\mathsf{MOD}_2 \circ \mathsf{AND}$ circuit, we obtain a $\mathsf{MAJ} \circ \mathsf{ACC}^0$ circuit, but with $2^{n^{1-O(\delta)}}$ size in some of its layers. By carefully applying known depth reduction techniques, we can convert the circuit into a depth-two circuit of size $2^{n^{1-\Omega(\varepsilon)}}$ which can then be evaluated efficiently on many inputs. (This is not obvious: applying the Beigel-Tarui depth reduction to a $2^{O(n^{1-\varepsilon})}$-size circuit would make its new size *quasi-polynomial in* $2^{O(n^{1-\varepsilon})}$, yielding an intractable bound of $2^{n^{O(1)}}$.)

Applying the known connection between circuit satisfiability algorithms and circuit lower bounds for E^{NP} problems [35], [36], [37], the following is immediate:

[2]Recall that for an integer $m \geq 2$, $\mathsf{AC}^0[m]$ refers to constant-depth unbounded fan-in circuits over the basis $\{\mathsf{AND}, \mathsf{OR}, \mathsf{MOD}_m\}$, where MOD_m outputs 1 iff the sum of its input bits is divisible by m.

[3]Recall ACC^0 is the infinite union of $\mathsf{AC}^0[m]$ for all integers $m \geq 2$.

Corollary I.1. *For every $d > 0$, $m > 1$, and $\delta \in (0,1)$, there is an $\varepsilon > 0$ such that the class E^{NP} does not have non-uniform circuits in $\mathsf{AC}^0[d,m] \circ \mathsf{LTF} \circ \mathsf{LTF}[2^{n^{\varepsilon}}, 2^{n^{\varepsilon}}, n^{2-\delta}]$. In particular, for every $\varepsilon > 0$, E^{NP} does not have $\mathsf{ACC}^0 \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuits where the $\mathsf{ACC}^0 \circ \mathsf{LTF}$ subcircuit has $2^{n^{o(1)}}$ size and the bottom LTF layer has $n^{2-\varepsilon}$ gates.*

Most notably, Corollary I.1 proves lower bounds with $n^{2-\varepsilon}$ LTFs on the bottom layer and *subexponentially many* LTFs on the second layer. This improves upon recent $\mathsf{LTF} \circ \mathsf{LTF}$ gate lower bounds of Kane and Williams [38], at the cost of raising the complexity of the hard function from TC^0_3 to E^{NP}.

A Powerful Randomized SAT Algorithm. Finally, combining the probabilistic PTF for MAJORITY (Theorem I.3) with the probabilistic polynomial of [5], we give a randomized SAT algorithm for a rather powerful class of circuits. The class $\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ denotes the class of circuits with a majority gate at the top, along with two layers of linear threshold gates, and arbitrary $O(1)$-depth AC^0 circuitry between these three layers. This circuit class is arguably much more powerful than TC^0_3 ($\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$), based on known low-depth circuit constructions for arithmetic functions (e.g. [39], [34], [40]).

Theorem I.9. *For all $\varepsilon > 0$ and integers $d \geq 1$, there is a $\delta > 0$ and a randomized satisfiability algorithm for $\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits of depth d running in $2^{n-\Omega(n^{\delta})}$ time, on circuits with the following properties:*

- *the top MAJ gate, along with every LTF on the middle layer, has $O(n^{6/5-\varepsilon})$ fan-in, and*
- *there are $O(2^{n^{\delta}})$ many AND/OR gates (anywhere) and LTF gates at the bottom layer.*

Theorem I.9 applies the probabilistic PTF of degree about $n^{1/3}$ (Theorem I.3) to the top MAJ gate, probabilistic polynomials over $\mathbb{Z}$ of degree about $n^{1/2}$ (Theorem I.1) to the middle LTFs, and weight reduction to the bottom LTFs; the rest can be represented with poly(n^{δ}) degree. The full algorithm is given in the full version of this paper.

It would not be surprising (to at least one author) if the above circuit class contained strong pseudorandom function candidates; that is, it seems likely that the Natural Proofs barrier applies to this circuit class. Hence from the circuit lower bounds perspective, the problem of derandomizing the SAT algorithm of Theorem I.9 is extremely interesting.

II. Preliminaries

Notation. In what follows, for $(x_1,\ldots,x_n) \in \{0,1\}^n$ define $|x| := \sum_{i=1}^n x_i$. For a logical predicate P, we use the notation $[P]$ to denote the function which outputs 1 when P is true, and 0 when P is false.

For $\theta \in [0,1]$, define $\mathrm{TH}_\theta : \{0,1\}^n \to \{0,1\}$ to be the *threshold function* $\mathrm{TH}_\theta(x_1,\ldots,x_n) := [|x|/n \geq \theta]$. In particular, $\mathrm{TH}_{1/2} = \text{MAJORITY}$.

For classes of circuits $\mathscr{C}$ and $\mathscr{D}$, $\mathscr{C} \circ \mathscr{D}$ denotes the class of circuits consisting of a single circuit $C \in \mathscr{C}$ whose inputs are the outputs of some circuits from $\mathscr{D}$. That is, $\mathscr{C} \circ \mathscr{D}$ is simply the composition of circuits from $\mathscr{C}$ and $\mathscr{D}$.

Rectangular Matrix Multiplication. One of our key tools is fast rectangular matrix multiplication:

Lemma II.1 (Coppersmith [41]). *For all sufficiently large N, multiplication of an $N \times N^{.172}$ matrix with an $N^{.172} \times N$ matrix can be done in $O(N^2 \log^2 N)$ arithmetic operations over any field.*

A proof can be found in the appendix of [31].

