TFNP in the Random Oracle Model

Noah Fleming[1] Mika Göös[2] Ziyi Guan[3] Oliver Korten[4] Daniel Mitropolsky[5]
Toniann Pitassi[6] Artur Riazanov[7] Dmitry Sokolov[8] Tianqi Yang[9] Weiqiang Yuan[10]

Abstract

We determine which major TFNP classes are hard in the random oracle model (ROM). Prior work establishes average-case hardness of highly structured subclasses of TFNP from strong cryptographic assumptions such as LWE and indistinguishability obfuscation. It is not known whether weaker assumptions such as one-way functions imply hardness of any of the standard TFNP subclasses, even in the worst case. The ROM, where all parties have query access to a random function, lies, morally, between these two regimes: a random function is one-way, but public-key cryptography cannot be built from it in a black-box way. We refer to this setting as *Romantica*.

Our first result settles a long-standing open problem: PPAD, the class of total search problems reducible to finding a Nash equilibrium, is hard on average in Romantica. Combined with the known hardness of PLS (local search), this gives average-case hardness of $\mathsf{EoPL} = \mathsf{PPAD} \cap \mathsf{PLS}$. In contrast, we show that UEoPL can be solved in Romantica with polynomially many oracle queries by a computationally unbounded solver, a sharp distinction between EoPL and UEoPL. This resolves the black-box hardness landscape of all major TFNP classes in Romantica.

Finally, we initiate the study of separations between TFNP subclasses in Romantica: PWPP and PPA do not reduce to PLS under *public-coin* reductions.

Table of Contents

1 Introduction **2**
2 Techniques **6**
3 Average-case separation between PPAD and FP **11**
4 Efficient search for UEoPL in the black-box ROM **20**
5 PLS separations from proof complexity **28**
Acknowledgements **35**
AI disclosure **36**
References **36**
A TFNP glossary **41**
B TFNP separations in the ROM **42**

[1]Lund University. noah.fleming@cs.lth.se
[2]EPFL. mika.goos@epfl.ch
[3]MIT. ziyiguan@mit.edu
[4]IAS. korten@ias.edu
[5]Tufts University, MIT. mitropol@mit.edu
[6]Columbia University. tp2684@columbia.edu
[7]Columbia University. tunyash@gmail.com
[8]Université de Montréal. dmitry.sokolov@umontreal.ca
[9]Columbia University. tianqi@cs.columbia.edu
[10]University of Copenhagen. weiqiang.yuan@di.ku.dk

1 Introduction

TFNP is the class of *total search problems* in NP, those for which solutions always exist and can be efficiently verified. TFNP is studied through its subclasses, each capturing a *principle of totality*. The smallest, FP, consists of the total problems solvable in polynomial time; every other subclass is defined by reduction to a canonical complete problem. Four well-studied classes are PPP, PWPP, PLS and PPAD. In PPP, for the *pigeonhole principle* [Pap94], the complete problem is to find a collision $C(x) = C(x')$ of a circuit $C\colon [2^n] \to [2^n - 1]$. PWPP, for the *weak* pigeonhole principle [Jeř16], has the same complete problem but now, $C\colon [2^n] \to [2^{n-1}]$ has twice as many inputs as outputs. The third, PLS, captures *local search* [JPY88]: given a monotonically *increasing* function $f\colon [2^n] \to [2^n]$, find a point at which it does not increase. Finally, PPAD captures the hardness of finding a Nash equilibrium in bimatrix games [DGP09, CDT09], although the original complete problem rests on a *parity argument on directed graphs* [Pap94]. Formal definitions are in Section A.

TFNP hardness and cryptography. The theory of total search problems studies under what conditions TFNP subclasses are *hard*, or *hard on average*, as evidence for the computational intractability of total problems such as finding a Nash equilibrium. It is unlikely that NP-hardness alone will suffice: if TFNP contains an NP-hard problem, then NP = co-NP [MP91]. Over the past decade, researchers have studied *cryptographic* assumptions as a source of hardness for TFNP. Conversely, this maps to studying which total problems are hard in each of Impagliazzo's "worlds" [Imp95]. All cryptographic assumptions trivially imply the hardness of NP, and are believed to in fact imply more, e.g. hardness of some NP-*intermediate* classes. TFNP is a candidate for such a class.

The main line of work in this field has centered on proving UEoPL average-case hardness from *strong* cryptographic assumptions, i.e. in the higher worlds. UEoPL [FGMS20] is the lowest of the standard TFNP subclasses: it is contained in every class mentioned so far. Average-case hardness of UEoPL was first shown from indistinguishability obfuscation [BPR15] (see also [BDY26]), that is, in Obfustopia, and later from public-key assumptions [GPS16, LV20, KPY20, JKKZ21, BCH+22, KLV23, CFJW25], that is, in Cryptomania.

In Minicrypt, the world where *one-way functions* (OWFs) exist but no public-key assumptions hold, TFNP itself is hard on average, either non-uniformly or under an additional derandomization assumption [HNY17]. However, no problem in any of its standard subclasses is known to be hard in Minicrypt, not even in the worst case, whereas the stronger assumptions above give average-case hardness of UEoPL, the most structured class of all. Barriers against restricted black-box constructions suggest that OWFs may not suffice [RSS21, HKKS20, FGH+24].

The random oracle model. Is public-key cryptography necessary? A natural starting point for this question is the random oracle model (ROM) [BR93], a setting often studied in cryptography when OWFs appear to be insufficient. Briefly, in the ROM, all parties (instance samplers, solvers, verifiers, etc.) have query-access to a randomly sampled function (see Section 2.1 for a complete definition). A random function is itself one-way, so morally, the ROM "contains" Minicrypt. On the other hand, public-key cryptography cannot be built in the ROM in a black-box way [IR89] (and hence not from OWFs either). The world with a random oracle, which we call **Romantica** ♡, therefore sits roughly between Minicrypt and Cryptomania.[1] A natural question is whether TFNP or its subclasses are hard in Romantica. Hardness in Romantica means an efficient sampler with oracle access whose output instances, which may contain oracle gates, no poly-time solver with the same oracle can solve. Three TFNP-subclasses were previously known to be hard on average in Romantica: PPP and PWPP (whose hard distributions are simple), as well as PLS, which requires a non-trivial construction based on proofs of sequential work [BG20]. If moreover #SAT is hard, then UEoPL, and hence every class above it, is hard on average in Romantica [CHK+19a, BCH+22]. Whether any other subclass, in particular PPAD, UEoPL, PPA or EoPL, is hard in Romantica without additional assumptions has been open until now.

[1] Neither world contains the other. A random oracle is a one-way function, so everything built from one-way functions in a black-box way exists in Romantica, but not everything Minicrypt may contain: for instance, no one-way permutation can be built from a random oracle in a black-box way [Rud88, KSS10]. Conversely, Romantica contains unconditional primitives that are not known to follow from public-key cryptography, such as succinct arguments [Mic00] and proofs of sequential work [CP18].

