Total Search Problems in ZPP

Noah Fleming
Lund University, Sweden
Columbia University, New York, NY, USA

Stefan Grosser
McGill University, Montreal, Canada

Siddhartha Jain
UT Austin, TX, USA

Jiawei Li
UT Austin, TX, USA

Hanlin Ren
Institute for Advanced Study, Princeton, NJ, USA

Morgan Shirley
Lund University, Sweden

Weiqiang Yuan
EPFL, Lausanne, Switzerland

Abstract

We initiate a systematic study of TFZPP, the class of total NP search problems solvable by polynomial time randomized algorithms. TFZPP contains a variety of important search problems such as Bertrand-Chebyshev (finding a prime between N and $2N$), refuter problems for many circuit lower bounds, and Lossy-Code. The Lossy-Code problem has found prominence due to its fundamental connections to derandomization, catalytic computing, and the metamathematics of complexity theory, among other areas.

While TFZPP collapses to FP under standard derandomization assumptions in the white-box setting, we are able to separate TFZPP from the major TFNP subclasses in the black-box setting. In fact, we are able to separate it from every uniform TFNP class assuming that NP is not in quasi-polynomial time. To do so, we extend the connection between proof complexity and black-box TFNP to randomized proof systems and randomized reductions.

Next, we turn to developing a taxonomy of TFZPP problems. We highlight a problem called Nephew, originating from an infinity axiom in set theory. We show that Nephew is in PWPP ∩ TFZPP and conjecture that it is not reducible to Lossy-Code. Intriguingly, except for some artificial examples, most other black-box TFZPP problems that we are aware of reduce to Lossy-Code:

- We define a problem called Empty-Child capturing finding a leaf in a rooted (binary) tree, and show that this problem is *equivalent* to Lossy-Code. We also show that a variant of Empty-Child with "heights" is complete for the intersection of SOPL and Lossy-Code.
- We strengthen Lossy-Code with several combinatorial inequalities such as the AM-GM inequality. Somewhat surprisingly, we show the resulting new problems are still reducible to Lossy-Code. A technical highlight of this result is that they are proved by *formalizations in bounded arithmetic*, specifically in Jeřábek's theory APC_1 (JSL 2007).
- Finally, we show that the Dense-Linear-Ordering problem reduces to Lossy-Code.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Proof complexity

Keywords and phrases TFNP, lossy code, randomized proof systems, query complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.60

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2025/204/

Funding *Noah Fleming*: Supported by an NSERC Discovery grant and the Swedish Research Council under grant number 2025-06762.
Stefan Grosser: Supported by the NSERC CGS D fellowship.
Siddhartha Jain: Supported by Scott Aaronson's Berkeley CIQC grant and an Amazon AI Fellowship.
Jiawei Li: Supported by Scott Aaronson's Open Philanthropy grant.
Morgan Shirley: Supported by NSERC and by Knut and Alice Wallenberg grant KAW 2023.0116.
Weiqiang Yuan: Supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00026.

Acknowledgements We thank Robert Robere, Yuhao Li, and Ben Davis for extensive discussions about the Nephew problem and TFZPP. As well, we thank the reviewers for suggestions which improved the presentation of this paper.

1 Introduction

Total search problems are abundant in theoretical computer science. The formal study of these problems has been highly impactful to a wide range of areas including game theory [24, 19], cryptography [43, 30, 6], proof complexity [5, 25, 62, 38, 81, 35, 61, 28], and recently in the study of explicit construction problems and derandomization [51, 53, 55]. Central to the latter has been the *Range Avoidance* (or simply Avoid) problem.

Avoid. Given a circuit $D : \{0,1\}^{n-1} \to \{0,1\}^n$, find $x \in \{0,1\}^n$ such that for every $y \in \{0,1\}^{n-1}$, $D(y) \neq x$.

Avoid captures the explicit construction problems for many combinatorial objects whose existence follows from the probabilistic method. Notable examples include functions with high circuit complexity, rigid matrices, Ramsey graphs, strong error correcting codes, and many more [53, 47, 37, 31]. By developing algorithms for Avoid, a line of work has shown circuit lower bounds against a variety of classes [77, 20, 14, 63].

Avoid belongs to the class $\mathsf{TF}\Sigma_2^P$, the second level of the total function polynomial hierarchy. If one *Herbrandizes*[1] the Avoid problem, then one obtains its TFNP sibling, the Lossy-Code problem (see [55] for a survey). This problem asks to find an element that is not in the range of a pair of compressing and decompressing maps C and D.

Lossy-Code. Given a pair of circuits $C : \{0,1\}^n \to \{0,1\}^{n-1}$ and $D : \{0,1\}^{n-1} \to \{0,1\}^n$, find $x \in \{0,1\}^n$ such that $D(C(x)) \neq x$.

Lossy-Code was originally defined by Jeřábek in [47, 48], under the name *retraction weak pigeonhole principle*, showing that it is equivalent to the set of problems whose totality is provable in APC_1. Since then, it has been considered predominantly through the lens of bounded arithmetic as a TFNP problem and as a combinatorial principle [66, 52]. Korten [54] asked to understand the set of TFNP problems that are reducible to Lossy-Code. Besides being an interesting problem on its own, Lossy-Code also arises naturally in a few other places, further motivating its study:

- **Derandomization.** In the recent *certified derandomization* framework [75], the derandomization algorithm is required to either output the correct answer, or report that the underlying circuit lower bound assumption is false by providing a small circuit violating the assumption. It turns out that certified derandomization is characterized by Lossy-Code [60].
  Such derandomization ideas are particularly explored in the context of *catalytic computing* [9, 65] in a framework known as "compress-or-random" [74, 23, 57, 1]: If the contents of the catalytic tape is "incompressible" (which usually means that it is not a solution of a certain Lossy-Code instance), then it can be used for derandomization; otherwise we can compress the catalytic tape and obtain more free space. As a result, although we are currently unable to prove that CL (catalytic logspace) is in P, we can show that CL reduces to Lossy-Code [23].

[1] Herbrandization is a basic construction in logic; see Appendix A in the full version for a short overview.

- **Metamathematics of complexity theory.** It turns out that (variants of) Lossy-Code captures the complexity of many *refuter* problems [15], which are natural total search problems reflecting the metamathematical complexity of proving lower bounds. Many lower bounds in circuit complexity and communication complexity have refuter problems equivalent to Lossy-Code [54, 16], and the refuter complexity for some proof complexity lower bounds is captured by variants of Lossy-Code as well [61].
- **Bounded arithmetic.** A basic theory of bounded arithmetic for approximate counting and reasoning about randomized computation is APC_1, developed in a series of papers by Jeřábek [45, 46, 47]. Wilkie's witnessing theorem [80, 45] implies that Lossy-Code is "complete" for APC_1 in the following sense: APC_1 proves the totality of Lossy-Code, and every TFNP problem provably total in APC_1 reduces to Lossy-Code.

Lossy-Code belongs to the class TFZPP, the subclass of TFNP containing the total search problems that admit polynomial-time randomized algorithms, introduced in [10]. Since we are dealing with total NP search problems, every randomized algorithm that may make mistakes can be turned into one that does not make any mistakes.[2] Hence, it seems that TFZPP is the only natural (semantic) subclass of TFNP capturing randomized polynomial time.

Besides Lossy-Code, there are a variety of important total search problems that sit inside TFZPP. We list two of them that we think reflect the importance of TFZPP the best:

▶ **Example 1.** The Bertrand–Chebyshev theorem states that for every integer $N \geq 1$, there is a prime number between N and $2N$. This motivates the following total search problem called Bertrand-Chebyshev: Given an integer N (represented in binary), output a prime number between N and $2N$. In fact, the Prime Number Theorem implies that there are $\Theta(N/\log N)$ such prime numbers, and the AKS primality test [2] provides a deterministic method for verifying solutions, hence Bertrand-Chebyshev is in TFZPP.

▶ **Example 2.** A family of important total search problems is *refuter problems* [17, 15, 18] for complexity lower bounds. Let $\mathcal{C}$ be a circuit class and L be a hard problem for $\mathcal{C}$, the refuter problem, Refuter($L \notin \mathcal{C}$), is the following total search problem: given a small $\mathcal{C}$ circuit C attempting to compute L, the goal is to output an instance x such that $C(x) \neq L(x)$. The complexity of these refuter problems are closely related to the provability of complexity lower bounds [16, 61].
Such refuter problems are often in TFZPP: In fact, if L is *average-case hard* against $\mathcal{C}$ (and both $\mathcal{C}$ and L are in polynomial-time), then Refuter($L \notin \mathcal{C}$) $\in$ TFZPP as the algorithm for the refuter problem can repeatedly sample inputs from the hard distribution until it finds a solution x where $C(x) \neq L(x)$. Even though average-case lower bounds against $\mathsf{AC}^0[p]$ circuits have been proved for nearly 40 years [76, 79], we are not aware of any non-trivial TFNP upper bound for the problem Refuter(MAJ $\notin \mathsf{AC}^0[2]$).

Finally, an additional motivation for studying TFZPP is its connection to Avoid and APEPP (the class of total search problems mapping reducible to Avoid [51]): it is the "projection" of Avoid to TFNP in the following sense:

▶ **Theorem 3.** $\mathsf{TFZPP} = \mathsf{TFNP} \cap \mathsf{APEPP}$.

[2] This was observed by Jeřábek [49]; in his terminology, we have $\mathsf{TFRP} = \mathsf{TFZPP}$.

Our Contributions

In this work, we initiate a formal study of TFZPP as a class of total search problems. Analogous to the setting of decision problems, we expect that TFZPP = FP. Indeed, this follows from the same assumption used in [44] – namely that E requires circuits of exponential size. Moreover, TFZPP = FP appears to be weaker than a full derandomization of BPP.

However, we show that this is not the case in the *black-box* setting in a very strong sense. In the black-box setting one only has access to the input via an oracle; black-box classes are denoted by a *dt* superscript (for "*d*ecision *t*rees"). We say that a TFNP^{dt} class is *uniformly generated* if it has a complete problem $R = \{R_n\}_{n\in\mathbb{N}}$ such that there is a Turing Machine which on input 1^n outputs R_n in polynomial time. Note that all of the major TFNP^{dt} subclasses in the literature are uniformly generated. Under the assumption that NP is not in quasi-polynomial time (QP), we show that no uniformly generated TFNP^{dt} class contains TFZPP^{dt}.

▶ **Theorem 4.** $\mathsf{TFZPP}^{dt} \not\subseteq \mathcal{C}$ *for every uniformly generated class* $\mathcal{C} \subseteq \mathsf{TFNP}^{dt}$*, unless* $\mathsf{NP} \subseteq \mathsf{QP}$*.*

To prove these separations, we employ a close connection between total search problems and proof complexity [5, 35, 11, 28]. This connection shows that, in the black-box setting, a search problem belongs to a class if and only if an associated proof system can prove the totality of that search problem. In this case, we say that the class is *characterized* by that proof system. To prove our separations, we first show that TFZPP^{dt} is characterized by the random tree-like resolution proof systems of Buss et al. [13].