Chebyshev Polynomials in TCS. Another key to our work is that we find new applications of Chebyshev polynomials to algorithm design. This is certainly not a new phenomenon in itself; here we briefly survey some prior related usages of Chebyshev polynomials. First, Nisan and Szegedy [16] used Chebyshev polynomials to compute the OR function on n Boolean variables with an "approximating" polynomial $p : \mathbb{R}^n \to \mathbb{R}$, such that for all $x \in \{0,1\}^n$ we have $|OR(x) - p(x)| \leq 1/3$, yet $\deg(p) = O(\sqrt{n})$. They also proved the degree bound is tight up to constants in the big-O; Paturi [17] generalized the upper and lower bound to all symmetric functions.

This work has led to several advances in learning theory. Building on the polynomials of Nisan and Szegedy, Klivans and Servedio [13] showed how to compute an OR of t ANDs of w variables with a PTF of degree $O(\sqrt{w}\log t)$, similar to our degree bound for computing an OR of t MAJORITYs of w variables of Theorem I.2 (however, note our bound in the "exact" setting is a bit better, due to our use of discrete Chebyshev polynomials). They also show how to compute an OR of s ANDs on n variables with a *deterministic* PTF of $O(n^{1/3}\log s)$ degree, similar to our cube-root-degree probabilistic PTF for the OR of MAJORITY of Theorem I.3 in the "exact" setting. However, it looks difficult to generalize Klivans-Servedio's $O(n^{1/3}\log s)$ degree bound to compute an OR of MAJORITY: part of their construction uses a reduction to decision lists which works for conjunctions but not for MAJORITY functions. Klivans, O'Donnell and Servedio [42] show how to compute an AND of k MAJORITY on n variables with a PTF of degree $O(\sqrt{w}\log k)$. By a simple transformation via De Morgan's law, there is a polynomial for OR of MAJORITY with the same degree. Their degree is only slightly worse than ours in terms of k (because we use discrete Chebyshev polynomials).

In streaming algorithms, Harvey, Nelson, and Onak [43] use Chebyshev polynomials to design efficient algorithms for computing various notions of entropy in a stream. As a consequence of a query upper bound in quantum computing, Ambainis et al. [44] show how to approximate any Boolean formula of size s with a polynomial of degree $\sqrt{s}^{1+o(1)}$, improving on earlier bounds of O'Donnell and Servedio [14] that use Chebyshev polynomials. Sachdeva and Vishnoi [45] give applications of Chebyshev polynomials to graph algorithms and matrix algebra. Linial and Nisan [46] use Chebyshev polynomials to approximate inclusion-exclusion formulas, and Sherstov [47] extends this to arbitrary symmetric functions.

III. Derandomizing Probabilistic Polynomials for Threshold Functions

In this section, we revisit the previous probabilistic polynomial for the majority function on n bits, and show it can be implemented using only $\mathrm{polylog}(n,s)$ random bits. Our construction is essentially identical to that of [5], except that we use far fewer random bits to sample entries from the input vector in the recursive step of the construction.

For the analysis, we need a Chernoff bound for bits with limited independence:

Lemma III.1 ([48] Theorem 5 (I)(b)). *If X is the sum of k-wise independent random variables, each of which is confined to the interval $[0,1]$, with $\mu = \mathbb{E}[X]$, $\delta \leq 1$, and $k = \lfloor \delta^2 \mu e^{-1/3} \rfloor$, then*

$$\Pr[|X - \mu| \geq \delta\mu] \leq e^{-\delta^2\mu/3}.$$

In particular, the following inequality appears in the analysis of [5]:

Corollary III.1. *If $x \in \{0,1\}^n$ with $|x|/n = w$, and $\tilde{x} \in \{0,1\}^{n/10}$ is a vector each of whose entries is k-wise independently chosen entry of x, where $k = \lfloor 20e^{-1/3}\log(1/\varepsilon) \rfloor$, with $|\tilde{x}|/(n/10) = v$, then for every $\varepsilon < 1/4$,*

$$\Pr\left[v \leq w - \frac{a}{\sqrt{n}}\right] \leq \frac{\varepsilon}{4},$$

where $a = \sqrt{10} \cdot \sqrt{\ln(1/\varepsilon)}$.

Proof: Apply Lemma III.1 with $X = |\tilde{x}|$, $\mu = \mathbb{E}[|\tilde{x}|] = wn$, and $\delta = \sqrt{40\log(1/\varepsilon)/n}$. ■

Reminder of Theorem I.1. *For any $0 \leq \theta \leq 1$, there is a probabilistic polynomial for the threshold function* TH_θ

of degree $O(\sqrt{n}\log s)$ on n bits with error $1/s$ that can be randomly sampled using $O(\log(n)\log(ns))$ random bits.

Proof: Our polynomial is defined recursively, just as in [5]. Set $\varepsilon = 1/s$. Using their notation, the polynomial $M_{n,\theta,\varepsilon}$ for computing TH_θ on n bits with error ε is defined by:

$$\begin{aligned} M_{n,\theta,\varepsilon}(x) := &A_{n,\theta,2a}(x)\cdot S_{n/10,\theta,a/\sqrt{n},\varepsilon/4}(\tilde{x}) \\ &+ M_{n/10,\theta,\varepsilon/4}(\tilde{x})\cdot(1 - S_{n/10,\theta,a/\sqrt{n},\varepsilon/4}(\tilde{x})). \end{aligned}$$

In [5], $\tilde{x}$ was a sample of $n/10$ bits of x, chosen independently at random. Here, we pick $\tilde{x}$ to be a sample of $n/10$ bits chosen k-wise independently, for $k = \lfloor 20e^{-1/3}\log(1/\varepsilon)\rfloor$. The other polynomials in this recursive definition are as in [5]:

- $M_{m,\theta,\varepsilon}$ for $m < n$ is the (recursively defined) probabilistic polynomial for TH_θ on m bits and ε error
- $S_{m,\theta,\delta,\varepsilon}(x) := (1 - M_{m,\theta+\delta,\varepsilon}(x))\cdot M_{m,\theta-\delta,\varepsilon}(x)$ for $m < n$
- $A_{n,\theta,g} : \{0,1\}^n \to \mathbb{Z}$ is an exact polynomial of degree at most $2g\sqrt{n}+1$ which gives the correct answer to TH_θ for any vector x with $|x| \in [\theta n - g\sqrt{n}, \theta n + g\sqrt{n}]$, and may give arbitrary answers on other vectors.