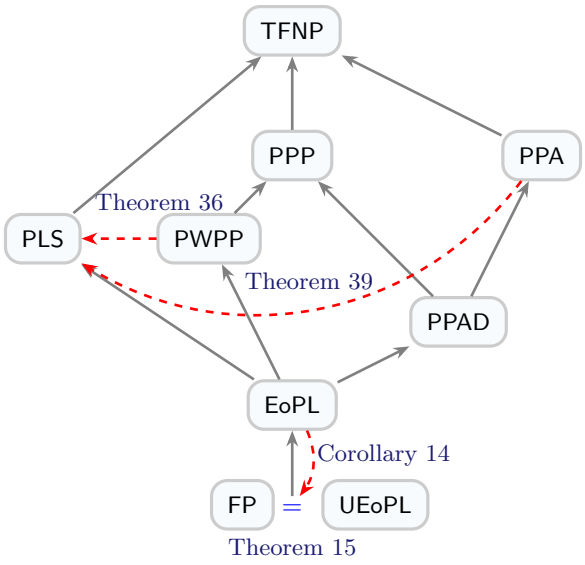

Figure 1: Separations, inclusions, and equivalences in the ROM (*Romantica*), red arrows are average-case *separations* and are the results of this paper. Average-case hardness is equivalent to average-case separation from FP.

1.1 Our results

We resolve, for every standard TFNP subclass, whether it is hard on average in Romantica; Figure 1 summarizes the results. Overall we show: PPAD is hard on average, and hence so is EoPL = PPAD ∩ PLS, whereas UEoPL collapses to FP in the *black-box* sense: a solver making polynomially many oracle-queries exists. Finally, we initiate the study of separations between TFNP subclasses in Romantica: neither PWPP nor PPA reduces to PLS under public-coin reductions. We now state these results more fully in turn.

Theorem (Theorem 1, Informal)**.** *PPAD is hard on average in the random oracle model. The hard distribution is efficiently samplable, and any solver making polynomially many oracle queries finds a solution with negligible probability.*

The proof modifies the Yamakawa–Zhandry hypergraph [YZ24], originally used to separate quantum from classical query algorithms, so that the search problem becomes total while remaining hard; see Section 2.2.

Theorem 1 and the known hardness of PLS [BG20] imply, via a simple or-composition, average-case hardness of PLS ∩ PPAD. This intersection equals EoPL [FGHS23, GHJ+22], the class of problems reducible to finding the end of a potential line [FGMS20].

Corollary (Corollary 14, Informal)**.** *PPAD ∩ PLS = EoPL is hard on average in the random oracle model.*

Since PPA ⊇ PPAD, every class in Figure 1 except UEoPL is now known to be hard on average in Romantica. The remaining question is whether UEoPL, the lowest widely studied non-trivial TFNP subclass (and in particular a subclass of EoPL = PPAD ∩ PLS), is also hard on average in Romantica. We answer this essentially in the *negative*, highlighting a key structural difference between UEoPL and EoPL with respect to random oracles:

Theorem (Theorem 15, Informal)**.** *In the random oracle model, given any poly-time samplable distribution over instances of UEoPL, there exists an algorithm that finds solutions with probability ≥ 2/3 and uses at most polynomially many queries to the random oracle.*

Theorem 15 concerns query efficiency, not time efficiency: the solver makes polynomially many oracle queries, but its computation need not be efficient. Indeed, as mentioned previously, UEoPL is known to be hard on average in Romantica with the additional assumption that #SAT is hard on average [CHK+19a]. This suggests that in Romantica, a computational assumption is necessary (and sufficient) to establish UEoPL-hardness. The proof rests on a recent dichotomy lemma [BCG26]; see Section 2.3.

To our knowledge, all previous average-case hardness results for PPAD and EoPL establish hardness of UEoPL and inherit it by containment. In Romantica, Theorem 15 rules out this route, so Theorem 1 is the first PPAD hardness result that does not go through UEoPL.

Finally, we also initiate the study of separations between different TFNP subclasses in Romantica: PWPP and PPA, the class for *parity arguments* on undirected graphs [Pap94], *cannot* be reduced to PLS under *public reductions*, whose only randomness is the random oracle itself:

Theorem (Theorem 36, Informal)**.** *In the random oracle model, there is no efficient public reduction from a PWPP-complete problem to a PLS-complete problem.*

Theorem (Theorem 39, Informal)**.** *In the random oracle model, there is no efficient public reduction from a PPA-complete problem to a PLS-complete problem.*

Our proofs reduce these separations to proof-complexity lower bounds; see Section 2.4. This step relies on the reductions being public. The oracle is visible to the solver of the target problem, so a public reduction hides nothing from the solver, who may choose which solution to return with full knowledge of the reduction. A *private-coin* reduction may in addition toss coins hidden from the solver, which may help the reduction. Separating TFNP subclasses in Romantica under private-coin reductions remains open, as do separations between other pairs of subclasses.

A proof-complexity perspective. Black-box TFNP subclasses correspond exactly to proof systems [BIK+96, BFI23, GHJ+22], and passing to the random oracle model, a reduction may now fail on a small fraction of oracles. From the proof side, this corresponds to proofs that are *approximately sound*, meaning that they are correct on all but a small fraction of *uniformly random* inputs. In particular, a ROM reduction to PLS is a narrow random resolution refutation with respect to a locally samplable distribution (see Section 5) so Theorems 36 and 39 are lower bounds for approximately sound proofs. We note that for stronger proof systems such as bounded-depth Frege systems, a recurring issue is how to define approximate soundness in a meaningful way. Note that for any Frege proof (even of low depth) of a tautology, every line in the proof is semantically equivalent to "1" so any useful notion of an approximate proof must incorporate the local nature of the rules in order to avoid this pitfall.

More generally lower bounds for approximate proofs have many applications outside of cryptography: (i) they underlie lower bounds on the extension complexity of LP and SDP relaxations for approximating NP-hard problems; (ii) they are the object of study in random resolution, with applications to theories of approximate counting; (iii) they are exactly the missing ingredient in the Razborov–Smolensky approach to $\mathrm{AC}^0[p]$-Frege lower bounds that motivated the introduction of Nullstellensatz and Polynomial Calculus; and (iv) they are the mirror image of the algorithmic problem of refuting random 3SAT, which reduces to certifying that a random 3XOR instance is highly unsatisfiable. We discuss the first three connections in Section 1.2.2.

1.2 Related work

1.2.1 Average-case TFNP and cryptography

Unconditional hardness of TFNP subclasses in Romantica. It is not hard to see that the existence of collision-resistant hash functions (CRHFs) implies the average-case hardness of the weak pigeonhole principle. Hence, the average-case hardness of PPP and PWPP in the ROM follows from the fact that CRHFs exist in the ROM. Apart from this, the only TFNP-subclass that is known to be hard (and in fact, average-case hard) prior to our work is PLS [BG20]. Their result is obtained by combining the construction of proofs of work in the ROM [CP18] and ideas from the Fiat-Shamir heuristics.

On the pursuit of PPAD hardness. Showing hardness of PPAD in the ROM has been a long-standing open problem. Prior to our work, the best known results were conditional hardness for UEoPL (and hence PPAD) which, in addition to the random oracle, involved a complexity-theoretic assumption like #SAT being hard on average [CHK+19a, Theorem 2], or hardness of EoPL from the iterated squaring assumption

[CHK+19b]. As mentioned previously, this does not contradict our new UEoPL upper-bound since Theorem 15 only proves a *query*-complexity upper bound (i.e., a black-box upper-bound); the algorithm we give is not computationally efficient in the white-box sense.