▶ **Theorem 5.** TFZPP^{dt} *is characterized by random tree-like resolution.*

More generally, we show that if a class $\mathcal{C}$ of total search problems is characterized by a proof system Π, then the class of problems that are efficiently *randomized* reducible to a complete problem in $\mathcal{C}$ is characterized by the proof system random Π. Theorem 4 then follows by combining the following two results: (1) Buss et al. [11] showed that every uniformly generated TFNP class has a characterizing proof system, and (2) Pudlák and Thapen [73] showed that a propositional proof system simulating random tree-like resolution would imply faster algorithms for NP.

A Highly Unsatisfiable Cook-Reckhow Program

Theorem 4 is striking as it suggests that TFZPP problems can be arbitrarily hard in the black-box model. This motivates an interesting direction of research: find explicit TFZPP problems that are hard for stronger and stronger subclasses of TFNP^{dt}. By the close connection between TFNP^{dt} and proof complexity [11], this can be seen as a Cook-Reckhow program for highly unsatisfiable formulas: for increasingly more expressive proof systems, exhibit a highly unsatisfiable CNF formula which is hard for that system.

Towards this program, we provide explicit separations of TFZPP^{dt} from every major TFNP class defined in the 1990s [50, 68]. Note that all those TFNP classes are contained in PLS, PPP, and PPA. The separation from PPA was shown by Beame et al. [5]. Leveraging the recent work of Hopkins and Lin [41], we are able to show the following.

▶ **Theorem 6.** *There are* explicit *(polynomial-time constructable) problems in* TFZPP^{dt} *which are not in either* PPP^{dt} *or* PLS^{dt}*.*

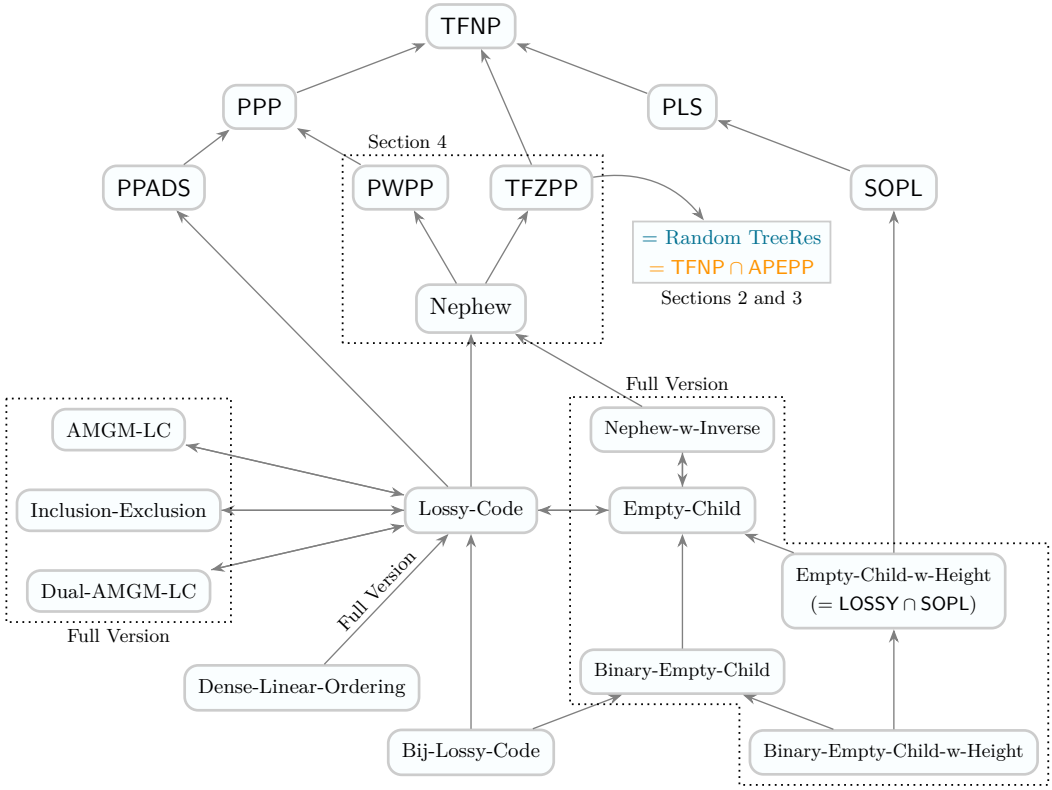


Figure 1 The TFZPP zoo.

A TFZPP Zoo

We now turn to studying the structure of problems inside of TFZPP. Figure 1 shows the zoo of problems within TFZPP that we consider, as well as their relationships.

Like ZPP, TFZPP is a semantic class and therefore it is unlikely to admit complete problems unless FP = TFZPP. However, we observe an interesting phenomenon: almost every TFZPP problem that has been studied in the literature is reducible to Lossy-Code![3] This raises the question: are there "natural" TFZPP problems which are not reducible Lossy-Code in the black-box setting, and what do they look like? Theorem 4 and Theorem 6 already provide examples of problems that are not reducible to Lossy-Code; however, we do not consider these problems natural – we are looking for problems that would be studied outside of the context of proving such separations.

While we are unable to resolve this question – indeed, many of our conjectured separating examples turned out to be reducible to Lossy-Code in surprising ways! – we provide natural TFZPP problems which we conjecture witness a separation, and which we believe are of independent interest. As well, we show several surprising reductions to Lossy-Code.

[3] One exception is the Bertrand–Chebyshev problem for which it is not known whether it is reducible to Lossy-Code. However, it is unclear how to define the Bertrand–Chebyshev problem in the *black-box* model, and we are unable to separate any natural black-box TFZPP problem from Lossy-Code.

Nephew

Our primary candidate is the following.

Nephew. Given a set V of vertices and two functions $f : V \to V$ and $g : V \to V$. Think of $f(v)$ as the *father* of v and $g(v)$ as the *nephew* of v. A solution is one of the following.
s1. $v \in V$ such that $f(f(g(v))) \neq f(v)$ (your nephew's grandparent is not your parent)
s2. $v \in V$ such that $f(g(v)) = v$ (you are your nephew's parent)

It may not be immediately obvious that Nephew is a total search problem. A proof may be found in the textbook of Börger, Grädel, and Gurevich [8, Proposition 6.5.5]; we sketch the argument here. Create a directed graph G_f with vertex set V and with an edge from u to v if $f(u) = v$. Then one can assign to each vertex a "level" that represents its distance to the core[4] of G_f. Let v^* be a vertex with maximum level $\ell_{\max}$. However, if v^* is not a solution to Nephew, it must be that $g(v^*)$ has level $\ell_{\max} + 1$, a contradiction. A similar intuition will be used in Section 4 for other proofs involving Nephew.

Nephew is derived from the minimal *axioms of infinity* in model theory. One method for constructing a total search problem is to begin with a sentence in logic that has an infinite model but no finite model. Such a sentence is known as an *axiom of infinity*, since any model for it must be infinite. Axioms of infinity are classified under the number of quantifiers and predicate and function symbols of certain arity that they contain, and there are 10 minimal classes ([8, Theorem 6.5.4]). Each of these classes corresponds to a total search problem, and in most cases, this problem is complete for a well-known TFNP class. The only exception is the one to which Nephew belongs; Nephew can be interpreted as the Herbrandization of

$$\forall x \exists y \left(F(F(y)) = F(x) \wedge F(y) \neq x\right). \tag{1}$$

The proof that Nephew belongs to TFZPP is highly non-trivial. Furthermore, our best upper bound on the complexity of Nephew is that it is contained within PWPP, the problems reducible to the weak pigeonhole principle, a relaxation of the Lossy-Code problem.

▶ **Theorem 7.** Nephew $\in$ TFZPP $\cap$ PWPP.

TFZPP Problems Studied in the Full Version

We define a number of natural problems in TFZPP which may be of independent interest. These problems are studied in detail in the full version of this paper.

Empty Child

The proof that Nephew is in TFZPP proceeds by reducing it to the observation that a leaf in a binary tree can be found in logarithmic time in expectation. We define a total search problem whose membership in TFZPP formalizes this observation.

Empty-Child. Given a set V of vertices and three functions $F, L, R : V \to V$, where $F(u)$ is the father of u, and $L(u), R(u)$ are the left and the right child of u respectively, a solution is one of the following.
s1. $u \in V$ such that $F(L(u)) \neq u$ or $F(R(u)) \neq u$ or $L(u) = R(u) \neq u$; (Empty child)
s2. 1, if $L(1) = 1$ or $R(1) = 1$ or $F(1) \neq 1$. (Wrong root)

[4] We intentionally leave "core" undefined for this brief sketch.

Surprisingly, Empty-Child is equivalent to Lossy-Code under decision tree reductions, denoted $=_{dt}$. Thus, if Nephew is indeed not reducible to Lossy-Code, the hardness of Nephew does not come from this portion of the reduction.

▶ **Theorem 8.** Empty-Child $=_{dt}$ Lossy-Code.

As a warm-up to the techniques needed to prove this theorem, we consider a variant of Empty-Child which includes an additional "height" function that outputs the height of a given node in the tree. We show that this is a complete problem for the class $\mathsf{LOSSY} \cap \mathsf{PLS} = \mathsf{LOSSY} \cap \mathsf{SOPL}$, where LOSSY is the class of problems efficiently reducible to Lossy-Code. The proof resembles previous intersection theorems from TFNP [26, 34].

Then, we use Empty-Child as an intermediate problem to study the relationship between Nephew and Lossy-Code.

▶ **Theorem 9.** Empty-Child $\leq_{dt}$ Nephew.

Thus, combining with Theorem 8, we have Lossy-Code reduces to Nephew.

AM-GM Lossy-Code

One of our original (and failed) candidates for separation from Lossy-Code was a problem called *AM-GM Lossy-Code*, obtained by combining Lossy-Code itself with the *AM-GM Inequality*: $\frac{a+b}{2} \geq \sqrt{ab}$. This problem was inspired by the Bad 2-Coloring problem in [70]: Given an undirected graph $G = (V, E)$ with $|V| = 2N$ vertices and $|E| = N^2 + 1$ edges along with a 2-coloring $C : V \to \{0, 1\}$ of V, find an edge $(x, y) \in E$ that is not colored properly. This problem is total exactly because of the AM-GM inequality: suppose there are a black vertices and $b = 2N - a$ white vertices and every edge is colored properly, then there are at most $ab \leq \left(\frac{a+b}{2}\right)^2 = N^2$ edges, contradicting $|E| > N^2$.

To compose this problem with Lossy-Code, we need to make two adaptations: First, to put it inside TFZPP, the number of edges needs to be *much* larger than N^2, say $(1 + \varepsilon)N^2$; second, we are given a function F from $[(1 + \varepsilon)N^2]$ to the set of properly colored edges and its purported inverse G and we need to find $x \in [(1 + \varepsilon)N^2]$ such that $G(F(x)) \neq x$. We arrive at the following problem:

c-**AMGM-LC.** Let $c > 1$ be a constant, $V := [2N]$ and $P := [c \cdot N^2]$. The input is a coloring function $C : V \to \{0, 1\}$ and two mappings $F : P \to V \times V$, $G : V \times V \to P$. Let $H := C^{-1}(0) \times C^{-1}(1)$. The goal is to find solutions of either type:

- **s1.** a pigeon $x \in P$ such that $G(F(x)) \neq x$; (Wrong Encoding-Decoding)
- **s2.** a pigeon $x \in P$ such that $F(x) \notin H$; (Invalid Hole)

Indeed, it seems unclear how to massage a Lossy-Code instance of the shape $[cN^2] \rightleftharpoons H$ into a standard Lossy-Code instance of the form $[2M] \rightleftharpoons [M]$, even though the AM-GM inequality implies that $cN^2 \gg |H|$. However, it turns out that such a reduction is possible (although highly non-trivial)! We encourage the reader to take a moment to think about how to reduce AMGM-LC to Lossy-Code.