Examining the proof of correctness in Alman and Williams [5], we see that the only requirement of the randomness is that it satisfies their Lemma 3.4, a concentration inequality for sampling $\tilde{x}$ from x. Our Corollary III.1 is identical to their Lemma 3.4, except that it replaces their method of sampling $\tilde{x}$ with k-wise sampling; the remainder of the proof of correctness is exactly as before.

Our polynomial construction is recursive: we divide n by 10 and divide ε by 4, each time we move from one recursive layer to the next. At the jth recursive level of our construction, for $1 \le j < \log_{10}(n)$, we need to $O(\log(4^j/\varepsilon))$-wise independently sample $n/10^j$ entries from a vector of length $n/10^{j-1}$. Summing across all of the layers, we need a total of $O(n)$ samples from a k-wise independent space, where k is never more than $O(n/\varepsilon)$. This can be done all together using $O(n)$ samples from $\{1,2,\ldots,n\}$ which are $O(n/\varepsilon)$-wise independent. Using standard constructions, this requires $O(\log(n)\log(n/\varepsilon))$ random bits. ∎

IV. PTFs for ORs of Threshold Functions

In this section, we show how to construct low-degree PTFs representing threshold functions that have good threshold behavior, and consequently obtain low-degree PTFs for an OR of many threshold functions.

A. Deterministic Construction

We begin by reviewing some basic facts about Chebyshev polynomials (see the full version of this paper for proof sketches). The *degree-q Chebyshev polynomial of the first kind* is

$$T_q(x) := \sum_{i=0}^{\lfloor q/2\rfloor}\binom{q}{2i}(x^2-1)^i x^{q-2i}.$$

Fact IV.1. *For any $\varepsilon \in (0,1)$,*

- *if $x \in [-1,1]$, then $|T_q(x)| \le 1$;*
- *if $x \in (1, 1+\varepsilon)$, then $T_q(x) > 1$;*
- *if $x \ge 1+\varepsilon$, then $T_q(x) \ge \frac{1}{2}e^{q\sqrt{\varepsilon}}$.*

In certain scenarios, we obtain slightly better results using a (lesser known) family of *discrete Chebyshev polynomials* defined as follows [49, page 59]:

$$D_{q,t}(x) := \sum_{i=0}^{q}(-1)^i\binom{q}{i}\binom{t-x}{q-i}\binom{x}{i}.$$

(See also [50, pages 33–34] or Chebyshev's original paper [51] with an essentially equivalent definition up to rescaling.)

Fact IV.2. *Let $c_{q,t} = (t+1)^{q+1}/q!$. For all $t > q \ge \sqrt{8(t+1)\ln(t+1)}$,*

- *if $x \in \{0,1,\ldots,t\}$, then $|D_{q,t}(x)| \le c_{q,t}$;*
- *if $x \le -1$, then $D_{q,t}(x) \ge e^{q^2/(8(t+1))}c_{q,t}$.*

Reminder of Theorem I.2. *We can construct a polynomial $P_{s,t,\varepsilon} : \mathbb{R} \to \mathbb{R}$ of degree $O(\sqrt{1/\varepsilon}\log s)$, such that*

- *if $x \in \{0,1,\ldots,t\}$, then $|P_{s,t,\varepsilon}(x)| \le 1$;*
- *if $x \in (t, (1+\varepsilon)t)$, then $P_{s,t,\varepsilon}(x) > 1$;*
- *if $x \ge (1+\varepsilon)t$, then $P_{s,t,\varepsilon}(x) \ge s$.*

For the "exact" setting with $\varepsilon = 1/t$, we can alternatively bound the degree by $O(\sqrt{t\log(st)})$.

Proof: Set $P_{s,t,\varepsilon}(x) := T_q(x/t)$ for a parameter q to be determined. The first two properties are obvious from Fact IV.1. On the other hand, if $x \ge (1+\varepsilon)t$, then Fact IV.1 shows that $P_{s,t,\varepsilon}(x) \ge \frac{1}{2}e^{q\sqrt{\varepsilon}} \ge s$, provided we set $q = \left\lceil \sqrt{1/\varepsilon}\ln(2s)\right\rceil$. This achieves $O(\sqrt{1/\varepsilon}\log s)$ degree.

When $\varepsilon = 1/t$ the above yields $O(\sqrt{t}\log s)$ degree; we can reduce the $\log s$ factor by instead defining $P_{s,t,\varepsilon}(x) := D_{q,t}(t-x)/c_{q,t}$. Now, if $x \ge t+1$, then $P_{s,t,\varepsilon}(x) \ge e^{q^2/(8(t+1))} \ge s$ by setting $q = \left\lceil\sqrt{8(t+1)\ln(\max\{s,t+1\})}\right\rceil$. ∎

Using Theorem I.2, we can construct a low-degree PTF for computing an OR of s thresholds of n bits:

Corollary IV.1. *Given n,s,t,ε, we can construct a polynomial $P : \{0,1\}^{ns} \to \mathbb{R}$ of degree at most $\Delta := O(\sqrt{1/\varepsilon}\log s)$ and at most $s \cdot \binom{n}{\Delta}$ monomials, such that*

- *if the formula $\bigvee_{i=1}^{s}\left[\sum_{j=1}^{n} x_{ij} > t\right]$ is false, then $|P(x_{11},\dots,x_{1n},\dots,x_{s1},\dots,x_{sn})| \le s$;*
- *if the formula $\bigvee_{i=1}^{s}\left[\sum_{j=1}^{n} x_{ij} \ge t+\varepsilon n\right]$ is true, then $P(x_{11},\dots,x_{1n},\dots,x_{s1},\dots,x_{sn}) > 2s$.*

For the exact setting with $\varepsilon = 1/n$, we can alternatively bound Δ by $O(\sqrt{n\log(ns)})$.