Theorem 1 is proved in a very different way compared to previous conditional hardness results. [CHK+19a, Theorem 2] is proved by applying Fiat-Shamir to the sum-check protocol for #SAT. It is worth noting that several works have studied the instantiation of the Fiat-Shamir heuristics using concrete cryptographic hash functions, obtaining conditional hardness of PPAD in the plain model (see, e.g., [LV20, JKKZ21, BCH+22, KLV23, CFJW25]). It remains an open problem whether one can replace the random oracle model in Theorem 1 with well-studied cryptographic assumptions; black-box hardness does not translate into white-box hardness in general [KNY19].

1.2.2 Average-case TFNP and proof complexity

Proof complexity and black-box TFNP classes are intimately connected: each $\mathsf{TFNP}^{\mathrm{dt}}$ subclass is characterized by a proof system [BFI23], with PLS corresponding to resolution width [BKT14, Kam19], PPA to Nullstellensatz over $\mathbb{F}_2$ [GKRS19], and PPAD to unary Nullstellensatz [GHJ+24].

When extending these equivalences to the ROM, the relevant new feature is error, which we model by a low-width "advice" DNF B which is satisfied with probability at most $1/3$ over the instance distribution $\omega(u)$ induced by the uniformly random oracle u. In particular, a ROM reduction from some problem F to PLS becomes a narrow resolution refutation of $F \vee B$ for an appropriate choice of B satisfying the aforementioned properties. This is closely related to "random resolution width" of [BKT14], and more generally to the notion of randomized reductions in TFNP [FGJ+26]. The key distinction is that in our case, the distribution $\omega(u)$ must be samplable by a low depth decision tree from a random oracle (i.e. a string of random independent bits); this is a significant restriction and rules out many candidate "hard distributions" one may attempt to use to prove a lower bound. Our separations are thus lower bounds for approximately sound proofs. We briefly survey three settings in which such proofs arise.

Extension complexity. Lower bounds on the size of LP and SDP relaxations for approximating CSPs [CLRS16, LRS15, KMR17] convert, via query-to-communication lifting, a small relaxation into a low-degree Sherali–Adams or Sum-of-Squares certificate. The extracted certificate is correct only up to a small error, measured on average over assignments, which is absorbed into the approximation guarantee. At the core of such a result is therefore a degree lower bound for approximate low-degree refutations of a particular gap statement.

Random resolution. Random resolution, defined in [BKT14] following a suggestion of Dantchev, formalizes resolution refutations that may err on a small fraction of assignments; it arose as the propositional counterpart of Jeřábek's theories of approximate counting. The first lower bounds for random resolution were obtained for the colored polynomial local search principle CPLS, and for the pigeonhole principle [PT19, KT22].

The relation to the random oracle model is as follows. A ROM reduction from a totality principle F to PLS produces a narrow resolution refutation of F together with advice that is correct with high probability over the instance distribution $\omega(u)$, where u is the uniformly random oracle and ω is a forest of shallow decision trees. That is, a ROM reduction to PLS is a narrow random resolution refutation with respect to a *locally samplable* distribution, and a ROM lower bound against PLS is a random resolution lower bound over that particular distribution. The converse does not always hold. For example, the previously established lower bounds for random resolution [PT19, KT22] yield distributions that are hard for random resolution, but these distributions are not known to be locally samplable from uniform bits, so they do not imply ROM separations. Separations between TFNP subclasses under deterministic black-box reductions are well understood [BCE+98, Mor01, BOM04, HY20, GKRS19, BT22, GHJ+24]. Similarly the randomized separations between pairs of TFNP subclasses in [Li24, JLRX24, Kis26], and the oracle separations of the multi-collision classes in [KNY18], do not apply to the ROM. In contrast, Bitansky and Gerichter [BG20] obtained a ROM separation of PLS from FP; that is, they constructed a distribution over PLS instances that is efficiently samplable from a uniformly random oracle, but that is hard for FP. Similarly we obtain the following ROM separations: (i) PWPP vs. PLS (Theorem 36), (ii) PPA vs. PLS (Theorem 39) and (iii) EoPL vs. FP (Corollary 14).

$\mathrm{AC}^0[p]$-Frege and the Razborov–Smolensky method. Nullstellensatz [BIK+96] and Polynomial Calculus [CEI96] were introduced in large part to obtain an analogue, for $\mathrm{AC}^0[p]$-Frege, of the Razborov–Smolensky method [Raz87, Smo87], which replaces each gate of a circuit by a low-degree polynomial over $\mathbb{F}_p$ agreeing with it on most inputs. Applied to a proof, the method replaces each line by a low-degree polynomial correct on most assignments, yielding an approximately sound low-degree algebraic refutation. [BIK+97] made this precise: $\mathrm{AC}^0[p]$-Frege lower bounds follow from degree lower bounds for Nullstellensatz over $\mathbb{F}_p$ with extension axioms formalizing the approximation step, axioms that are sound only with high probability over a random choice of auxiliary variables. Strong degree lower bounds for the exact systems followed [BIK+96, Raz98, IPS99, BGIP01], and these underlie the Nullstellensatz characterizations of PPA and PPAD used in this paper; the robust half of the program is what remains open.

2 Techniques

We overview the techniques behind our results. Section 2.1 recalls the decision-tree formulation of the random oracle model used throughout. Section 2.2 sketches the proof of PPAD hardness (Theorem 1), Section 2.3 the UEoPL algorithm (Theorem 15), and Section 2.4 the separations from PLS (Theorems 36 and 39).

2.1 TFNP in the ROM briefing

In the cryptographic formulation, a TFNP problem in the ROM is specified by a *verifier*, a polynomial-time algorithm with oracle access that, given an instance and a candidate solution, decides whether the candidate is a solution; totality means that every instance has at least one solution. An instance is a circuit C with oracle gates mapping $\{0,1\}^{\mathrm{poly}(n)}$ to Σ, with $|\Sigma| \le 2^{\mathrm{poly}(n)}$. The problem is hard on average if some efficient *sampler* with oracle access outputs instances that no polynomial-time *solver* with oracle access solves with probability $2/3$ over the oracle. We view the oracle as a random string $u \in \{0,1\}^{\mathbb{N}}$. Let the truth table of C be $x \in \Sigma^N$, $N = 2^{\mathrm{poly}(n)}$. Then for a fixed oracle u, each entry of x is one evaluation of C, which reads $\mathrm{poly}(n) = \mathrm{polylog}(N)$ bits of u, and checking a candidate solution means evaluating C at a few points. Please see Appendix B for a formal definition of the cryptographic formulation of the ROM model.

Note that only these query counts matter for our lower bound proofs, which leads to the following *decision-tree formulation*, used throughout the paper. A $\mathit{TFNP}^{\mathrm{dt}}$ *problem* is a relation $S \subseteq \Sigma^N \times \mathcal{O}$, with $|\mathcal{O}| \le 2^{\mathrm{polylog}(N)}$, that is total and such that for every $o \in \mathcal{O}$ the predicate $(x, o) \in S$ is decided by a decision tree of depth $\mathrm{polylog}(N)$ over x. An *instance distribution* is a forest $\omega = (\omega_1, \dots, \omega_N)$ of decision trees $\omega_i \colon \{0,1\}^{\mathbb{N}} \to \Sigma$ of depth $\mathrm{polylog}(N)$, mapping the oracle u to the instance $\omega(u) \in \Sigma^N$. A *solver* is a decision tree $A \colon \{0,1\}^{\mathbb{N}} \to \mathcal{O}$.