▶ **Theorem 10.** *For every constant $c > 1$,* c-AMGM-LC $\leq_{dt}$ Lossy-Code.

Our reduction uses *bounded arithmetic*: we formalize the totality of c-AMGM-LC in Jeřábek's theory APC_1 [47], and Wilkie's witnessing theorem for APC_1 [80, 45] implies a reduction from c-AMGM-LC to Lossy-Code. In particular, like every formalization in APC_1, our reduction makes use of the *Nisan–Wigderson generator* [67].

Moreover, the techniques underlying APC_1 allow us to reduce problems to Lossy-Code in a *systematic* way; we provide two additional examples (c-Dual-AMGM-LC and a problem capturing the Inclusion-Exclusion principle). A secondary goal of expounding these reductions is to introduce the ideas of APC_1 to audiences who are less familiar with bounded arithmetic.

▶ Remark 11. Unfortunately, it seems unclear how to formalize the totality of Nephew in APC_1, hence the bounded arithmetic approach does not seem to provide a reduction from Nephew to Lossy-Code. For example, our proof that Nephew $\in \mathsf{TFZPP}$ requires reasoning about the *level* of each node, which seems to be global reasoning that is infeasible in APC_1.

Linear Ordering

Finally, we consider one more natural problem in TFZPP. The Linear Ordering Principle has a storied history in proof complexity [59, 7, 71] and bounded arithmetic [21, 39, 13, 4]. It has been studied in the context of total search problems as well [56, 40], where it was used in order to construct new algorithms for Avoid. A line of works in proof complexity [78, 3, 36, 22] has also considered a dense variant of this problem defined as follows, which lies in TFZPP.

Dense-Linear-Ordering. The input consists of the descriptions of a linear ordering $\prec$ over N elements and a *median function* $\mathsf{med} : [N] \times [N] \to [N]$. Without loss of generality, we may assume that for $x \neq y \in [N]$, exactly one of $(x \prec y)$ and $(y \prec x)$ is true, and that $\mathsf{med}(x, y) = \mathsf{med}(y, x)$. (That is, $\prec$ is represented by a string of $\binom{N}{2}$ bits and med is represented by a list of $\binom{n}{2}$ elements in $[N]$.) A solution is one of the following.

s1. $x, y, z \in [N]$ such that $x \prec y$, $y \prec z$, and $z \prec x$; or (Transitivity violation)

s2. $x, y \in [N]$ such that $x \prec y$, but neither $x \prec \mathsf{med}(x, y)$ nor $\mathsf{med}(x, y) \prec y$. (Invalid median)

While Avoid reduces to the Linear Ordering Principle [56], we show a converse in the dense setting.

▶ **Theorem 12.** Dense-Linear-Ordering $\leq_{dt}$ Lossy-Code.

2 Preliminaries

2.1 Basics of TFNP

TFNP contains all search problems which are (i) total: a solution is guaranteed to exist, and (ii) in NP: there is an efficient procedure to check whether a candidate solution is valid. It is believed that TFNP does not admit complete problems [72] and much of the research in this area has focused on studying syntactic subclasses (those with complete problems) which capture many of the total search problems of interest. These classes are typically defined by simple existence principles that capture the totality of the problems within that class. These naturally give rise to total search problems. For example, PWPP is the class of all search problems whose totality is witnessed by the existence principle: any map from $2N$ to N must have a collision [49]. To make these problems non-trivial, the input is presented succinctly as a circuit C that on input i outputs the i-th bit of the search problem. For example, the existence principle for PWPP gives rise to the following (white-box) total search problem.

Weak-Pigeon. Given $P : \{0,1\}^n \to \{0,1\}^{n-1}$, a solution is $x \neq y$ such that $P(x) = P(y)$.

PWPP is then the class of all total search problems which are efficiently reducible to Weak-Pigeon.

A major thrust of this line of work is to understand the relationships between these classes. However, a separation between classes would imply $\mathsf{P} \neq \mathsf{NP}$. As a proxy, and as natural objects in their own right, researchers have studied total search problems in the *black-box* model. In this setting, the input C is given as a black box which can be queried, but we no longer have access to the description of C.

In this setting a *(query) search problem* is a sequence of relations $R_n \subseteq \{0,1\}^n \times \mathcal{O}_n$, for each $n \in \mathbb{N}$. It is total if for every $x \in \{0,1\}^n$ there is $o \in \mathcal{O}_n$ such that $(x, o) \in R_n$. We think of the input $x \in \{0,1\}^n$ as being accessed by querying the individual bits, and we will measure the complexity of solving R_n as the number of bits that must be queried to determine some suitable $o \in \mathcal{O}_n$. An efficient algorithm is one that makes at most $\mathrm{poly}(\log n)$-many queries[5]; these problems belong to the class FP^{dt}, where dt indicates that it is the black-box version of the class. Similarly, $R \in \mathsf{TFNP}^{dt}$ if for every $n \in \mathbb{N}$ and each $o \in \mathcal{O}_n$ there exists a $\mathrm{poly}(\log n)$-depth decision tree $T_o : \{0,1\}^n \to \{0,1\}$ such that $T_o(x) = 1$ iff $(x, o) \in R_n$. While search problems are formally defined as a sequence $R = (R_n)_{n\in\mathbb{N}}$, we will often want to speak about individual members of this sequence. For readability, we will abuse notation and refer to elements R_n in the sequence as total search problems. Furthermore, we will often drop the subscript n and rely on context to differentiate.

We compare the complexity of total search problems by reductions between them; the following is the black-box (decision tree) analogue of a deterministic polynomial-time reduction between search problems.

▶ **Definition 13.** *For total search problems $R \subseteq \{0,1\}^n \times \mathcal{O}_n$ and $S \subseteq \{0,1\}^m \times \mathcal{O}'_m$, there is an S-formulation of R if for every $i \in [m]$ and $o \in \mathcal{O}'_m$ there are functions $f_i : \{0,1\}^n \to \{0,1\}$ and $g_o : \{0,1\}^n \to \mathcal{O}_n$ such that*

$$(f(x), o) \in S) \implies (x, g_o(x)) \in R, \tag{2}$$

where $f(x) := (f_1(x), \ldots, f_m(x))$. The depth *of the S-formulation is*

$$d := \max\left(\{\mathsf{depth}(f_i) : i \in [m]\} \cup \{\mathsf{depth}(g_o) : o \in \mathcal{O}'_m\}\right),$$

where $\mathsf{depth}(f)$ denotes the minimum depth of any decision tree which computes f. The size *of the S-formulation is m, the number of input bits to S. The* complexity *of an S-formulation is $\log m + d$ and the complexity of reducing R to S is the minimum complexity of any S-formulation of R.*

This definition extends to sequences naturally. If $S = (S_n)$ is a sequence and R_n is a single search problem, then the complexity of reducing R_n to S is the minimum over m of the complexity of reducing R_n to S_m. For two sequences $S = (S_n)$ and $R = (R_n)$, the complexity of reducing R to S is the complexity of reducing R_n to S for each n. We say that a reduction from R to S is efficient *if its complexity is $\mathrm{poly}(\log(n))$ and denote this by $R \leq_{dt} S$.*

2.2 TFZPP

In this work, we will be particularly interested in the total search problems which are solvable in *randomized* polynomial time. Formally, $R \subseteq \{0,1\}^* \times \{0,1\}^* \in \mathsf{TFZPP}$ if there is a distribution $\mathcal{D}$ over polynomial-time Turing Machines A with range $\{0, 1, \perp\}$, such that $\Pr_{A\sim\mathcal{D}}[A(x) = \perp] \leq 1/3$ and

[5] As the input is succinctly encoded, this corresponds to looking at a polynomial part of the entire input.

TFZPP is defined semantically and it is unlikely to have complete problems. However, we show that it is exactly the TFNP problems in the $\mathsf{TF}\Sigma_2^P$ class APEPP, where APEPP is the class of total search problems that are reducible to Avoid, as defined in [51].

▶ **Theorem 3.** $\mathsf{TFZPP} = \mathsf{TFNP} \cap \mathsf{APEPP}$.

Proof. Let $L \in \mathsf{TFNP} \cap \mathsf{APEPP}$. This means that given an instance x of L, there are deterministic polynomial-time algorithms V, C, and R, such that:

- V **is a TFNP verifier for** L**.** For every string z of length polynomial in $|x|$, $V(x,z) = 1$ if and only if z is a valid solution for x.
- (C,R) **is a reduction from** x **to** Avoid. The output of $C(x)$ is a circuit C_x mapping ℓ input bits to $\ell+1$ output bits, where $\ell \le \mathrm{poly}(|x|)$; given any $y \in \{0,1\}^{\ell+1} \setminus \mathrm{Range}(C_x)$, $R(x,y)$ outputs a valid solution for x.

Then we can solve L in TFZPP via the following procedure. Guess $y \in \{0,1\}^{\ell+1}$ uniformly at random and compute $z := R(x,y)$. If $V(x,z)$ accepts, then we output z; otherwise, we output $\perp$. By the correctness of V, if we did not output $\perp$, then our output is a valid solution of x. On the other hand, since at least a $1/2$ fraction of strings $y \in \{0,1\}^{\ell+1}$ are valid outputs of Avoid on the instance C_x, by the correctness of (R,C), we will output a valid solution w.p. at least $1/2$.

Now we prove the converse direction. If $L \in \mathsf{TFZPP}$ then clearly $L \in \mathsf{TFNP}$; hence we only need to show that there is a mapping reduction from L to Avoid. Let $U(x,r)$ be the zero-error randomized algorithm for L, i.e., $U(x,r)$ outputs a valid solution for x w.p. at least $1/2$ over its randomness r, and it outputs $\perp$ whenever it fails to output a valid solution.