Proof: Define $P(x_{11},\dots,x_{1n},\dots,x_{s1},\dots,x_{sn}) := \sum_{i=1}^{s} P_{n,3s,t,\varepsilon}\left(\sum_{j=1}^{n} x_{ij}\right)$, where $P_{n,3s,t,\varepsilon}$ is from Theorem I.2. The stated properties clearly hold. (In the second case, the output is at least $3s-(s-1) > 2s$.) ■

B. Probabilistic Construction

Allowing ourselves a distribution of PTFs to randomly draw from, we can achieve noticeably lower degree than the previous section. We start with a fact which follows easily from the (tight) probabilistic polynomial for MAJORITY:

Fact IV.3. (Alman–Williams [5], or Theorem I.1) *We can construct a probabilistic polynomial $Q_{n,s,t} : \{0,1\}^n \to \mathbb{R}$ of degree $O(\sqrt{n\log s})$, such that*

- *if $\sum_{i=1}^{n} x_i \le t$, then $Q_{n,s,t}(x_1,\dots,x_n) = 0$ with probability at least $1-1/s$;*
- *if $\sum_{i=1}^{n} x_i > t$, then $Q_{n,s,t}(x_1,\dots,x_n) = 1$ with probability at least $1-1/s$.*

Reminder of Theorem I.3. *We can construct a probabilistic polynomial $\widetilde{P}_{n,s,t,\varepsilon} : \{0,1\}^n \to \mathbb{R}$ of degree $O((1/\varepsilon)^{1/3}\log s)$, such that*

- *if $\sum_{i=1}^{n} x_i \le t$, then $|\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_n)| \le 1$ with probability at least $1-1/s$;*
- *if $\sum_{i=1}^{n} x_i \in (t, t+\varepsilon n)$, then $\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_n) > 1$ with probability at least $1-1/s$;*
- *if $\sum_{i=1}^{n} x_i \ge t+\varepsilon n$, then $\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_n) \ge s$ with probability at least $1-1/s$.*

For the "exact" setting with $\varepsilon = 1/n$, we can alternatively bound the degree by $O(n^{1/3}\log^{2/3}(ns))$.

Proof: Let r and q be parameters to be set later. Draw a random sample $R \subseteq \{1,\dots,n\}$ of size r. Let

$$t_R := \frac{tr}{n} - c_0\sqrt{r\log s} \quad \text{and} \quad t^- := t - 2c_0\left(\frac{n}{\sqrt{r}}\right)\sqrt{\log s}$$

for a sufficiently large constant c_0. Define

$$\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_d) := Q_{r,2s,t_R}(\{x_i\}_{i\in R}) \cdot P_{s,t',\varepsilon'}\left(\sum_{i=1}^{n} x_i - t^-\right),$$

where $P_{s,t',\varepsilon'}$ is the polynomial from Theorem I.2, with $t' := t - t^- = \Theta((n/\sqrt{r})\sqrt{\log s})$ and $\varepsilon' := \varepsilon n/t' = \Theta(\varepsilon\sqrt{r}/\sqrt{\log s})$.

To verify the stated properties, consider three cases:

- CASE 1: $\sum_{i=1}^{n} x_i < t^-$. By a standard Chernoff bound, with probability at least $1-1/(2s)$, we have $\sum_{i\in R} x_i < t^- r/n + c_0\sqrt{r\log s} \le t_R$ (assuming that $r \ge \log s$). Thus, with probability at least $1-1/s$, we have $Q_{n,2s,t_R}(\{x_i\}_{i\in R}) = 0$ and so $\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_n) = 0$.
- CASE 2: $\sum_{i=1}^{n} x_i \in [t^-, t]$. With probability at least $1-1/s$, we have $Q_{r,2s,t_R}(\{x_i\}_{i\in R}) \in \{0,1\}$ and so $|\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_n)| \le 1$.
- CASE 3: $\sum_{i=1}^{n} x_i > t$. By a standard Chernoff bound, with probability at least $1-1/(2s)$, we have $\sum_{i\in R} x_i \ge tr/n + c_0\sqrt{r\log s} = t_R$. Thus, with probability at least $1-1/s$, we have $Q_{r,2s,t_R}(\{x_i\}_{i\in R}) = 1$ and so $\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_n) > 1$ for $\sum_{i=1}^{n} x_i \in (t, t+\varepsilon n)$, or $\widetilde{P}_{n,s,t,\varepsilon}(x_1,\dots,x_n) \ge s$ for $\sum_{i=1}^{n} x_i \ge t+\varepsilon n$.

The degree of $\widetilde{P}_{n,s,t,\varepsilon}$ is

$$O\left(\sqrt{r\log s} + \sqrt{(1/(\varepsilon\sqrt{r}))\sqrt{\log s}}\log s\right)$$

and we can set $r = \lceil (1/\varepsilon)^{2/3}\log s\rceil$. For the exact setting, the degree is

$$O\left(\sqrt{r\log s} + \sqrt{(n/\sqrt{r})\sqrt{\log s}\cdot\log(ns)}\right)$$

and we can set $r = \left\lceil n^{2/3}\log^{1/3}(ns)\right\rceil$. ■

Remark 1. Using the same techniques as in Theorem I.1, we can sample a probabilistic polynomial from Theorem I.3 with only $O(\log(n)\log(ns))$ random bits.