We say that $S \in \mathsf{FP}$ *in the black-box ROM* if for every decision forest ω there is a solver A of depth $\mathrm{polylog}(N)$ with

$$\Pr_{\boldsymbol{u} \sim \{0,1\}^{\mathbb{N}}}[(\omega(\boldsymbol{u}), A(\boldsymbol{u})) \in S] \ge 2/3,$$

and that S is *hard on average in Romantica* otherwise. A polynomial-time solver with oracle access makes polynomially many queries, so it is a solver of depth $\mathrm{polylog}(N)$, and black-box hardness implies hardness in the cryptographic formulation.

Since ω is itself a shallow forest over u, it can be absorbed into the problem: $S' = \{(u, o) \mid (\omega(u), o) \in S\}$ is again a $\mathsf{TFNP}^{\mathrm{dt}}$ problem, now over the oracle string. Throughout the paper the instance is therefore the oracle itself, and for convenience we take $\boldsymbol{u} \sim \Sigma^{\mathbb{N}}$.[2]

2.2 PPAD lower bound

Consider the following search problem template: given a large hypergraph with a labeling of the vertices, find an edge whose vertex labels satisfy a constraint. This yields a search problem for every choice of hypergraph and constraint. For several simple choices of the two, it is easy to show that the problem is *total*, but then it is also easy to solve. Our proof takes the *Yamakawa–Zhandry hypergraph*, originally used to separate quantum

[2] It is not hard to see that this is without loss of generality: for every Σ the uniform distribution Σ^N can be sampled with $O(\log(N|\Sigma|))$ queries to $\boldsymbol{u}$ per coordinate within total variation distance $(N|\Sigma|)^{-\Omega(1)}$, see e.g. [HLS+26].

and classical algorithms [YZ24]. The *constraint* part of their search problem does not guarantee totality, but the search problem is hard for classical algorithms. The key idea is to tweak their construction, which sits at the hard-but-not-total end of this spectrum, toward the total-but-easy end, stopping at a problem that is total (in fact, in PPAD) while still hard.

Which class the resulting problem lands in is determined by how its totality is proved. We first show hardness-on-average in Romantica of the class PPA, for *parity arguments*, which contains the subclass PPAD. We then refine the construction to obtain a hard-on-average problem in PPAD. Informally, problems in PPA are exactly the problems whose totality can be proven by an efficient parity argument, and the problems in PPAD are the ones whose totality can be proven by an efficient signed counting argument. In our construction, these arguments simply sum local constraints modulo 2 or over $\mathbb{Z}$ with signs ± 1, respectively. These connections have been made formal in [GKRS19] and [GHJ+24] respectively. The corresponding polynomial refutations must have suitably bounded degree and size, with coefficients measured in unary over $\mathbb{Z}$; the formal proof verifies these bounds and provides efficient pairings witnessing the cancellations. In this exposition, we suppress these efficiency requirements, making the task:

Find a problem that is hard on average given random access to a uniformly random string, and whose totality can be proven by counting in $\mathbb{F}_2$ (for PPA) or $\mathbb{Z}$ (for PPAD).

PPA worst-case hardness. A canonical PPA problem is finding a *falsified equation* in an unsatisfiable system of linear equations over $\mathbb{F}_2$. Let $N = 2^n$, and let $x_1, \ldots, x_N$ be the variables of the system. We identify each equation $\sum_{v \in e} x_v = 1$ (over $\mathbb{F}_2$) with a *hyperedge* $e \subseteq [N]$, and an *evaluation* $x \in \{0,1\}^N$ of all N variables with a binary labeling of the nodes of a hypergraph $G = ([N], E)$. Given any hypergraph G with edges of size $|e| \leq \mathrm{poly}(n)$ and an efficiently-accessible labeling of its nodes, we obtain the following NP-relation:

$$\textsc{Lin}_G \coloneqq \left\{ (x, e) \,\middle|\, \sum\nolimits_{v \in e} x_v = 0 \right\} \tag{$\star$}$$

Of course, this NP-relation is only total if we can guarantee the unsatisfiability of the linear system. One way is to enforce that each node $v \in [N]$ is contained in an *even* number (at most $\mathrm{poly}(n)$) of hyperedges, and that the number of hyperedges $|E|$ is *odd*. Then, one can show that the problem lies in PPA.

However, these conditions do not make the problem $\textsc{Lin}_G$ hard, even in the worst-case (e.g. G can be an odd cycle). Previous work has shown that *worst*-case hardness holds if G has strong *expansion* properties [Tse70, Urq87]. However, it is not clear how to extend this to average-case hardness; for instance, if the labeling $\boldsymbol{x} \sim \{0,1\}^N$ is sampled uniformly (a distribution that can be easily realized in the ROM), then finding a solution of ($\star$) is easy for *every* G: any edge is going to be a solution with probability $1/2$.

Yamakawa–Zhandry problem. Our second key ingredient is an NP-search problem defined by Yamakawa and Zhandry [YZ24] in order to separate quantum and classical algorithms for search problems in the ROM. Let $G = ([N], E)$ be a fixed hypergraph with $E \subseteq \binom{[N]}{\ell}$, where $\ell = \mathrm{poly}(n)$. The generic version of the problem $\textsc{YamakawaZhandry}_G$ is: given a labeling $x \in \{0,1\}^N$ find a hyperedge $e \in E$ such that $x_v = 0$ for all $v \in e$. Note that this is unlike the condition that makes an edge a solution in the previous section, and makes this problem very far from total. However, Yamakawa and Zhandry construct a specific hypergraph for which it is hard to find a solution given *uniformly random* $\boldsymbol{x} \sim \{0,1\}^N$. In other words, in the previous section we saw constructions of a hypergraph and edge-condition which yielded a total problem, but solutions were easy to find for random labelings. On the other end of the spectrum, Yamakawa-Zhandry's hypergraph and edge-condition yields a hard search problem over uniform labelings, but not all instances will have solutions.

The key property of their construction is *well-spreadness*: $G = ([N], E)$ is (r, a, b)-well-spread if for every set $S \in \binom{[N]}{a}$ the number of hyperedges $e \in E$ such that $|e \cap S| \geq r$ is at most b. Why does well-spreadness make $\textsc{YamakawaZhandry}_G$ hard? Suppose a decision tree queried at most a labels in $\boldsymbol{x}$. Then there are at most b hyperedges for which the tree queries more than r labels. For each of those hyperedges, consider the moment when exactly r labels are queried. Then the conditional probability that the remaining queries make the edge all-0 is $2^{r-\ell}$. So if $b \cdot 2^{r-\ell} \ll 1$, we are unlikely to discover any solution.