By standard results in derandomization [67, 44, 82], there exist absolute constants $c, d \ge 1$ and a deterministic polynomial-time algorithm PRG such that the following holds. For every truth table f of length s^{10c}, if the circuit complexity of f is at least s^c, then $\mathsf{PRG}(f)$ outputs a list of s^d strings that $(1/s^2)$-fools every size-s^2 circuit. That is, for every circuit $C : \{0,1\}^s \to \{0,1\}$ of size at most s^2,

$$\left| \Pr_{x \sim \{0,1\}^s}[C(x) = 1] - \Pr_{x \sim \mathsf{PRG}(f)}[C(x) = 1] \right| \le 1/s^2.$$

Now, let $s \le \mathrm{poly}(|x|)$ be the circuit complexity of U. Consider the *truth table generator* $\mathsf{TT} : \{0,1\}^{O(s^c \log s)} \to \{0,1\}^{s^{10c}}$ that takes the description of a size-s^c circuit $C : \{0,1\}^{10c \log s} \to \{0,1\}$ as input and outputs the length-s^{10c} truth table of C. We treat TT as an instance for Avoid and reduce x to TT.[6]

It remains to show how to solve the instance x deterministically given a non-output of TT. Note that if $f \in \{0,1\}^{s^{10c}}$ is a non-output of TT, then the circuit complexity of f is at least s^c, hence $\mathsf{PRG}(f)$ outputs a list of $\mathrm{poly}(s)$ strings that $(1/s^2)$-fools every size-s^2 circuits. This implies that

$$\Pr_{r \sim \mathsf{PRG}(f)}[U(x,r) \ne \perp] \ge \Pr_{r \sim \{0,1\}^s}[U(x,r) \ne \perp] - 1/s^2 \ge 1/2 - 1/s^2 > 0,$$

and in particular, there exists at least one string $r \in \mathsf{PRG}(f)$ such that $U(x,r) \ne \perp$. We can solve the instance x by cycling through every $r \in \mathsf{PRG}(f)$ and outputting $U(x,r)$ whenever we encounter such a good r. ◀

Note that this equivalence holds even in the black-box model, since its proof is relativizing.

[6] An unusual aspect of this reduction is that TT does *not* depend on x!

▶ **Definition 14** (TFZPP). *A total* NP *search problem* $R \subseteq \{0,1\}^* \times \{0,1\}^*$ *is in* TFZPP *if there is a distribution* $\mathcal{D}$ *over polynomial-time algorithms* A *with output in* $\{0,1,\perp\}$ *such that:*

1. *For every* $x \in \{0,1\}^*$ *and every* $A \sim \mathcal{D}$*, if* $A(x) \neq \perp$ *then* $(x, A(x)) \in R$*,*
2. *For every* $x \in \{0,1\}^*$*,*

$$\Pr_{A\sim\mathcal{D}}[A(x) = \perp] \leq 1/3.$$

Similarly, $R \in \mathsf{TFZPP}^{dt}$ *if there is a family of distributions* $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ *over* $\operatorname{poly}\log(n)$*-depth decision trees with leaves labeled in* $\{0,1,\perp\}$*, where on input* x *we sample a decision tree* $A \sim \mathcal{D}_{|x|}$*, and* $\mathcal{D}$ *satisfies (1) and (2).*

2.3 Lossy-Code

As mentioned in the introduction, Lossy-Code is the Herbrandization of Avoid. Let $N < M$ be two parameters (think of $N \ll M$), (the black-box version of) Lossy-Code is the following problem:

Lossy-Code$_{N\to M}$. Given query access to a pair of functions $f : [N] \to [M]$ and $g : [M] \to [N]$, find $x \in [M]$ such that $f(g(x)) \neq x$.

We need the following basic fact about Lossy-Code that roughly states that the "stretch function" of Lossy-Code does not influence its complexity as long as it is in the "weak" regime. This fact and similar statements for other variants of the weak pigeonhole principle have been very useful in bounded arithmetic [69, 80, 58, 45, 47, 16], total search problems [53, 54, 61], and cryptography [33, 64].

▶ **Lemma 15.** *Let* $\varepsilon > 0$ *and* $M > (1+\varepsilon)N$*. There is a decision tree reduction of complexity* $O(\varepsilon^{-1}\log(M/N))$ *from* Lossy-Code$_{N\to(1+\varepsilon)N}$ *to* Lossy-Code$_{N\to M}$*.*

By Lemma 15, Lossy-Code$_{N\to 1.01N}$, Lossy-Code$_{N\to 2N}$, and Lossy-Code$_{N\to N^{100}}$ are equivalent up to decision tree reductions of polylog(N) depth. In this paper, unless otherwise stated, Lossy-Code always stands for Lossy-Code$_{N\to 2N}$.

We denote LOSSY as the class of total search problems reducible to Lossy-Code.[7] It follows from Lemma 15 that LOSSY is robust in the sense that it does not matter whether it is defined using Lossy-Code$_{N\to 1.01N}$ or Lossy-Code$_{N\to N^{100}}$ as the complete problem. In fact, LOSSY is *extremely* robust: it is closed under Turing reductions ($\mathsf{FP}^{\mathsf{LOSSY}} = \mathsf{LOSSY}$ [12, 60]) and it is self-low ($\mathsf{LOSSY}^{\mathsf{LOSSY}} = \mathsf{LOSSY}$ [32]).

3 Randomized Proof Complexity and Explicit Separations

We begin by describing the connection between black-box TFZPP and proof complexity, and how this can be leveraged to obtain explicit separations from other natural classes. Proof complexity is concerned with the efficient provability of propositional theorems (unsatisfiable CNF formulas) in various *proof systems* – simply a verifier for the language UNSAT of unsatisfiable CNF formulas.

[7] Previous literature [60, 23] defined LOSSY as the class of *decision problems* reducible to Lossy-Code. In our context, it is more natural to define LOSSY as a class of total search problems.

▶ **Definition 16.** *A propositional proof system is a polynomial-time machine $\mathcal{P}$ such that for every CNF formula F, $F \in$ UNSAT iff there exists a proof $\Pi \in \{0,1\}^*$ such that $\mathcal{P}(F,\Pi) = 1$. We say that Π is a $\mathcal{P}$-proof of F and define the size of Π to be $s(\Pi) := |\Pi|$.*

When studying connections between proof systems and TFNP *classes, it is standard to also consider an associated notion of the* width *of a proof $w(\Pi)$. This is typically specific to the proof system – for example, in resolution (defined next), it is the maximum number of literals in a clause in Π, while for algebraic systems such as Sum-of-Squares, the width is the degree of the polynomials occurring in the proof.*

With a definition of width, the complexity *of proving F in $\mathcal{P}$ is*

$$\mathcal{P}(F) \coloneqq \min_{\mathcal{P}\text{-proof } \Pi \text{ of } F} w(\Pi) + \log s(\Pi).$$

A standard example is the *resolution* proof system. A resolution proof of an unsatisfiable CNF formula F consists of a sequence of clauses $C_1, \ldots, C_t = \emptyset$ ending with the empty clause which contains no literals, such that each clause C_i either belongs to F or is derived from earlier clauses in the sequence according to the *resolution rule*.

▶ **Resolution rule.** *From two clauses with complementary literals $A \vee x$ and $B \vee \overline{x}$, derive $A \vee B$.*

The size of a resolution proof is the number of clauses that it contains, while the width is the maximum number of literals within any clause in the proof. A resolution proof $C_1, \ldots, C_t$ is *tree-like* if each C_i is used at most once as a premise for the resolution proof. They are named as such because the implication graph of such proofs is a tree.

There is a long line of work connecting proof complexity and black-box TFNP [5, 35, 38, 11, 28, 42, 81, 25, 62, 27]. These connections show that a total search problem is contained within a class iff an associated proof system can prove the totality of that search problem. We can phrase the totality of any total search problem $R \subseteq \{0,1\}^n \times \mathcal{O}$ as an unsatisfiable CNF formula in the following way: for each $o \in \mathcal{O}$ let V_o be a decision tree which checks whether o is a solution; that is, $V_o(x) = 1$ iff $(x,o) \in R$. A root-to-leaf path in V_o is a 1-*path* if its leaf is labeled 1. We will associate with any path p the conjunction of literals that it follows. Then the totality of R is expressed as

$$F_R \coloneqq \neg\left(\bigvee_{o \in \mathcal{O}} \bigvee_{1\text{-path } p \in V_o} p\right).$$

If $R \in \mathsf{TFNP}^{dt}$ then V_o can be assumed to have depth $\operatorname{poly}\log(n)$, and hence the width of F_R is also $\operatorname{poly}\log(n)$.

Similarly, we can associate with any unsatisfiable CNF formula $F = C_1 \wedge \ldots \wedge C_m$ a total search problem $\textsc{Search}_F \subseteq \{0,1\}^n \times [m]$ such that $(x,o) \in \textsc{Search}_F$ iff $C_o(x) = 0$. Observe that whenever F has $\operatorname{poly}(\log(n))$ width then $\textsc{Search}_F \in \mathsf{TFNP}^{dt}$ and furthermore that $\textsc{Search}_{F_R}$ is reducible to R by decision trees of depth at most the width of F_R.

For a syntactic class $\mathcal{C} \subseteq \mathsf{TFNP}^{dt}$ we will denote by $\mathcal{C}(R)$ the complexity of reducing R to S, where S is any complete problem for $\mathcal{C}$. We say that a proof system $\mathcal{P}$ is *characterized* by a class $\mathcal{C} \subseteq \mathsf{TFNP}^{dt}$ if $R \in \mathcal{C}$ iff $\mathcal{P}(F) = \operatorname{poly}(\mathcal{C}(R))$. A standard example is that FP^{dt} characterizes tree-like resolution. Said differently, decision trees are equivalent to tree-like resolution proofs.

We extend these characterizations to capture randomized reductions. We show that randomized reductions between total search problems give rise to proofs in randomized proof systems, a notion introduced by Buss, Kołodziejczyk, and Thapen [13].

▶ **Definition 17.** *Let $\mathcal{P}$ be any propositional proof system. A* randomized *$\mathcal{P}$-proof, denoted $r\mathcal{P}$, of an unsatisfiable formula F is a distribution $\mathcal{D}$ supported on pairs (Π, B), such that*

1. *Each B is a CNF formula over the variables of F,*
2. *Π is a $\mathcal{P}$ proof of $F \wedge B$,*
3. *For any assignment $x \in \{0,1\}^n$, $\Pr_{(\Pi,B)\sim\mathcal{D}}[B(x) = 1] \geq 2/3$.*

The size *$s(\mathcal{D})$, and* width *$w(\mathcal{D})$ of an $r\mathcal{P}$-proof $\mathcal{D}$ is the maximum width and size of a proof Π in the support of $\mathcal{D}$. The* complexity *of proving F in $r\mathcal{P}$ is*

$$r\mathcal{P}(F) := \min_{r\mathcal{P}\text{-proof } \mathcal{D} \text{ of } F} w(\mathcal{D}) + \log s(\mathcal{D}).$$

Note that a randomized proof system is not a Cook-Reckhow proof system in the sense of Definition 16 since its proofs typically cannot be polynomial-time verified [73].

The main theorem of this section, Theorem 19, shows that a proof system $\mathcal{P}$ is characterized by class $\mathcal{C}$ iff the totality of the total search problems R which are randomly reducible to any complete problem for $\mathcal{C}$ is provable in $r\mathcal{P}$. The following definition is equivalent to the probabilistic reduction in [49].