Corollary IV.2. *Given d,s,t,ε, we can construct a probabilistic polynomial $\widetilde{P} : \{0,1\}^{ns} \to \mathbb{R}$ of degree at most $\Delta := O((1/\varepsilon)^{1/3}\log s)$ with at most $s \cdot \binom{n}{D}$ monomials, such that*

- *if $\bigvee_{i=1}^{s}\left[\sum_{j=1}^{n} x_{ij} \ge t\right]$ is false, then $|\widetilde{P}(x_{11},\dots,x_{1n},\dots,x_{s1},\dots,x_{sn})| \le s$ with probability at least $2/3$;*
- *if $\bigvee_{i=1}^{s}\left[\sum_{j=1}^{d} x_{ij} \ge t+\varepsilon n\right]$ is true, then $\widetilde{P}(x_{11},\dots,x_{1n},\dots,x_{s1},\dots,x_{sn}) > 2s$ with probability at least $2/3$.*

For the exact setting with $\varepsilon = 1/n$, we can alternatively bound Δ by $O(n^{1/3}\log^{2/3}(ns))$.

Proof: Define $\widetilde{P}(x_{11},\dots,x_{1n},\dots,x_{s1},\dots,x_{sn}) := \sum_{i=1}^{s} \widetilde{P}_{n,3s,t_i,\varepsilon}(x_{i1},\dots,x_{in})$. ■

Remark 2. The coefficients of the polynomials from Fact IV.3 are poly(n)-bit integers, and it can be checked that the coefficients of all our deterministic and probabilistic polynomials are rational numbers with poly(n)-bit numerators and a common poly(n)-bit denominator, and that the same bound for the number of monomials holds for the construction time, up to poly(n) factors. That is, computations with these polynomials have low computational overhead relative to n.

V. Exact and Approximate Offline Nearest Neighbor Search

We now apply our new probabilistic PTF construction to obtain a faster algorithm for offline exact nearest/farthest neighbor search in Hamming space:

Reminder of Theorem I.4. *Given n red and n blue points in $\{0,1\}^d$ for $d = c\log n \ll \log^3 n/\log^5 \log n$, we can find an (exact) Hamming nearest/farthest blue neighbor for every red point in randomized time $n^{2-1/O(\sqrt{c}\log^{3/2} c)}$.*

Proof: We proceed as in Abboud, Williams, and Yu's algorithm for Boolean orthogonal vectors [3] or Alman and Williams' algorithm for Hamming closest pair [5]. For a fixed t, we first solve the decision problem of testing whether the nearest neighbor distance is less than t for each red point. (Farthest neighbors are similar.) Let $s = n^\alpha$ for some parameter α to be set later. Arbitrarily divide the blue point set into n/s groups of s points. For every group G of blue points and every red point q, we want to test whether

$$\begin{aligned} F(G,q) &:= \left[\min_{p\in G}\|p-q\|_1 < t\right] \\ &= \bigvee_{p\in G}\left[\sum_{i=1}^{d}(p_iq_i + (1-p_i)(1-q_i)) > d-t\right] \end{aligned}$$

(where p_i denotes the i-th coordinate of a point p). By Corollary IV.2, we can express $F(G,q)$ as a probabilistic polynomial that has the following number of monomials:

$$\begin{aligned} & s\cdot\binom{O(d)}{O(d^{1/3}\log^{2/3}(ds))} \\ &\le n^\alpha\cdot O\left(\frac{c\log n}{c^{1/3}\alpha^{2/3}\log n}\right)^{O(c^{1/3}\alpha^{2/3}\log n)} \\ &\le n^\alpha\cdot n^{O(c^{1/3}\alpha^{2/3}\log\frac{c}{\alpha})} \ \ll\ (n/s)^{0.1} \end{aligned}$$

for large enough n, by setting α to be a sufficiently small constant times $1/(c^{1/3}\log^{3/2} c)$. The same bound holds for the construction time of the polynomial.

We can rewrite the polynomial for $F(G,q)$ as the dot product of two vectors $\phi(G)$ and $\psi(q)$ in $(n/s)^{0.1}$ dimensions over $\mathbb{R}$. The problem of evaluating $F(G,q)$ over all n/s groups G of blue points and all red points q then reduces to multiplying an $n/s\times(n/s)^{0.1}$ with an $(n/s)^{0.1}\times n$ matrix over $\mathbb{R}$. This in turn reduces to s instances of multiplication of $n/s\times(n/s)^{0.1}$ with $(n/s)^{0.1}\times n/s$ matrices, each of which can be done in $\tilde{O}(n/s)^2$ arithmetic operations on poly(d)-bit numbers over an appropriately large field (Lemma II.1). The total time is $\tilde{O}(\text{poly}(d)n^2/s) = O(n^{2-1/O(c^{1/3}\log^{3/2} c)})$.