PPA-hard problem. One obvious issue with $\textsc{YamakawaZhandry}_G$ for us is that it is not total, e.g. the labeling $x = 1^N$ has no solutions. Hence we will need to modify the *condition* for an edge to be a solution. For the problem to additionally be in PPA, a *parity argument* must guarantee that at least one edge satisfies our chosen condition. For a given graph G, let $p_e \colon \{0,1\}^N \to \{0,1\}$ be the condition *verifier* of e, i.e. $p_e(x) = 1$ iff e is a solution for x. One way to guarantee totality is to choose p_e such that

$$\sum_{e \in E} p_e(x) \equiv 1 \bmod 2.$$

As witnessed by $\textsc{Lin}_G$, a choice of condition $p_e = 1 + \sum_{v \in e} x_v$ over $\mathbb{F}_2$ satisfies the equation above (i.e. yields a total problem) whenever every node is contained in an even number of hyperedges and the total number of edges is odd. However, as we also discussed, $\Pr[p_e(\boldsymbol{x}) = 1] = 1/2$, so even if G is well-spread, finding the solution will be easy.

There is a natural fix for that: suppose that instead of binary labels, we used labels $x_1, \ldots, x_N \in \{0,1\}^r$, and we set $p_e(x) = [\![\oplus_{v \in e} x_v = 0^r]\!]$. Then over a uniformly-random labeling, $\Pr[p_e(\boldsymbol{x}) = 1] = 2^{-r}$, so for every well-spread G finding e with $p_e(\boldsymbol{x}) = 1$ is hard.

In order to see what is needed from G to make the problem total, it is convenient to plant a forbidden solution. Take $e^* \in E$ and fix all its labels to 0^r, then forbid e^* as a solution. That makes the condition on the hypergraph G we need more convenient: $\sum_{e \in E} p_e(x) \equiv 0 \bmod 2$. Since $p_{e^*}(x) = 1$, this gives $\sum_{e \neq e^*} p_e(x) \equiv 1 \bmod 2$, guaranteeing a non-forbidden solution. To see what we need from G, write p_e as a polynomial in the *bits* of the labels; the target expression becomes

$$\sum_{e \in E} \prod_{i \in [r]} (1 + \sum_{v \in e} x_{v,i}) \equiv 0 \bmod 2.$$

Expanding the brackets we get that the coefficient of monomial $x_{v_1,i_1} \cdot \cdots \cdot x_{v_k,i_k}$, with distinct bit coordinates $i_1, \ldots, i_k$, is the number of edges containing $\{v_1, \ldots, v_k\}$ modulo 2. That gives us a sufficient condition on the hypergraph: we say that $G = ([N], E)$ is r-even if for every $S \in \binom{[N]}{\leq r}$ the number of edges $e \supseteq S$ is even. $S = \emptyset$ ensures cancellation of the constant term. In Section 3.3, we show that a variant of the Yamakawa–Zhandry hypergraph is r-even for $r = \Theta(\ell)$.

PPAD-hard problem. To obtain a problem in PPAD, we replace the parity argument above with an integer cancellation argument. The input to the new hyperedge-search problem is a labeling $x \in ([r] \cup \{\bot\})^N$, where $\bot$ is a void label that we do not count. We fix all labels on the distinguished edge e^* to $\bot$ and again forbid e^* as a solution. The polynomial formulation of this problem has essentially the same structure:

$$p_e \coloneqq \prod_{i \in [r]} (1 - \sum_{v \in e} [\![x_v = i]\!]).$$

Thus, $p_e(x) \neq 0$ iff every non-$\bot$ label occurring on e occurs at least twice. The values of p_e are now integers rather than solution indicators. We choose signs $\varepsilon_e \in \{\pm 1\}$ so that $\sum_{e \in E} \varepsilon_e p_e \equiv 0$. In the expansion, the coefficient of a degree-k indicator monomial on distinct vertices $S = \{v_1, \ldots, v_k\}$ and distinct labels is $(-1)^k \sum_{e \supseteq S} \varepsilon_e$, for $0 \leq k \leq r$; terms assigning different labels to the same vertex vanish. Hence, it suffices to require that for every $S \in \binom{[N]}{\leq r}$, including the empty set, the signs of the edges containing S sum to zero. We show in Section 3.3 that the Yamakawa-Zhandry hypergraph admits such signing.

Since all labels on e^* are $\bot$, we have $p_{e^*}(x) = 1$. Normalizing the signing so that $\varepsilon_{e^*} = 1$ and removing this planted contribution gives

$$-\sum_{e \in E \setminus \{e^*\}} \varepsilon_e p_e(x) = 1.$$

At least one non-forbidden edge therefore has $p_e(x) \neq 0$, proving totality via *counting over integers*. We make this formal in Section 3.2.

2.3 UEoPL upper bound

The problem setup. It is convenient to use the grid form of UEoPL from [GHJ+24]. The nodes are the cells (i, j) of an $N \times N$ grid, i the row. Every node has a successor pointer $\textsc{Succ}_{(i,j)} \in [N] \cup \{\perp\}$, read as naming the node $(i+1, \textsc{Succ}_{(i,j)})$ of the next row, and every node below row 1 has a predecessor pointer. A node is *active* if its successor's predecessor points back to it: $\textsc{Succ}_{(i,j)} = k$ and $\textsc{Pred}_{(i+1,k)} = j$ (in the last row, active just means $\textsc{Succ} \neq \perp$). If $(1, 1)$ is active, following successors from it traces a *line* that descends one row per step, so it must end at an inactive node that an active node points to (a *proper sink*) or at an active node of row N. A solution is any of: $(1, 1)$ inactive; a proper sink; an active node of row N; an active node other than $(1, 1)$ with no active predecessor; or *two active nodes in the same row*. The last kind is what makes this UEoPL rather than EoPL: proof of a second line is also a solution. Activity is the only notion our algorithm uses, and testing it costs at most two pointer reads.

In the random oracle model an instance is generated from the oracle $u \in \{0, 1\}^{\mathbb{N}}$ by decision trees: each pointer is a tree of depth q over the bits of u. We show (Theorem 15) that there exists a decision tree of depth $O(q^5 \log N)$, querying the bits of u, that outputs a solution with probability $2/3$ over $\boldsymbol{u}$, and never outputs a wrong answer. This is a *black-box* statement: it counts queries, not computation.

Jumping and a win-win argument. The obvious strategy is to begin at the active cell $(1, 1)$ in row 1, and move down one row at a time until we either reach a proper sink or an active node in the last row. The query cost is $2q$ queries per row, hence $\Theta(qN)$ queries. To get to $\text{poly}(q) \log N$ queries, we jump: a jump of 2^r rows is two jumps of 2^{r-1} rows. Composed naively this doubles the cost at each of the $\log N$ levels, which is no better. The proof therefore rests on a *dichotomy lemma* (Lemma 16, originally proved in [BCG26]): an algorithm that finds solutions, whatever its query cost, can be collapsed either to a query-efficient one or to one that outputs *two* solutions. In our setting it says the following about any tree D, however deep, that almost always outputs an active node of a target row b: either there is a tree of depth $O(q^2)$, independent of the depth of D, that outputs an active node of row b with probability $1/2$; or there is a tree, about twice as deep as D, that outputs *two* active nodes of row b, hence a solution, with probability $\Omega(1/q)$. The first case lets us replace the composition of two half-jumps by a tree whose depth does not grow with the level; the second ends the search.