▶ **Definition 18.** *A randomized (ZPP) reduction from a search problem $S \subseteq [t] \times \mathcal{O}$ to $R \subseteq [n] \times \mathcal{Q}$ is a distribution $\mathcal{D}$ over deterministic reductions $\mathcal{T} = (T, \{T_o\})$ such that each output decision tree is labeled either by some $j \in \mathcal{O}$ or by $\perp$, and $\mathcal{D}$ satisfies*

1. *For every $x \in [t]$ and every $\mathcal{T} \sim \mathcal{D}$, if $(T(x), o) \in R$ then either $T_o(x) = \perp$ or $(x, T_o(x)) \in S$.*
2. *For every $x \in [t]$,*

$$\Pr_{\mathcal{T}\sim\mathcal{D}}[\exists o \in \mathcal{O} : (T(x), o) \in R \ \wedge \ T_o(x) = \perp] \leq \varepsilon$$

▶ **Theorem 19.** *If a proof system $\mathcal{P}$ is characterized by the total search problems reducible to $R \in \mathsf{TFNP}^{dt}$, then $r\mathcal{P}$ is characterized by the total search problems that are randomized-reducible to R.*

The intuition for this theorem is most clear in the case of randomized reductions to FP^{dt} (which is TFZPP^{dt}) and random tree-like resolution. This is also the case that we will use to derive consequences about TFZPP^{dt}. We leave the proof of Theorem 19 to the full version.

We remark that Theorem 19 reduces the task of showing that a TFNP class is closed under randomized reduction to showing that the corresponding proof system is *closed* in the sense that $\mathcal{P} = r\mathcal{P}$. We are not aware of any such proof system, and it would be interesting to exhibit one.

▶ **Lemma 20.** *There is a quasi-polynomial size random tree-like resolution proof of $F = C_1 \wedge \ldots \wedge C_m$ iff $\mathsf{Search}_F \in \mathsf{TFZPP}^{dt}$.*

Proof. Suppose that $\textsc{Search}_F \in \mathsf{TFZPP}^{dt}$ and let $\mathcal{D}$ be a distribution over depth-d decision trees solving $\textsc{Search}_F$ as in the definition of TFZPP^{dt}. We construct a ε-error randomized tree resolution proof $\mathcal{P}$, which is defined by the following sampling procedure:

1. Sample $T \sim \mathcal{D}$.
2. Let B be the set of clauses obtained by taking the negation of each root-to-leaf path in T ending in $\perp$,

$$B := \{\neg p : p \in T \text{ is a root-to-}\perp \text{ path}\},$$

where we think of a path as the conjunctions of the literals that appear along it (a queried variable is a positive literal if p took the 1-branch, and a negative literal if p took the 0-branch).

3. To construct Π, we will use the equivalence between a decision tree solving the false-clause search problem and tree-resolution proofs. Let T^* be obtained from T by relabeling each path p ending in $\bot$ by the clause $\neg p \in B$. Let $F \cup B$ be the CNF formula whose clauses are the clauses of F and those in B, and observe that T^* is a depth-d decision tree solving $\textsc{Search}_{F \cup B}$. Thus, there is a depth-d tree resolution proof Π of $F \cup B$.

As B states that we do not follow any root-to-$\bot$ path in T, for any $x \in \{0,1\}^n$, $T(x) \neq \bot$ iff x satisfies all of B. Therefore, $\Pr_{T \sim \mathcal{D}}[T(x) = \bot] \leq \varepsilon$ implies that $\Pr_{(B,\Pi) \sim \mathcal{P}}[B(x) = 1] \geq 1 - \varepsilon$ for every $x \in \{0,1\}^n$.

In the other direction, suppose that $\mathcal{P}$ is a ε-random tree resolution proof of F with complexity c. We construct a distribution $\mathcal{D}$ over decision trees solving $\textsc{Search}_F$ by the following sampling procedure:

1. Sample $(\Pi, B) \sim \mathcal{P}$.
2. Let T^* be the depth-d decision tree solving $\textsc{Search}_{F \cup B}$ obtained from Π obtained by the equivalence between tree resolution proofs and decision trees in the same manner as in point (3) above. It is well-known that the depth of a resolution proof is bounded by its width, and hence T^* has depth at most c.
3. Let T be the decision tree obtained from T^* by relabeling each leaf of Π that is labeled by a clause in B by $\bot$.

As for any $x \in \{0,1\}^n$, $\Pr_{(B,\Pi) \sim \mathcal{D}}[B(x) = 1] \geq 1 - \varepsilon$, we have that $\Pr_{T \sim \mathcal{D}}[T(x) = \bot] \leq \varepsilon$. ◀

3.1 Separations

We now use Lemma 20 to show that TFZPP^{dt} is not contained within any uniformly generated TFNP^{dt} class unless NP is contained within quasi-polynomial time (QP).

▶ **Theorem 21.** *$\mathsf{TFZPP}^{dt} \not\subseteq \mathcal{C}$ for any uniformly generated class $\mathcal{C} \subseteq \mathsf{TFNP}^{dt}$ unless $\mathsf{NP} \subseteq \mathsf{QP}$.*

This theorem follows by combining the characterization of uniform TFNP^{dt} classes by proof systems of Buss et al. [11], Lemma 20, and the following result of Pudlák and Thapen [73].

This separation relies on a theorem of Pudlák and Thapen [73] who showed that random resolution cannot be simulated by any propositional proof system unless $\mathsf{P} \neq \mathsf{NP}$. A straightforward examination of their theorem reveals that it also holds for tree-like resolution and can be stated in the following form.

▶ **Theorem 22** (Proposition 10 in [73])**.** *There is a family of unsatisfiable 3-CNFs $\mathcal{F}$ such that:*

1. *There are $O(\log(n))$-complexity random tree-like resolution proofs of $\mathcal{F}$.*
2. *If there is a propositional proof system which has $\operatorname{poly}\log(n)$-complexity proofs of $\mathcal{F}$ then $\mathsf{NP} \subseteq \mathsf{QP}$.*

Indeed, to prove their theorem Pudlák and Thapen observe that random (treelike) resolution has small proofs of any highly unsatisfiable formula (one for which any assignment falsifies many clauses), and that the PCP theorem can be used to put any 3-CNF formula into this form (the family $\mathcal{F}$). Using this, they show that if there existed a propositional proof system which could (quasi-polynomially) simulate random tree-like resolution, then one could use its variability to decide SAT.

Combining this theorem with Lemma 20 and the characterization of TFNP^{dt} classes by propositional proof systems due to Buss et al. [11] proves Theorem 21. Say that $R = \{R_n\} \in \mathsf{TFNP}^{dt}$ is *uniformly generated* if there is a Turing Machine which on input 1^n outputs R_n, and say that a class is uniformly generated if it has a uniformly generated complete problem. Note that all major TFNP^{dt} subclasses are uniformly generated.

Proof of Theorem 21. Let $\mathcal{C}$ be a uniformly generated class such that $\mathsf{TFZPP}^{dt} \subseteq \mathcal{C}$. Buss et al. [11] showed that every uniformly generated TFNP^{dt} subclass is characterized by a proof system; let $\mathcal{P}$ be the system for $\mathcal{C}$. Consider the family of formulas $\mathcal{F}$ from Theorem 22. As $\mathsf{TFZPP}^{dt} \subseteq \mathcal{C}$, there are poly $\log(n)$-complexity $\mathcal{P}$-proofs of $\mathcal{F}$. Hence, Theorem 22 implies that $\mathsf{NP} \subseteq \mathsf{QP}$. ◀

3.2 Explicit Separations

The separating examples in Theorem 21 rely on an unproven hypothesis. We end this section by proving explicit separations between TFZPP^{dt} and every major TFNP^{dt} subclass which do not rely on any unproven assumptions. A separation of PPA^{dt} from TFZPP^{dt} was implicitly shown by Beame et. al. [5], who proved Nullstellensatz lower bounds for Lossy-Code.[8] The remaining major TFNP^{dt} classes are contained within PLS^{dt} and PPP^{dt}. We show the following.

▶ **Theorem 23.** *There exist explicit total search problems in* TFZPP^{dt} *which are not contained in* PLS^{dt} *nor* PPP^{dt}.

It is known that if a total search problem $\textsc{Search}_F$ is in PLS^{dt} or PPP^{dt}, then F has a small low-degree *Sum-of-Squares* (SoS) proof; see [29] for an exposition on this proof system. Our hard instance is based on the recent work of Hopkins and Lin [41], who exhibited the first explicit hard 3-XOR instance for SoS.

▶ **Theorem 24** ([41])**.** *There exist constants* $\mu_1, \mu_2 \in (0,1)$ *and a polynomial time algorithm which, given* 1^n *as input, outputs a 3-XOR formula* $F = C_1 \wedge \cdots \wedge C_m$ *on* n *variables such that:*

- *For every* $x \in \{0,1\}^n$, $\Pr_{i\sim[m]}[C_i(x) = 1] \leq 1 - \mu_1$.
- *Any Sum-of-Squares refutation of* F *requires degree at least* $\mu_2 n$.

Proof of Theorem 23. Let F be as in Theorem 24. We show that $R := \textsc{Search}_F$ satisfies the desired properties. We first prove $R \in \mathsf{TFZPP}^{dt}$. Consider the following simple algorithm: sample $i \sim [m]$ uniformly at random, and make three queries to check if $C_i(x) = 0$. If so, output i; otherwise, output $\perp$. By the first item of Theorem 24, the algorithm succeeds with probability at least μ_1. Repeating the procedure $O(1/\mu_1)$ times boosts the success probability to at least $2/3$.

To separate R from PPP^{dt} and PLS^{dt}, we only need to show there is no efficient black-box reduction from R to a complete problem for one of these classes. If there was, then there would be an efficient SoS proof of F, contradicting the second item of Theorem 24. ◀

4 Nephew

Recall the Nephew problem, which is our main candidate for a total search problem in TFZPP but not reducible to Lossy-Code:

Nephew. Given a set V of vertices and two functions $f : V \to V$ and $g : V \to V$. Think of $f(v)$ as the *father* of v and $g(v)$ as the *nephew* of v. A solution is one of the following.

s1. $v \in V$ such that $f(f(g(v))) \neq f(v)$ (your nephew's grandparent is not your parent)

s2. $v \in V$ such that $f(g(v)) = v$ (you are your nephew's parent)

[8] More specifically, Beame et. al. [5, Theorem 12] proved the Nullstellensatz degree lower bounds for Weak-Pigeon. They also showed that any Nullstellensatz degree lower bounds for Weak-Pigeon implies the same Nullstellensatz degree lower bounds for Lossy-Code in [5, Lemma 10] (see also Definition 3.1, 3.2 in [5] for the definition of Lossy-Code and Weak-Pigeon).

The main result of this section is the inclusion theorem:

▶ **Theorem 7.** Nephew ∈ TFZPP ∩ PWPP.

The intuition behind Theorem 7 is as follows:

- We can treat certain vertices in a Nephew instance like the root of a directed binary tree, where the leaves of the tree correspond to solutions of the Nephew instance.
- Finding a leaf of a rooted binary tree is easy for both PWPP and TFZPP computations.

Once we have proven the above, we are nearly done. First, choose an arbitrary vertex. Perhaps that vertex is one of the many that we can treat as the root of a binary tree, and so from it we can find a solution. Otherwise, we use a procedure to find such a root vertex. In fact, we will be able to find two vertices, one of which must be a good root vertex. To show inclusion in TFZPP, we just need to pick one of these two randomly. For PWPP, we will have to consider both. We make an adjustment to the argument for a single root vertex so that it works for two potential root vertices.