The error probability for each pair (G,q) is at most $1/3$, which can be lowered to $O(1/n^3)$, for example, by repeating $O(\log n)$ times (and taking the majority of the answers). The overall error probability is then $O(1/n)$. This solves the decision problem for a fixed t, but we can compute all nearest neighbor distances by calling the decision algorithm d times for all values of t. For each red point, we can find an actual nearest neighbor in additional $O(s)$ time, since we know which group achieves the nearest neighbor distance. ■

The same approach can be applied to solve *approximate* nearest neighbor search in Hamming space:

Theorem V.1. *Given n red and n blue points in $\{0,1\}^d$ and $\varepsilon \gg \log^6(d\log n)/\log^3 n$, we can find an approximate Hamming nearest/farthest blue neighbor with additive error at most εd for each red point in randomized time $n^{2-\Omega(\varepsilon^{1/3}/\log(\frac{d}{\varepsilon\log n}))}$.*

Proof: We mimic the proof of Theorem I.4 up to the definition of the polynomial $F(G,q)$. However, instead of applying the exact polynomial of Corollary IV.2, we insert the *approximate* polynomial construction from the same corollary. While the exact polynomial had degree $O(d^{1/3}\log^{2/3}(ds))$, the approximate one has degree $O((1/\varepsilon)^{1/3}\log s)$. Setting

$$s := n^\alpha := n^{\Omega(\varepsilon^{1/3}/\log(\frac{d}{\varepsilon\log n}))},$$

the number of monomials in the new polynomial is now

$$\begin{aligned} & s\cdot\binom{O(d)}{O((1/\varepsilon)^{1/3}\log s)} \\ &\le n^\alpha\cdot O\left(\frac{d}{(\alpha/\varepsilon^{1/3})\log n}\right)^{O((\alpha/\varepsilon^{1/3})\log n)} \\ &\le n^\alpha\cdot n^{O((\alpha/\varepsilon^{1/3})\log\frac{d}{\alpha\log n})} \ \ll\ (n/s)^{0.1}, \end{aligned}$$

for large enough n. The remainder of the algorithm is the same as the proof of Theorem I.4, and the running time is $\tilde{O}(n^2/s^2) \le n^{2-\Omega(\varepsilon^{1/3}/\log(\frac{d}{\varepsilon\log n}))}$. ■

Remark 3. For deterministic algorithms, using Corollary IV.1 instead, the time bounds for Theorems I.4 and I.5 become $n^{2-1/O(c\log^2 c)}$ and $n^{2-\Omega(\sqrt{\varepsilon}/\log(\frac{d}{\varepsilon\log n}))}$ respectively.

The algorithm of Theorem V.1 still has three drawbacks: (i) the exponent in the time bound depends on the dimension d, (ii) the result requires additive instead of multiplicative error, and (iii) the result is for Hamming space instead of more generally ℓ_1 or ℓ_2. We can resolve all three issues at once, by using known dimension reduction techniques:

Reminder of Theorem I.5. *Given n red and n blue points in $[U]^d$ and $\varepsilon \gg \frac{\log^6 \log n}{\log^3 n}$, we can find a $(1+\varepsilon)$-approximate ℓ_1 or ℓ_2 nearest/farthest blue neighbor for each red point in $(dn + n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}) \cdot$ poly$(\log(nU))$ randomized time.*

Proof: **(The ℓ_1 case.)** We first solve the decision problem for a fixed threshold value t. We use a variant of ℓ_1 locality-sensitive hashing (see [52]) to map points from ℓ_1 into low-dimensional Hamming space (providing an alternative to Kushilevitz, Ostrovsky, and Rabani's dimension reduction technique for Hamming space [53]). For each red/blue point p and each $i \in \{1,\ldots,k\}$, define $h_i(p) = (h_{i1}(p),\ldots,h_{id}(p))$ with $h_{ij}(p) = \lfloor (p_{a_{ij}} + b_{ij})/(2t) \rfloor$ where $a_{ij} \in \{1,\ldots,d\}$ and $b_{ij} \in [0,2t)$ are independent uniformly distributed random variables. For each of the $O(n)$ hashed values of h_i, pick a random bit; let $f_i(p)$ be the random bit associated with $h_i(p)$. Finally, define $f(p) = (f_1(p),\ldots,f_k(p)) \in \{0,1\}^k$. For any fixed p,q,

$$\begin{aligned}\Pr[h_{ij}(p) \neq h_{ij}(q)] &= \frac{1}{d}\sum_{a=1}^{d} \min\left\{\frac{|p_a - q_a|}{2t}, 1\right\},\\ \Pr[f_i(p) \neq f_i(q)] &= \frac{1}{2}\Pr[h_i(p) \neq h_i(q)]\\ &= \frac{1}{2}\Pr\left[\bigvee_{j=1}^{k} [h_{ij}(p) \neq h_{ij}(q)]\right].\end{aligned}$$

- If $\|p-q\|_1 \le t$, then $\Pr[h_{ij}(p) \neq h_{ij}(q)] \le \frac{\|p-q\|_1}{2dt} \le \frac{1}{2d}$ and $\Pr[f_i(p) \neq f_i(q)] \le \alpha_0 := \frac{1}{2}(1-(1-\frac{1}{2d})^d)$;
- if $\|p-q\|_1 \ge (1+\varepsilon)t$, then $\Pr[h_{ij}(p) \neq h_{ij}(q)] \ge \min\{\frac{\|p-q\|_1}{2dt}, \frac{1}{d}\} \ge \frac{1+\varepsilon}{2d}$ and $\Pr[f_i(p) \neq f_i(q)] \ge \alpha_1 := \frac{1}{2}(1-(1-\frac{1+\varepsilon}{2d})^d)$.

Note that $\alpha_1 - \alpha_0 = \Omega(\varepsilon)$. By a Chernoff bound, it follows (assuming $k \ge \log n$) that

- if $\|p-q\|_1 \le t$, then $\|f(p)-f(q)\|_1 \le A_0 := \alpha_0 k + O(\sqrt{k \log n})$ with probability $1-O(1/n^3)$;
- if $\|p-q\|_1 \ge (1+\varepsilon)t$, then $\|f(p)-f(q)\|_1 \ge A_1 := \alpha_1 k - O(\sqrt{k \log n})$ with probability $1-O(1/n^3)$.