Why the dichotomy holds. Think of each accepting leaf of the activity tree of a node y in row b as a *certificate*: at most $2q$ oracle bits that, once seen, guarantee that y is active. Since D almost always outputs an active node, almost every oracle lies in at least one certificate. Either the certificates fit together neatly, each oracle u lying in essentially one of them, or they do not. If they do, D's depth was never needed: a greedy procedure locates the certificate containing the oracle with $O(q^2)$ queries, by repeatedly querying the bits of the candidate certificate least likely to overlap the others. If they do not, then many oracles lie in two certificates for different nodes, so they have two active nodes in row b, which is a solution; and the second node is not hidden, because on nearby oracles inside its certificate D outputs it, so running D once more on a slightly perturbed copy of the oracle tends to reveal it. Either way we win.

The algorithm. A subroutine $\textsc{Jump}(v, r)$ takes an active node v and tries to reach the row 2^r below. It *defines*, but does not run, the tree C that composes two half-jumps, and applies the dichotomy lemma to C and the activity trees of the target row. In the first case it runs the shallow tree and jumps; in the second it returns the pair-finding tree, which is as deep as two runs of C, all the way up the recursion, and the main loop runs it once and restarts on failure. C itself is run only when it is likely to produce a solution or such a tree, which ends the computation, so this happens $O(q)$ times per level. The cost of a jump is thus linear in r rather than exponential, and $r = \lceil \log_2 N \rceil$ levels give the bound.

2.4 Separations from PLS

The known worst-case separations from PLS are obtained by proving *resolution-width* lower bounds. A low-width resolution refutation can be viewed as a generalization of a low-height decision tree. In particular, let $F \subseteq \Sigma^n \times \mathcal{O}$ be a total relation defining the search problem "given $x \in \Sigma^n$, find $o \in \mathcal{O}$ such that $(x, o) \in F$." A height-w decision tree solving F queries indices of an input $x \in \Sigma^n$ adaptively for w rounds and determines

some $o \in \mathcal{O}$ such that $(x, o) \in F$. We can view this "success condition" in the following way: for each x, the decision tree queries x on w indices and hence at the end knows some partial assignment $\rho \in (\Sigma \cup \{\star\})^n$ of width w (meaning it has w non-star values) which x extends. It must then be the case that the partial assignment ρ already "fixes a solution," meaning that there exists $o_\rho \in \mathcal{O}$ such that $(x', o_\rho) \in F$ for all x' extending ρ.

We now describe a width-w resolution refutation of F. This is another kind of query procedure for solving F, which also has the property that its current "knowledge" about an input x can be described purely in terms of a width $\leq w$ partial assignment ρ which x extends. Unlike a decision tree, a resolution refutation is permitted to run for an unbounded number of rounds; however, we require that the next query at each step, or the choice to halt and output some solution o, is *purely a function of the current known partial assignment* (and cannot depend otherwise on the history of the query procedure). To allow this process to run for more than w rounds, we permit the query procedure to "forget" the answers to queries it has made in the past in order to free space for the value of new queries while staying within the total budget w on the size of the partial assignment representing its current state. Up to increasing the width by at most 1, we may assume that whenever the current state is a partial assignment of width exactly w, a solution must be determined, and whenever its width is $< w$, a further query is made (possibly forgetting some past queries in the process). With this convention we formally define the model as follows.

Definition 1. A resolution refutation of width w consists of a set $\mathcal{F} \subseteq (\Sigma \cup \{\star\})^n$ of partial assignments together with a function $S \colon \mathcal{F} \times \Sigma \to \mathcal{F}$ such that $\star^n \in \mathcal{F}$ and for all $\rho \in \mathcal{F}$:

- ρ has width at most w.
- If $|\rho| = w$ it *fixes a solution*: there is some $o \in \mathcal{O}$ such that $(x, o) \in F$ for all $x \in \Sigma^n$ extending ρ.
- If $|\rho| < w$, there is some $i = i(\rho) \in [n]$, $\rho(i) = \star$, such that for every $a \in \Sigma$, we have that $S(\rho, a) \subseteq \rho \cup \{i \mapsto a\}$.

Finally, there is a strict total ordering $\prec$ on $\mathcal{F}$ such that $\rho \prec S(\rho, a)$ for all $\rho \in \mathcal{F}$ with $|\rho| < w$ and $a \in \Sigma$.

In the terminology of our previously described query procedure, $\mathcal{F}$ is the set of allowable "states" (partial assignments), and S maps each non-terminal state to its successor based on the value of a query. The acyclicity condition is in fact a strict strengthening of the requirement that the query process reaches a terminal state and outputs a solution on every input; we require this termination condition to be enforced "syntactically" over the transition operations. The connection to PLS is given as follows:

Fact 1 ([BKT14, Kam19])**.** *For any $F \in \mathit{TFNP}^{\mathrm{dt}}$, $F \in \mathit{PLS}^{\mathrm{dt}}$ iff there is a width-*$\mathrm{poly}(\log n)$ *pair $(\mathcal{F}, S)$ that solves F.*

With this perspective we can easily describe the classical separation $\mathsf{PPP}^{\mathrm{dt}} \not\subseteq \mathsf{PLS}^{\mathrm{dt}}$ [BCE+98]. Let $\Sigma = \{0,1\}^n \setminus \{0^n\}$ and $N = 2^n$. A PPP instance is naturally represented as an element $x \in \Sigma^N$, listing the values $x(v)$ for each $v \in \{0,1\}^n$ in lexicographical order. We want to show that no width-$w = \mathrm{polylog}(N)$ refutation can find a collision for x. We can actually prove this for width as large as $N - 2$: we simply observe that for any collision-free partial assignment ρ with at least two $\star$ values, and any v with $\rho(v) = \star$, there exists $a \in \Sigma$ such that $\rho \cup \{v \mapsto a\}$ does not fix any solution; we simply take a to be any value in the set $\Sigma \setminus \{\rho(v') \mid v' \leq N\}$, which must be nonempty.

To prove our actual separation, we need to somehow extend this argument to show that it holds "on average over a random $\boldsymbol{x} \sim \Sigma^N$." Consider some candidate width $w = \mathrm{polylog}(N)$ resolution refutation $\mathcal{F}, S$, satisfying all conditions above except the terminal state condition. Our task will be to show that *with high probability* over $\boldsymbol{x} \sim \Sigma^N$, there exists some terminal state $\rho \in \mathcal{F}$, $\boldsymbol{x} \supseteq \rho$, such that ρ does not determine a PPP solution (meaning there are no two values $v \neq v' \in [N]$ such that $\rho(v) = \rho(v') \in \Sigma$). To approach this, we observe that our previous worst-case lower bound did not exploit a useful property of PPP: if ρ is a partial assignment of width $\mathrm{polylog}(N)$, then for any additional query $v \in [N]$, $\rho \cup \{v \mapsto a\}$ will not fix a PPP solution *with overwhelming probability* over a random $\boldsymbol{a} \sim \Sigma$; in particular the failure probability is at most $\frac{\mathrm{polylog}(N)}{N-1}$. We call this property *robustness*; it implies that for any nonterminal ρ which does not fix a solution,

$$\Pr_{\boldsymbol{x} \sim \Sigma^N}[S(\rho, \boldsymbol{x}_{i(\rho)}) \text{ fixes a solution } \mid \boldsymbol{x} \supseteq \rho] \leq \frac{\mathrm{polylog}(N)}{N-1}$$

A naive strategy would be to start at $\rho = \star^N$ and iteratively apply this bound as we traverse the refutation using S, arguing at each step that the probability we fix a solution is small. The obvious issue here is that the number of steps may far exceed N and so a union bound will fail to establish that a solution-fixing state is never reached. Indeed, requiring the refutation to fail when we specifically follow the path starting at the state $\star^N$ far overshoots our intended goal, which is to find *any* terminal $x \supseteq \rho \in \mathcal{F}$ which does not fix a solution; this will directly correspond to a PLS solution which fails to determine any PPP solution, hence ruling out the correctness of the reduction on this x.