To be more concrete, we will reduce Nephew to the following *promise* search problem.

Leaf-of-Rooted-Tree. An instance consists of a set V of vertices, a special vertex v^*, and two functions $L : V \to V \cup \{\perp\}$ and $R : V \to V \cup \{\perp\}$. We define a subset $V^* \subseteq V$ recursively: $v^* \in V^*$ and, for every $v \in V^*$, we add $L(v)$ to V^* if $L(v) \neq \perp$ and $R(v)$ to V^* if $R(v) \neq \perp$. We promise that the induced subgraph on V^* is a (directed) tree rooted at v^* and that for all $v \in V^*$ either:

- $L(v) = R(v) = \perp$, or
- $L(v) \neq \perp$ and $R(v) \neq \perp$ and $L(v) \neq R(v)$.

A solution is a $(\lceil \log |V| \rceil + 1)$-length path, represented by a string $p \in \{L, R\}^{\lceil \log |V| \rceil + 1}$, where starting at v^* and descending by the functions specified by the characters of p in order will at some point reach a vertex v where $L(v) = R(v) = \perp$.

Note that instead of simply asking for a leaf, we require a root-to-leaf path for a solution. This is to confirm that the leaf is in the binary tree rooted at v^*.

Finding a root-to-leaf path in a rooted binary tree

It is easy for a PWPP or TFZPP computation to find a solution to Leaf-of-Rooted-Tree. This follows from the simple observation that there are (many) more paths of length $(\lceil \log |V| \rceil + 1)$ than vertices in the tree, so most paths must be solutions.

▶ **Lemma 25.** *A solution to* Leaf-of-Rooted-Tree *can be found with high probability using a randomized algorithm.*

Proof. The algorithm guesses a random path of length $\lceil \log |V| \rceil + 1$, which will be a solution with probability at least 5/6. This is because if a path p is not a solution, then following p reaches a vertex v_p with two children. The path p is the only non-solution path of length $\lceil \log |V| \rceil + 1$ to reach v_p; furthermore, because V^* is an induced tree, no other path ends at a vertex with $L(v_p)$ or $R(v_p)$ as children. This means that the set $\{v_p, L(v_p), R(v_p)\}$ are uniquely reached by p out of all of the non-solution paths. Therefore, there are at least 3 times as many vertices in V as there are non-solution paths. Let the fraction of non-solution paths be α. Then

$$|V| \geq 3\alpha \left| \{L, R\}^{\lceil \log |V| \rceil + 1} \right| \geq 3\alpha \cdot 2|V|,$$

and so $\alpha \leq 1/6$. ◀

To show inclusion in PWPP, we reduce to the PWPP-complete problem Weak-Pigeon. (Note that this is not a reduction between TFNP problems, as Leaf-of-Rooted-Tree is a promise problem.) We recall its definition here:

Weak-Pigeon. Given $h : \{0,1\}^n \to \{0,1\}^{n-1}$ a solution is $x \neq y$ such that $h(x) = h(y)$.

▶ **Lemma 26.** Leaf-of-Rooted-Tree *reduces to* Weak-Pigeon.

Proof. Let $n = \lceil \log |V| \rceil + 1$. Let v_p be the vertex reached by p if p is a non-solution path. Then we define h as a map from $(\lceil \log |V| \rceil + 1)$-length paths to vertices:

$$h(p) = \begin{cases} v_p & \text{if } p \text{ is not a solution;} \\ v^* & \text{otherwise.} \end{cases}$$

(Recall that since v^* is the root, we have that $v_p \neq v^*$ for any path p.) Thus, paths (x, y) are a collision if and only if x and y are both solutions to Leaf-of-Rooted-Tree, and so from any collision we can find a solution to Leaf-of-Rooted-Tree by arbitrarily choosing one from the pair. ◀

The structure of Nephew instances and finding a rooted binary tree

For any Nephew instance (V, f, g), let G_f be the directed graph with vertex set V and where (u, v) is an edge if and only if $v = f(u)$. Then G_f is a directed graph with out-degree one, and therefore the connected components of G_f have a simple structure: they are composed of a cycle (perhaps a self-loop) and trees (with edges oriented leaf-to-root) that are rooted at vertices in the cycle. For any vertex $v \in V$, define its *level* (denoted $\ell(v)$) as the distance from v to any vertex on the cycle of its connected component. So, any vertex on the cycle has level 0, any non-cycle vertex pointing to a cycle vertex has level 1, and so on. See Figure 2 for an illustration.

We give a reduction from Nephew to Leaf-of-Rooted-Tree under the assumption that we can find a vertex v^* with $\ell(v^*) \geq 2$. After proving this, the hard part will be finding such a vertex. The main component of this reduction is the procedure `Find-Children` (Algorithm 1), which is based on the functions f and g from an Nephew instance. Define `CheckSol`(u) to be the procedure that returns True iff u is a solution to the Nephew instance, that is, iff $f(f(g(u))) \neq f(u)$ or $f(g(u)) = u$.

Algorithm 1 Procedure `Find-Children`$_{f,g}(v)$.

```
h(v) ← g(f(g(v)))                                   ▷ Rename for notational brevity
if CheckSol(v) ∨ CheckSol(g(v)) ∨ CheckSol(f(g(v))) ∨ CheckSol(h(v)) then
    return (⊥, ⊥)                                   ▷ We have found a solution, so we can stop here
else
    return (g(v), h(v))
```

See Figure 3a for an illustration. The idea is to construct the tree by defining the left and right children of v to be two nodes reachable from v, unless the procedure finds a nearby solution, in which case we can make v into a leaf, as it is easy to compute a solution given v.

Although the intuition behind the reduction is straightforward, some work needs to be done to show that `Find-Children` gives a valid binary tree. The following properties will help.

▶ **Lemma 27.** *Let* $\texttt{Find-Children}_{f,g}(v) = (a, b)$. *If* $(a, b) \neq (\perp, \perp)$, *then*

(i) $a \neq b$,
(ii) $f(a) \neq f(b)$,
(iii) $f(f(a)) = f(f(b)) = f(v)$, *and*
(iv) *if* $\ell(v) \geq 2$, *then* $\ell(a) = \ell(b) = \ell(v) + 1$.

Proof.

(i) Since $(a, b) \neq (\perp, \perp)$, we have that $f(g(v)) = f(a)$ is not an Nephew solution, and in particular $f(g(f(a))) \neq f(a)$. This means that $a \neq b$, as $b = g(f(a))$.
(ii) As $f(g(v))$ is not a solution, applying $f(g(u)) \neq u$ to $u = f(g(v))$ gives $f(b) = f(g(f(g(v)))) \neq f(g(v)) = f(a)$.
(iii) As v is not a solution, $f(f(a)) = f(f(g(v))) = f(v)$. As $f(g(v))$ is not a solution, applying $f(f(g(u))) = f(u)$ for $u = f(g(v))$ gives $f(f(b)) = f(f(g(f(g(v))))) = f(f(g(v))) = f(v)$.
(iv) Under the assumption that $\ell(v) \geq 2$, we know that $\ell(f(v)) = \ell(v) - 1 > 0$, which implies that any vertex with $f(f(u)) = f(v)$ has $\ell(u) = \ell(f(v)) + 2 = \ell(v) + 1$. By Item iii, this applies to a and b. ◀

▶ Remark 28. Lemma 27 iv does not hold without the hypothesis $\ell(v) \geq 2$, as if $\ell(f(v)) = 0$ there is no guarantee that vertices with edges pointing to $f(v)$ have level greater than 0. This is why it is necessary to assume $\ell(v^*) \geq 2$ for the simple reduction given here.

▶ **Lemma 29.** *Let* (V, f, g) *be an instance of* Nephew. *Then define* L *and* R *by*

$$(L(v), R(v)) \leftarrow \texttt{Find-Children}_{f,g}(v).$$

Given $v^* \in V$ *with* $\ell(v^*) \geq 2$, (V, v^*, L, R) *is a valid instance to* Leaf-of-Rooted-Tree.

Proof. All we need to show is that v^* is the root of an induced tree. Recall that V^* denotes the set of vertices reachable from v^* via L and R (i.e., using edges of the form $(u \to L(u))$ and $(u \to R(u))$). By Lemma 27 iv, the induced subgraph on V^* can be assigned levels such that edges are directed only from lower levels to higher levels, i.e. it is a DAG. By Lemma 27 i, the DAG has outdegree 2. To prove this induced DAG is indeed a tree, it suffices to show that no two vertices share the same child. Indeed, if there are vertices $u, u' \in V^*$ such that $\{L(u), R(u)\} \cap \{L(u'), R(u')\} \neq \varnothing$, then by Lemma 27 iii, we have $f(u) = f(u')$. Hence, it suffices to prove the following claim:

▷ Claim 30. If $u, u' \in V^*$ are two different vertices, then $f(u) \neq f(u')$.

Proof of Claim 30. We may assume that $\ell(u) = \ell(u')$, as otherwise $f(u) \neq f(u')$ by the observation about levels in V^*. The proof is by induction on levels. At level $\ell(v^*)$, there is only one vertex, hence the base case is trivially true. Assume that at level k (where $k \geq \ell(v^*)$), no two vertices in V^* map via f to the same vertex. We will prove that it also holds for level $k + 1$.

For two different vertices u, u' with $\ell(u) = \ell(u') = k + 1$, we may assume u and u' are the children of two different vertices in V^*: if they are children of the same vertex, by Lemma 27 ii $\{u, u'\} = \{L(w), R(w)\}$ implies $f(u) \neq f(u')$ and we are done. So, let $w \neq w'$ be such that $u \in \{L(w), R(w)\}$ and $u' \in \{L(w'), R(w')\}$. Then $\ell(w) = \ell(w') = k$ and by the inductive hypothesis $f(w) \neq f(w')$.

Assume that $f(u) = f(u')$. We will prove shortly that all vertices in V^* that map to u or u' by L or R map to some common vertex z by f, a contradiction with $f(w) \neq f(w')$. Indeed, set $z = f(f(u)) = f(f(u'))$. If a vertex $x \in V^*$ has $L(x) = g(x) = u$ (the same arguments will apply to x that maps to u'), then

$$z = f(f(u)) = f(f(g(x))) = f(x).$$

If a vertex $x \in V^*$ has $R(x) = g(f(g(x))) = u$, then

$$z = f(f(u)) = f(f(g(f(g(x))))) = f(f(g(x))) = f(x),$$

where we have applied $f(f(g(y))) = f(y)$ to $y = f(g(x))$ which we know is not a Nephew solution because it was checked by `Find-Children`. ◁

See Figure 3b for an illustration of the argument behind Claim 30. ◀

Completing the argument

It remains to find a vertex v^* with $\ell(v^*) \geq 2$. We do not know how to achieve this deterministically; instead, we will find a pair of vertices (v, v') such that *either* $\ell(v) \geq 2$ or $\ell(v') \geq 2$.