Note that $A_1 - A_0 = \Omega(\varepsilon k)$ by setting k to be a sufficiently large constant times $(1/\varepsilon)^2 \log n$. We have thus reduced the problem to an approximate problem with additive error $O(\varepsilon k)$ for Hamming space in $k = O((1/\varepsilon^2)\log n)$ dimensions, which by Theorem V.1 requires $n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}$ time. The initial cost of applying the mapping f is $O(dkn)$.

This solves the decision problem; we can solve the original problem by calling the decision algorithm $O(\log_{1+\varepsilon} U)$ times for all t's that are powers of $1+\varepsilon$. ■

Proof: **(The ℓ_2 case.)** We use a version of the Johnson–Lindenstrauss lemma to map from ℓ_2 to ℓ_1 (see for example [54]). For each red/blue point p, define $f(p) = (f_1(p),\ldots,f_k(p)) \in \mathbb{R}^k$ with $f_i(p) = \sum_{j=1}^{k} a_{ij} p_j$, where the a_{ij}'s are independent normally distributed random variables with mean 0 and variance 1. For each fixed $p,q \in \mathbb{R}^d$, it is known that after rescaling by a constant, $\|f(p)-f(q)\|_1$ approximates $\|p-q\|_2$ to within $1 \pm O(\varepsilon)$ factor with probability $1-O(1/n^3)$, by setting $k = O((1/\varepsilon)^2 \log n)$. It suffices to keep $O(\log U)$-bit precision of the mapped points. The initial cost of applying the mapping f is $O(dkn)$ (which can be slightly improved by utilizing a sparse Johnson–Lindenstrauss transform [55]). ■

Numerous applications to high-dimensional computational geometry now follow. We briefly mention just one such application, building on the work of [56], [22]:

Corollary V.1. *Given n points in $[U]^d$ and $\varepsilon \gg \log^6 \log n / \log^3 n$, we can find a $(1+\varepsilon)$-approximate ℓ_1 or ℓ_2 minimum spanning tree in $(dn + n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}) \cdot$ poly$(\log(nU))$ randomized time.*

Acknowledgments

The authors thank the FOCS referees for their helpful comments. J.A. was supported by NSF CCF-1212372 and NSF DGE-114747. T.M.C. was supported by an NSERC grant. R.W. was supported in part by NSF CCF-1212372 and CCF-1552651 (CAREER). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] R. Williams, "Faster all-pairs shortest paths via circuit complexity," in *STOC*, 2014, pp. 664–673.
[2] R. Williams and H. Yu, "Finding orthogonal vectors in discrete structures," in *SODA*, 2014, pp. 1867–1877.
[3] A. Abboud, R. Williams, and H. Yu, "More applications of the polynomial method to algorithm design," in *SODA*, 2015, pp. 218–230.
[4] R. Williams, "The polynomial method in circuit complexity applied to algorithm design (invited talk)," in *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, 2014, pp. 47–60.
[5] J. Alman and R. Williams, "Probabilistic polynomials and Hamming nearest neighbors," in *FOCS*, 2015, pp. 136–150.
[6] D. Moeller, R. Paturi, and S. Schneider, "Subquadratic algorithms for succinct stable matching," in *Computer Science Symposium in Russia*, 2016, pp. 294–308.
[7] A. A. Razborov, "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition," *Mathematical Notes of the Academy of Sciences of the USSR*, vol. 41, no. 4, pp. 333–338, 1987.
[8] R. Smolensky, "Algebraic methods in the theory of lower bounds for boolean circuit complexity," in *STOC*, 1987, pp. 77–82.
[9] R. Beigel, N. Reingold, and D. A. Spielman, "The perceptron strikes back," in *IEEE Structure in Complexity Theory Conference*, 1991, pp. 286–291.

[10] J. Bruck and R. Smolensky, "Polynomial threshold functions, AC^0 functions, and spectral norms," *SIAM J. Comput.*, vol. 21, no. 1, pp. 33–42, 1992.

[11] J. Aspnes, R. Beigel, M. Furst, and S. Rudich, "The expressive power of voting polynomials," *Combinatorica*, vol. 14, no. 2, pp. 135–148, 1994.

[12] R. Beigel, "The polynomial method in circuit complexity," in *IEEE Structure in Complexity Theory Conference*, 1995, pp. 82–95.

[13] A. R. Klivans and R. Servedio, "Learning DNF in time $2^{\tilde{O}(n^{1/3})}$," in *STOC*, 2001, pp. 258–265.

[14] R. O'Donnell and R. A. Servedio, "New degree bounds for polynomial threshold functions," *Combinatorica*, vol. 30, no. 3, pp. 327–358, 2010.

[15] A. A. Sherstov, "Breaking the Minsky–Papert barrier for constant-depth circuits," in *STOC*, 2014, pp. 223–232.

[16] N. Nisan and M. Szegedy, "On the degree of boolean functions as real polynomials," *Computational Complexity*, vol. 4, no. 4, pp. 301–313, 1994.

[17] R. Paturi, "On the degree of polynomials that approximate symmetric boolean functions (preliminary version)," in *STOC*, 1992, pp. 468–474.

[18] A. A. Sherstov, "Making polynomials robust to noise," *Theory of Computing*, vol. 9, pp. 593–615, 2013.

[19] G. Valiant, "Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem," *J. ACM*, vol. 62, no. 2, p. 13, 2015. Preliminary version in FOCS'12.

[20] P. Indyk, "Nearest neighbors in high-dimensional spaces," in *Handbook of Discrete and Computational Geometry*, 2nd ed. Chapman and Hall, 2004, pp. 877–892.

[21] O. Barkol and Y. Rabani, "Tighter lower bounds for nearest neighbor search and related problems in the cell probe model," *JCSS*, vol. 64, no. 4, pp. 873–896, 2002.

[22] S. Har-Peled, P. Indyk, and R. Motwani, "Approximate nearest neighbor: Towards removing the curse of dimensionality," *Theory of Computing*, vol. 8, no. 1, pp. 321–350, 2012.