Our approach will instead be more global. Let $\mathcal{E} \subseteq \mathcal{F}$ be the terminal states in $\mathcal{F}$ which do not fix a PPP solution and let $B \subseteq \Sigma^N$ be the "bad" inputs for which there exists $\rho \in \mathcal{E}$ such that $x \supseteq \rho$. We aim to show that $\Pr[\boldsymbol{x} \in B] \geq 1 - (\log N)^{-\omega(1)}$ for a uniformly random $\boldsymbol{x}$, i.e. the reduction to PLS fails on all but a negligible fraction of inputs. To prove this, we will define the auxiliary relation/search problem F': given $x \in \Sigma^N$, find either a PPP solution for x, or else find $\rho \in \mathcal{E}$ such that $x \supseteq \rho$. We then observe that $\mathcal{F}$ now serves as a *worst-case resolution refutation* for the relation F'. It hence suffices to prove the following: if $F \subseteq \Sigma^N \times \mathcal{O}$ is any $\mathsf{TFNP}^{\mathrm{dt}}$ search problem with the aforementioned "robustness" property, and we augment it to another $\mathsf{TFNP}^{\mathrm{dt}}$ problem $F' \subseteq \Sigma^N \times (\mathcal{O} \sqcup \mathcal{O}')$ where (1) for $x \in \Sigma^N, o \in \mathcal{O}$, $(x, o) \in F' \longleftrightarrow (x, o) \in F$ and (2) $\Pr[\exists o' \in \mathcal{O}', (x, o') \in F'] \leq 1 - (\mathrm{polylog}(N))^{-1}$, then F' does not have a $\mathrm{polylog}(N)$-width refutation. The key is that we have offloaded the "average case" condition into a modification of the search problem itself, and the remaining task is to prove a "worst case" resolution width lower bound for this modified search problem.

To prove this resolution bound we utilize a linear-algebraic embedding originally developed by Razborov [Raz03] as a key step in his *pseudo-width* method. To prove a resolution lower bound for F', it is natural to try to focus on the set of "safe states" ρ which fix no PPP solution and which admit at least one extension with no $\mathcal{O}'$ solution, and argue that this set is closed under "query/forget" operations. This would immediately yield the width lower bound for F', since any width-w refutation would fail to "escape" this set of safe partial assignments (and none of them fix a solution for F'). While this naive safeness property holds for the empty partial assignment and is downward closed (i.e. closed under "forgetting" previous queries), it need not be closed under performing additional queries. The linear-algebraic embedding will provide us with a special downward-closed (and nonempty) subset of these safe states which is in fact perfectly closed under queries: for any $|\rho| < w$ in this set and any i with $\rho(i) = \star$, there exists $a \in \Sigma$ such that $\rho \cup \{i \mapsto a\}$ also lies in the set.

As alluded to above, the only property of PPP we will use is its "robustness" (we have not defined robustness formally at this point). This property is easily shown to hold for the subclass PWPP, and so in Section 5.3 we prove the stronger ROM separation $\mathsf{PWPP} \not\subseteq \mathsf{PLS}$ directly. In Section 5.4 we exhibit a problem in the class $\mathsf{PPA}^{\mathrm{dt}}$ which satisfies the same robustness property and hence obtain the ROM separation $\mathsf{PPA} \not\subseteq \mathsf{PLS}$ as well. Unlike in the case of PWPP it is not obvious how to efficiently construct a robust distribution of PPA; here we require a modification of the previously-discussed construction showing $\mathsf{PPAD} \not\subseteq \mathsf{FP}$ in the ROM.

3 Average-case separation between PPAD and FP

In this section we prove the following theorem, giving an exponential separation between PPAD and FP in the ROM setting.

Theorem 1. *There exists a TFNP search problem $F = \{F_N \subseteq \Sigma^N \times \mathcal{O}\}$ in PPAD satisfying that: every decision tree T with depth $\exp\left((\log N)^{o(1)}\right)$ solves F with probability at most*

$$\Pr_{\boldsymbol{u} \sim \Sigma^N}\left[(\boldsymbol{u}, T(\boldsymbol{u})) \in F_N\right] \leq \exp\left(-(\log N)^{\Omega(1)}\right).$$

In particular, $\mathsf{PPAD} \not\subseteq_{\mathrm{ROM}} \mathsf{FP}$.

We start by defining the hard problem. Let $G = ([N], E)$ be an ℓ-uniform hypergraph, i.e. $E \subseteq \binom{[N]}{\ell}$. We first describe a simplified problem: given a labeling $h\colon [N] \to [\tau]$ find an edge $e = \{v_1, \ldots, v_\ell\} \in E$ such that no label occurs exactly once in the list $h(v_1), \ldots, h(v_\ell)$. That is, for each $i \in [\tau]$ either $i \notin \{h(v_1), \ldots, h(v_\ell)\}$ or it occurs in the list at least twice. The actual problem has two differences: we allow labels to be voided

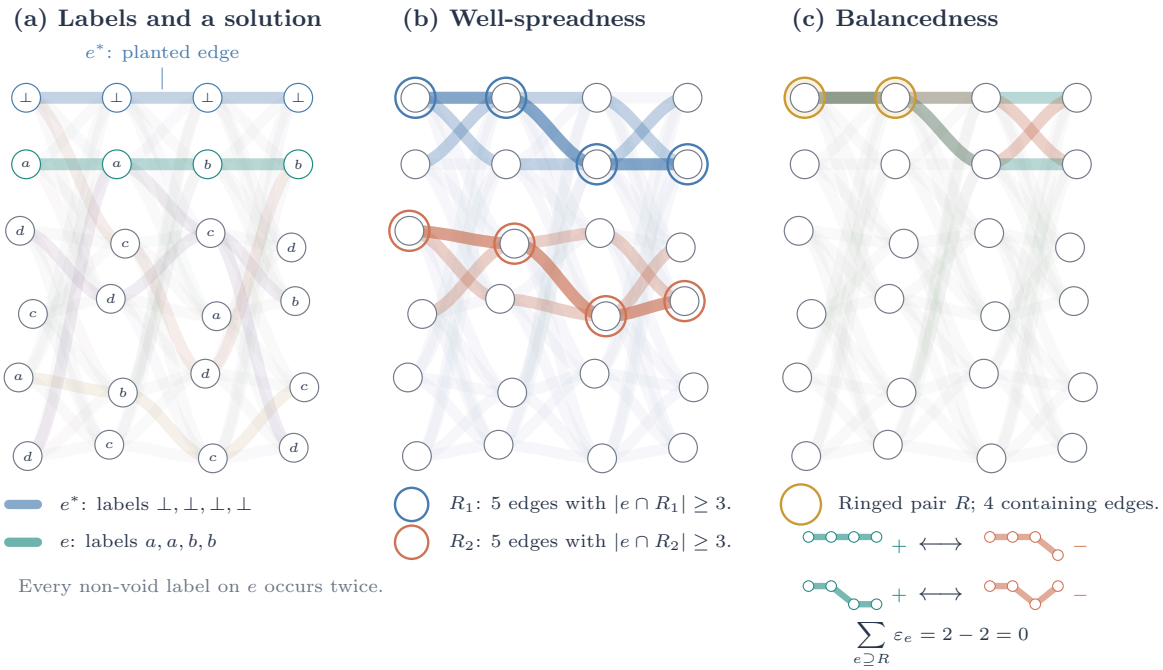

Figure 2: This figure illustrates the setting of the problem SINGLETON.