▶ **Lemma 31.** *Let $u \in V$ be an arbitrary starting vertex. Let $v = g(u)$ and $v' = g(f(u))$. Then either*

- *at least one of $\{u, v, v'\}$ is a solution, or*
- *at least one of $\{v, v'\}$ is at level at least 2.*

Proof. Assume that there are no solutions among $\{u, v, v'\}$. There are three cases:

- **Case I:** Suppose that $\ell(u) = 0$. We claim that $\ell(v) = 2$ in this case. Indeed, u is the only vertex at level 0 that maps via f to $f(u)$. Since $f(f(v)) = f(f(g(u))) = f(u)$ but $f(v) = f(g(u)) \neq u$, $f(v)$ has to be at level 1, hence v is at level 2.
- **Case II:** Suppose that $\ell(u) = 1$. In this case, we have $\ell(f(u)) = 0$, hence the same argument as above applies to show that $v' = g(f(u))$ is at level 2.
- **Case III:** Suppose that $\ell(u) \geq 2$. In this case, $f(u)$ is at level at least 1. Since $f(f(v)) = f(f(g(u))) = f(u)$, we have that v is at level at least 2. ◀

See Figures 3c and 3d for an illustration of cases I and II of Lemma 31.

Now we are ready to prove our main results. Recall the statement of Theorem 7:

▶ **Theorem 7.** Nephew $\in$ TFZPP $\cap$ PWPP.

Proof. The proof of Nephew $\in$ TFZPP is easy. Pick an arbitrary vertex u and define $v = g(u)$ and $v' = g(f(u))$. By Lemma 31, if none of u, v, v' are solutions of Nephew, then at least one of v and v' is at level at least 2. Therefore, we can randomly select one vertex in $\{v, v'\}$ as the root v^* and run the randomized algorithm for Leaf-of-Rooted-Tree on the instance (V, v^*, L, R). The correctness is guaranteed by Lemma 29.

Now we prove that Nephew $\in$ PWPP. Intuitively, we will run the reduction of Lemma 29 in parallel on (the direct product of) two graphs where the root is v in one graph and is v' in the other. Because we can only guarantee that our starting point is at level ≥ 2 in only one of these graphs, we only know that one of the graphs has an induced tree rooted at its respective starting point by the reduction process. Fortunately, the overall graph on $V \times V$ will have a rooted induced tree.

By Lemma 31, starting with an arbitrary vertex, we can obtain either a Nephew solution (in which case we are done) or a pair (v, v') where at least one is at level at least 2. We will reduce to Leaf-of-Rooted-Tree on the vertex set $V \times V$. Set

$$(A(u, u'), B(u, u')) \leftarrow \texttt{Find-Children}_{f,g}(u),$$

$$(C(u, u'), D(u, u')) \leftarrow \texttt{Find-Children}_{f,g}(u').$$

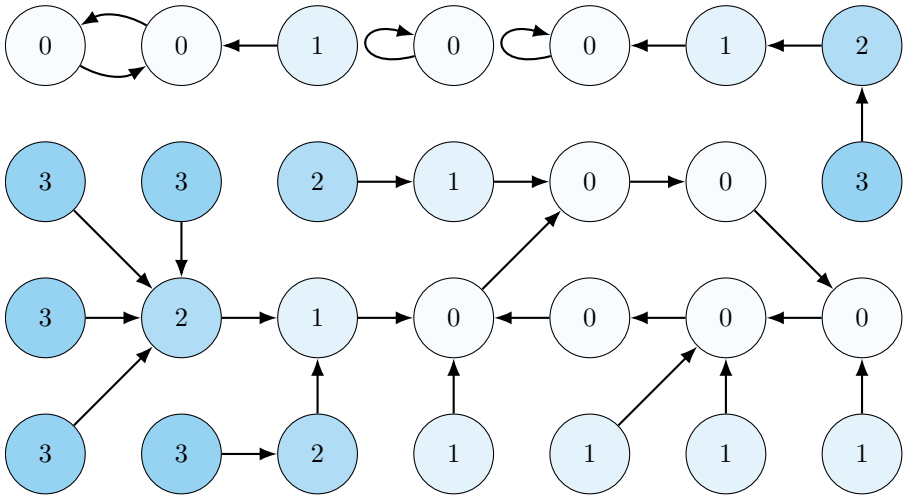
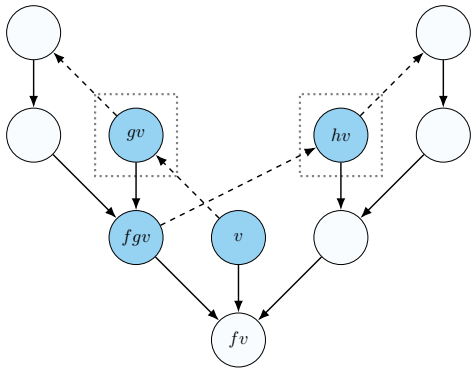
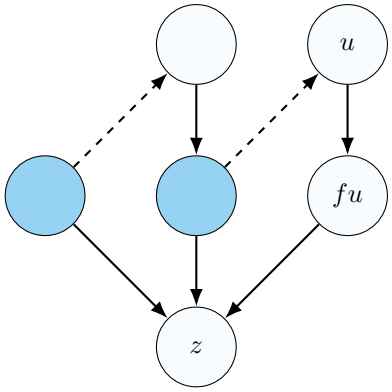


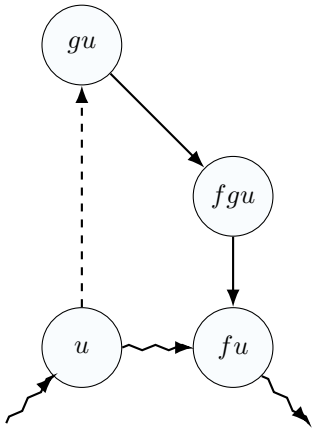
Figure 2 An example G_f with levels marked.



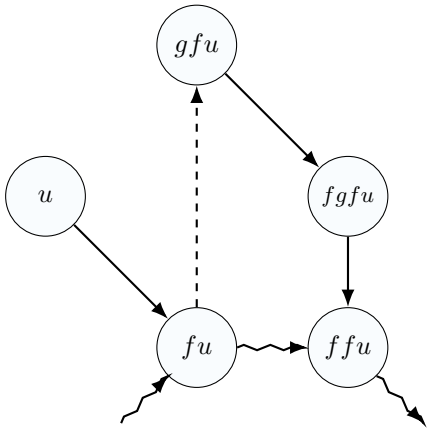
(a) `Find-Children`$_{f,g}(v)$. Shaded vertices are checked by `CheckSol`. Vertices $g(v)$ and $h(v)$ will be returned as the children of v.



(b) The argument behind Claim 30. The shaded vertices are possible locations of the $w \in V^*$ such that either $L(w) = u$ or $R(w) = u$.



(c) Case I of Lemma 31. If $\ell(u) = 0$, then $\ell(g(u)) = 2$.



(d) Case II of Lemma 31. If $\ell(u) = 1$, then $\ell(g(f(u))) = 2$.

Figure 3 Illustrations of NEPHEW substructures from proofs in this section. Solid arrows represent f and dashed arrows represent g. Zigzag f arrows are between vertices of level 0. Parentheses are omitted in labels.

Set $L(u,u') = (A(u,u'), C(u,u'))$, unless one of $A(u,u'), C(u,u')$ is $\perp$, in which case $L(u,u') = \perp$. Set $R(u,u') = (B(u,u'), D(u,u'))$, again unless one side of the pair is $\perp$ in which case $R(u,u') = \perp$. By the construction of `Find-Children`, if one of L, R is set to $\perp$ the other one will as well. The special vertex in $V \times V$ will be (v,v').

Then $V^* \subset V \times V$ induces a binary tree (recall that V^* is the set of vertices reachable from (v,v') by L, R). Say that $\ell(v) \geq 2$; the other case follows by symmetry. Starting at (v,v'), L and R will yield a tree structure on the first member of the pair by the same argument as Lemma 29. Let p, p' be two distinct paths of any length which start at (v,v') and traverse with L and R. The vertex reached by p will have a different first pair element than the vertex reached by p' (by the tree structure on the first element of the pair), and thus p and p' arrive at distinct vertices. This means that there are no (undirected) cycles in the induced graph on V^*, and so it is a tree. ◀

5 Open Problems

We end with some future directions. The main problem left open by this work is to exhibit a natural problem in TFZPP^{dt} which is not reducible to Lossy-Code; we conjecture that Nephew is such a problem. Some additional open questions are the following:

- Find a TFZPP upper bound for Bertrand-Chebyshev, i.e., a natural problem in TFZPP to which Bertrand-Chebyshev reduces. The best upper bound we are aware of is only $\mathsf{LOSSY}^{\text{Factoring}}$ [69, 54], which is in $\mathsf{LOSSY}^{\mathsf{PPA}}$ and $\mathsf{LOSSY}^{\mathsf{PWPP}}$ under the generalized Riemann Hypothesis [49]. Unfortunately, none of these upper bound classes are in TFZPP.
- Find a TFZPP upper bound for the following problem, which we call Razborov-Smolensky [76, 79]: The input is an $\mathsf{AC}^0[2]$ circuit C of depth d and size at most $2^{n^{1/10d}}$ that attempts to compute MAJ (the Majority function), and the goal is to output an instance $x \in \{0,1\}^n$ such that $C(x) \neq \text{MAJ}(x)$. Since MAJ is average-case hard against such circuits, this problem sits in TFZPP. This problem is trivially solvable in deterministic quasi-polynomial time (note that the naïve algorithm runs in $2^{O(n)}$ time while the input size is $2^{n^{\Omega(1)}}$), hence we are interested in the regime where only polynomial-time reductions are allowed. We are not aware of any syntactic subclass of TFZPP that contains this problem.

See the full paper for open questions related to problems studied there which have been omitted in this shorter version.

References

1 Aryan Agarwala and Ian Mertz. Bipartite matching is in catalytic logspace. In *FOCS*, 2025. To appear. doi:10.48550/arXiv.2504.09991.

2 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. doi:10.4007/annals.2004.160.781.

3 Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008. doi:10.1016/J.JCSS.2007.06.025.

4 Albert Atserias and Neil Thapen. The ordering principle in a fragment of approximate counting. *ACM Trans. Comput. Log.*, 15(4):29:1–29:11, 2014. doi:10.1145/2629555.

5 Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998. doi:10.1006/JCSS.1998.1575.

6 Huck Bennett, Surendra Ghentiyala, and Noah Stephens-Davidowitz. The more the merrier! On total coding and lattice problems and the complexity of finding multicollisions. In *16th Innovations in Theoretical Computer Science Conference*, volume 325 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 14, 22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/lipics.itcs.2025.14.

7 Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Comput. Complex.*, 10(4):261–276, 2001. doi:10.1007/S000370100000.

8 Egon Börger, Erich Grädel, and Yuri Gurevich. *The classical decision problem*. Springer Science & Business Media, 2001.

9 Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. Computing with a full memory: catalytic space. In *STOC*, pages 857–866. ACM, 2014. doi:10.1145/2591796.2591874.

10 Joshua Buresh-Oppenheim. On the TFNP complexity of factoring. *Unpublished*, 2006.

11 Sam Buss, Noah Fleming, and Russell Impagliazzo. TFNP characterizations of proof systems and monotone circuits. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 30:1–30:40. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.30.