[23] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *FOCS*, 2006, pp. 459–468.

[24] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn, "Beyond locality-sensitive hashing," in *SODA*, 2014, pp. 1018–1028.

[25] A. Andoni and I. Razenshteyn, "Optimal data-dependent hashing for approximate near neighbors," in *STOC*, 2015, pp. 793–801.

[26] E. Dantsin and A. Wolpert, "MAX-SAT for formulas with constant clause density can be solved faster than in $O(2^n)$ time," in *SAT*, 2006, pp. 266–276.

[27] T. Sakai, K. Seto, S. Tamaki, and J. Teruyama, "Improved exact algorithms for mildly sparse instances of Max SAT," in *IPEC*, 2015, pp. 90–101.

[28] R. Chen and R. Santhanam, "Improved algorithms for sparse MAX-SAT and MAX-k-CSP," in *SAT*, 2015, pp. 33–45.

[29] R. Williams, "A new algorithm for optimal 2-constraint satisfaction and its implications," *Theor. Comput. Sci.*, vol. 348, no. 2-3, pp. 357–365, 2005. See also ICALP'04.

[30] T. Sakai, K. Seto, S. Tamaki, and J. Teruyama, "A satisfiability algorithm for depth-2 circuits with a symmetric gate at the top and AND gates at the bottom," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 22, p. 136, 2015.

[31] R. Williams, "New algorithms and lower bounds for circuits with linear threshold gates," in *STOC*, 2014, pp. 194–202.

[32] R. Chen, R. Santhanam, and S. Srinivasan, "Average-case lower bounds and satisfiability algorithms for small threshold circuits," in *CCC*, 2016, pp. 1:1–1:35.

[33] R. Impagliazzo, R. Paturi, and S. Schneider, "A satisfiability algorithm for sparse depth two threshold circuits," in *FOCS*, 2013, pp. 479–488.

[34] A. Maciel and D. Thérien, "Threshold circuits of small majority-depth," *Information and Computation*, vol. 146, no. 1, pp. 55–83, 1998.

[35] R. Williams, "Improving exhaustive search implies superpolynomial lower bounds," *SIAM J. Comput.*, vol. 42, no. 3, pp. 1218–1244, 2013. See also STOC'10.

[36] ——, "Nonuniform ACC circuit lower bounds," *J. ACM*, vol. 61, no. 1, p. 2, 2014.

[37] H. Jahanjou, E. Miles, and E. Viola, "Local reductions," in *ICALP Part I*, 2015, pp. 749–760.

[38] D. M. Kane and R. Williams, "Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 22, p. 188, 2015. To appear in STOC'16.

[39] A. K. Chandra, L. Stockmeyer, and U. Vishkin, "Constant depth reducibility," *SIAM J. Computing*, vol. 13, no. 2, pp. 423–439, 1984.

[40] A. Maciel and D. Thérien, "Efficient threshold circuits for power series," *Information and Computation*, vol. 152, no. 1, pp. 62–73, 1999.

[41] D. Coppersmith, "Rapid multiplication of rectangular matrices," *SIAM J. Comput.*, vol. 11, no. 3, pp. 467–471, 1982.

[42] A. R. Klivans, R. O'Donnell, and R. A. Servedio, "Learning intersections and thresholds of halfspaces," *J. Comput. Syst. Sci.*, vol. 68, no. 4, pp. 808–840, 2004.

[43] N. J. A. Harvey, J. Nelson, and K. Onak, "Sketching and streaming entropy via approximation theory," in *FOCS*, 2008, pp. 489–498.

[44] A. Ambainis, A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang, "Any and-or formula of size n can be evaluated in time $n^{1/2+o(1)}$ on a quantum computer," *SIAM J. Computing*, vol. 39, no. 6, pp. 2513–2530, 2010.

[45] S. Sachdeva and N. K. Vishnoi, "Faster algorithms via approximation theory," *Theoretical Computer Science*, vol. 9, no. 2, pp. 125–210, 2013.

[46] N. Linial and N. Nisan, "Approximate inclusion-exclusion," *Combinatorica*, vol. 10, no. 4, pp. 349–365, 1990.

[47] A. A. Sherstov, "Approximate inclusion-exclusion for arbitrary symmetric functions," in *CCC*, 2008, pp. 112–123.

[48] J. P. Schmidt, A. Siegel, and A. Srinivasan, "Chernoff-Hoeffding bounds for applications with limited independence," *SIAM J. Discrete Mathematics*, vol. 8, no. 2, pp. 223–250, 1995.

[49] M. Hirvensalo, "Studies on boolean functions related to quantum computing," Ph.D. dissertation, University of Turka, 2003.

[50] G. Szegö, *Orthogonal Polynomials*. American Mathematical Society, 1975.

[51] P. L. Chebyshev, "Sur l'interpolation," in *Oeuvres de P. L. Tchebychef*, A. Markoff and N. Sonin, Eds. Commissionaires de L'Académie Impériale des Sciences, 1899, vol. 1, pp. 539–560.

[52] A. Andoni, "Approximate nearest neighbor problem in high dimensions," Master's thesis, MIT, 2005.

[53] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, "Efficient search for approximate nearest neighbor in high dimensional spaces," *SIAM J. Computing*, vol. 30, no. 2, pp. 457–474, 2000.

[54] J. Matoušek, "On variants of the Johnson–Lindenstrauss lemma," *Random Struct. Algorithms*, vol. 33, no. 2, pp. 142–156, 2008.

[55] N. Ailon and B. Chazelle, "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors," *SIAM J. Comput.*, vol. 39, no. 1, pp. 302–322, 2009.

[56] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *STOC*, 1998, pp. 604–613.