12 Samuel R Buss and Alan S Johnson. Propositional proofs and reductions between np search problems. *Annals of Pure and Applied Logic*, 163(9):1163–1182, 2012. doi:10.1016/J.APAL.2012.01.015.

13 Samuel R. Buss, Leszek Aleksander Kołodziejczyk, and Neil Thapen. Fragments of approximate counting. *J. Symb. Log.*, 79(2):496–525, 2014. doi:10.1017/JSL.2013.37.

14 Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *STOC'24—Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1990–1999. ACM, New York, [2024] ©2024. doi:10.1145/3618260.3649624.

15 Lijie Chen, Ce Jin, Rahul Santhanam, and Ryan Williams. Constructive separations and their consequences. *TheoretiCS*, 3, 2024. doi:10.46298/THEORETICS.24.3.

16 Lijie Chen, Jiatu Li, and Igor C. Oliveira. Reverse mathematics of complexity lower bounds. In *FOCS*, pages 505–527. IEEE, 2024. doi:10.1109/FOCS61266.2024.00040.

17 Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *FOCS*, pages 1–12. IEEE, 2020. doi:10.1109/FOCS46700.2020.00009.

18 Lijie Chen, Roei Tell, and Ryan Williams. Derandomization vs refutation: A unified framework for characterizing derandomization. In *FOCS*, pages 1008–1047. IEEE, 2023. doi:10.1109/FOCS57990.2023.00062.

19 Xi Chen and Xiaotie Deng. Settling the complexity of two-player Nash equilibrium. In *FOCS*, pages 261–272. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.69.

20 Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *STOC*, pages 1058–1066. ACM, 2023. doi:10.1145/3564246.3585147.

21 Mario Chiari and Jan Krajícek. Witnessing functions in bounded arithmetic and search problems. *J. Symb. Log.*, 63(3):1095–1115, 1998. doi:10.2307/2586729.

22 Jonas Conneryd, Susanna F. de Rezende, Jakob Nordström, Shuo Pang, and Kilian Risse. Graph colouring is hard on average for Polynomial Calculus and Nullstellensatz. In *FOCS*, pages 1–11. IEEE, 2023. doi:10.1109/FOCS57990.2023.00007.

23 James Cook, Jiatu Li, Ian Mertz, and Edward Pyne. The structure of catalytic space: Capturing randomness and time via compression. In *STOC*, pages 554–564. ACM, 2025. doi:10.1145/3717823.3718112.

24 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009. doi:10.1137/070699652.

25 Ben Davis and Robert Robere. Colourful TFNP and propositional proofs. In Amnon Ta-Shma, editor, *38th Computational Complexity Conference, CCC 2023, July 17-20, 2023, Warwick, UK*, volume 264 of *LIPIcs*, pages 36:1–36:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CCC.2023.36.

26 John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS = PPAD ∩ PLS. *J. ACM*, 70(1):7:1–7:74, 2023. doi:10.1145/3568163.

27 Noah Fleming, Stefan Grosser, Toniann Pitassi, and Robert Robere. Black-box PPP is not Turing-closed. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1405–1414. ACM, 2024. doi:10.1145/3618260.3649769.

28 Noah Fleming, Deniz Imrek, and Christophe Marciot. Provably total functions in the polynomial hierarchy. In Srikanth Srinivasan, editor, *40th Computational Complexity Conference, CCC 2025, August 5-8, 2025, Toronto, Canada*, volume 339 of *LIPIcs*, pages 28:1–28:40. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.CCC.2025.28.

29 Noah Fleming, Pravesh Kothari, and Toniann Pitassi. Semialgebraic proofs and efficient algorithm design. *Found. Trends Theor. Comput. Sci.*, 14(1-2):1–221, 2019. doi:10.1561/0400000086.

30 Lukáš Folwarczný, Mika Göös, Pavel Hubáček, Gilbert Maystre, and Weiqiang Yuan. One-way functions vs. TFNP: simpler and improved. In *15th Innovations in Theoretical Computer Science Conference*, volume 287 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 50, 14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/lipics.itcs.2024.50.

31 Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In *APPROX/RANDOM*, volume 275 of *LIPIcs*, pages 65:1–65:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.65.

32 Surendra Ghentiyala and Zeyong Li. Hierarchies within TFNP: building blocks and collapses. *CoRR*, 2025. doi:10.48550/arXiv.2507.21550.

33 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

34 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. *SIAM J. Comput.*, 53(3):573–587, 2024. doi:10.1137/22M1498346.

35 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 38:1–38:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.38.

36 Svyatoslav Gryaznov. Notes on resolution over linear equations. In *CSR*, volume 11532 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2019. doi:10.1007/978-3-030-19955-5_15.

37 Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. *ACM Trans. Comput. Theory*, 17(2):14:1–14:23, 2025. doi:10.1145/3718745.

38 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1150–1161, 2022. doi:10.1109/FOCS54457.2022.00111.

39 Jiří Hanika. *Search Problems and Bounded Arithmetic*. PhD thesis, Charles University, Prague, 2004.

40 Edward A. Hirsch and Ilya Volkovich. Upper and lower bounds for the linear ordering principle. *CoRR*, 2025. doi:10.48550/arXiv.2503.19188.

41 Max Hopkins and Ting-Chun Lin. Explicit lower bounds against $\Omega(n)$-rounds of sum-of-squares. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 662–673. IEEE, 2022. doi:10.1109/FOCS54457.2022.00069.

42 Pavel Hubácek, Erfan Khaniki, and Neil Thapen. TFNP intersections through the lens of feasible disjunction. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 63:1–63:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ITCS.2024.63.

43 Pavel Hubáček, Chethan Kamath, Karel Král, and Veronika Slívová. On average-case hardness in TFNP from one-way functions. In *Theory of cryptography. Part III*, volume 12552 of *Lecture Notes in Comput. Sci.*, pages 614–638. Springer, Cham, [2020] ©2020. doi:10.1007/978-3-030-64381-2_22.

44 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229. ACM, 1997. doi:10.1145/258533.258590.

45 Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Log.*, 129(1-3):1–37, 2004. doi:10.1016/j.apal.2003.12.003.

46 Emil Jeřábek. *Weak pigeonhole principle and randomized computation*. PhD thesis, Charles University in Prague, 2005.

47 Emil Jeřábek. Approximate counting in bounded arithmetic. *J. Symb. Log.*, 72(3):959–993, 2007. doi:10.2178/JSL/1191333850.

48 Emil Jeřábek. On independence of variants of the weak pigeonhole principle. *J. Log. Comput.*, 17(3):587–604, 2007. doi:10.1093/LOGCOM/EXM017.

49 Emil Jeřábek. Integer factoring and modular square roots. *J. Comput. Syst. Sci.*, 82(2):380–394, 2016. doi:10.1016/J.JCSS.2015.08.001.

50 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.

51 Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.44.

52 Leszek Aleksander Kolodziejczyk and Neil Thapen. Approximate counting and NP search problems. *J. Math. Log.*, 22(3):2250012:1–2250012:31, 2022. doi:10.1142/S021906132250012X.

53 Oliver Korten. The hardest explicit construction. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science—FOCS 2021*, pages 433–444. IEEE Computer Soc., Los Alamitos, CA, 2021. doi:10.1109/FOCS52979.2021.00051.

54 Oliver Korten. Derandomization from time-space tradeoffs. In *CCC*, volume 234 of *LIPIcs*, pages 37:1–37:26. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.37.

55 Oliver Korten. Range avoidance and the complexity of explicit constructions. *Bull. EATCS*, 145, 2025. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/825.

56 Oliver Korten and Toniann Pitassi. Strong vs. weak range avoidance and the linear ordering principle. In *FOCS*, pages 1388–1407. IEEE, 2024. doi:10.1109/FOCS61266.2024.00089.

57 Michal Koucký, Ian Mertz, Edward Pyne, and Sasha Sami. Collapsing catalytic classes. In *FOCS*, 2025. To appear. doi:10.48550/arXiv.2504.08444.

58 Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *J. Symb. Log.*, 69(1):265–286, 2004. doi:10.2178/jsl/1080938841.

59 Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22(3):253–275, 1985. doi:10.1007/BF00265682.

60 Jiatu Li, Edward Pyne, and Roei Tell. Distinguishing, predicting, and certifying: On the long reach of partial notions of pseudorandomness. In *FOCS*, pages 1–13. IEEE, 2024. doi:10.1109/FOCS61266.2024.00095.

61 Jiawei Li, Yuhao Li, and Hanlin Ren. Metamathematics of resolution lower bounds: A TFNP perspective. *CoRR*, abs/2411.15515, 2024. doi:10.48550/arXiv.2411.15515.

62 Yuhao Li, William Pires, and Robert Robere. Intersection classes in TFNP and proof complexity. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 74:1–74:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ITCS.2024.74.

63 Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *STOC*, pages 2000–2007. ACM, 2024. doi:10.1145/3618260.3649615.

64 Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987. doi:10.1007/3-540-48184-2_32.

65 Ian Mertz. Reusing space: Techniques and open problems. *Bull. EATCS*, 141, 2023. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/780.

66 Moritz Müller. Typical forcings, NP search problems and an extension of a theorem of riis. *Ann. Pure Appl. Log.*, 172(4):102930, 2021. doi:10.1016/J.APAL.2020.102930.

67 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

68 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

69 Jeff B. Paris, A. J. Wilkie, and Alan R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symb. Log.*, 53(4):1235–1244, 1988. doi:10.1017/S0022481200028061.

70 Amol Pasarkar, Christos H. Papadimitriou, and Mihalis Yannakakis. Extremal combinatorics, iterated pigeonhole arguments and generalizations of PPP. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 88:1–88:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.88.

71 Aaron Potechin. Sum of squares bounds for the ordering principle. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 38:1–38:37. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.38.

72 Pavel Pudlák. On the complexity of finding falsifying assignments for Herbrand disjunctions. *Arch. Math. Log.*, 54(7-8):769–783, 2015. doi:10.1007/S00153-015-0439-6.

73 Pavel Pudlák and Neil Thapen. Random resolution refutations. *Comput. Complex.*, 28(2):185–239, 2019. doi:10.1007/S00037-019-00182-7.

74 Edward Pyne. Derandomizing logspace with a small shared hard drive. In *CCC*, volume 300 of *LIPIcs*, pages 4:1–4:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.CCC.2024.4.

75 Edward Pyne, Ran Raz, and Wei Zhan. Certified hardness vs. randomness for log-space. In *FOCS*, pages 989–1007. IEEE, 2023. doi:10.1109/FOCS57990.2023.00061.

76 Alexander 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*, 41(4):333–338, 1987.

77 Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *FOCS*, pages 640–650. IEEE, 2022. doi:10.1109/FOCS54457.2022.00067.

78 Søren Riis. A complexity gap for tree resolution. *Comput. Complex.*, 10(3):179–209, 2001. doi:10.1007/S00037-001-8194-Y.

79 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

80 Neil Thapen. *The weak pigeonhole principle in models of bounded arithmetic*. PhD thesis, University of Oxford, 2002.

81 Neil Thapen. How to fit large complexity classes into TFNP. *CoRR*, abs/2412.09984, 2024. doi:10.48550/arXiv.2412.09984.

82 Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003. doi:10.1016/S0022-0000(03)00046-1.