$(\perp)$, and we plant a special edge $e^* = [\ell]$ and fix all labels in $[\ell]$ to $\perp$. This makes e^* a trivial solution, so the problem asks for *another* solution.

Formally, let $\mathcal{H}$ be the set of functions $h\colon [N] \to (\{\perp\} \cup [\tau])$ satisfying $h([\ell]) = \{\perp\}$. Then the problem $\text{SINGLETON}_G \subseteq \mathcal{H} \times (E \setminus \{e^*\})$ is as follows: given $h \in \mathcal{H}$, find an edge $e = \{v_1, \ldots, v_\ell\} \in E \setminus \{e^*\}$ such that all non-$\perp$ elements in the list $h(v_1), \ldots, h(v_\ell)$ occur at least twice. We will show that there exists G that simultaneously satisfies the following properties:

Hardness For $\boldsymbol{h} \sim \mathcal{H}$, every decision tree T that makes at most $2^{\ell^{1/4}}$ queries to $\boldsymbol{h}$ is very unlikely to solve SINGLETON_G:

$$\Pr[(\boldsymbol{h}, T(\boldsymbol{h})) \in \text{SINGLETON}_G] \leq \exp(-\ell^{\Omega(1)}).$$

PPAD-inclusion $\text{SINGLETON}_G \in \mathsf{PPAD}$;

In Section 3.1 we prove that SINGLETON_G is hard whenever the set of edges E is well-spread:

We say that a subset family $E \subseteq \binom{[N]}{\ell}$ is (r, a, b)-*well-spread* if for every set $S \subseteq [N]$ of size at most a the number of edges $e \in E$ such that $|e \cap S| \geq r$ is at most b. See Figure 2 for an illustration.

Then in Section 3.2 we show that $\text{SINGLETON}_G \in \mathsf{PPAD}$ if E is balanced: $E \subseteq \binom{[N]}{\ell}$ is r-*balanced* if there exists a *signing* $\varepsilon \in \{\pm 1\}^E$ such that for every $R \in \binom{[N]}{r}$ we have $\sum_{e \supseteq R} \varepsilon_e = 0$.

Finally, in Section 3.3 we show that there is an efficient construction of G satisfying both well-spreadness and balancedness.

3.1 PPAD hardness

Theorem 2. *Fix τ to be an even integer in $[\ell/4, \ell/2.6]$. Suppose that E is a $(3\tau/2, q, b)$-well-spread set and $T\colon \mathcal{H} \to E \setminus \{e^*\}$ is a decision tree of depth at most q. Additionally assume that for every $e \in E \setminus \{e^*\}$ we have $|e \cap [\ell]| \leq \tau$. Then the probability that $(\boldsymbol{h}, T(\boldsymbol{h})) \in \text{SINGLETON}_{([N],E)}$ is at most $O(b) \cdot \exp(-\Omega(\tau))$.*

Proof. Associated with each path in T is a partial assignment $\rho \in (\{\star, \perp\} \cup [\tau])^N$ to the underlying variables that are queried along the path. Also let $\mathsf{fix}(\rho)$ denote the set of queries made along path ρ: $\mathsf{fix}(\rho) := \rho^{-1}([\tau] \cup \{\perp\})$.

For every vertex $v \in T$, let ρ_v denote the partial path/restriction from the root of T to vertex v, and for a path ρ let $\text{Leaf}(\rho)$ denote the leaf vertex on this path.

Let $T(\rho)$ be the output of T in $\text{Leaf}(\rho)$. Thus the overall probability that T succeeds on $\boldsymbol{h} \sim \mathcal{H}$ is:

$$\sum_{\rho \text{ a root-to-leaf path in } T} \Pr[\boldsymbol{h} \text{ is consistent with } \rho] \cdot \Pr[(\boldsymbol{h}, T(\rho)) \in \textsc{Singleton}_G \mid \boldsymbol{h} \text{ is consistent with } \rho]$$

For a path ρ in T and an edge $e \in E$, we say that e is **exposed** by ρ if there are at least $3\tau/2$ nodes in e whose h-values are fixed by ρ.

The following Lemma states that for any path ρ in T and for any $e \neq e^*$ such that at most $3\tau/2$ h-values of e are fixed by ρ, the probability that e is a solution is exponentially small. We prove Lemma 3 in Section 3.1.1.

Lemma 3. *Let G, e^*, ℓ, τ be as in Theorem 2. Let ρ be a partial assignment. Fix an edge $e \neq e^*$ and suppose that $|\mathsf{fix}(\rho) \cap e| \leq 3\tau/2$. Then, given $\ell \geq 2.6\tau$ and $\ell = O(\tau)$, we have*

$$\Pr[(\boldsymbol{h}, e) \in \textsc{Singleton}_G \mid \boldsymbol{h} \textit{ extends } \rho] \leq \exp(-\Omega(\tau)).$$

Equipped with Lemma 3, we proceed with the lower bound. There are two cases to consider.

Case I. The first case is if the path ρ in T is such that $T(\rho)$ is *not* exposed. This is the easy case, and we can apply Lemma 3 directly to deduce that $\Pr[(\boldsymbol{h}, T(\rho)) \in \textsc{Singleton}_G \mid \boldsymbol{h} \text{ extends } \rho] \leq \exp(-\Omega(\tau))$.

Case II. The second case is the harder and more interesting case, where ρ is a path in T and $T(\rho)$ has been exposed by ρ. Note that every path ρ could output an exposed edge e, so here we want to argue that the probability (over $\boldsymbol{h}$) that $T(\rho)$ is exposed along its path *and* that $(\boldsymbol{h}, T(\rho)) \in \textsc{Singleton}_G$ is very small. In other words, we need to show that very few assignments h lead to exposed *solutions*.

The rest of the proof will focus on Case II. First, the following claim bounds the number of exposed edges on any given path of T.

Claim 4. *For every path ρ in T, at most b edges are exposed by ρ.*

Proof. Recall that $\mathsf{fix}(\rho)$ is the set of queries made along path ρ, so since $|\mathsf{fix}(\rho)| \leq q$ applying $(3\tau/2, q, b)$-well-spreadness to this set implies that at most b edges $e \in E$ satisfy $|e \cap \mathsf{fix}(\rho)| \geq 3\tau/2$. □

For each path ρ in T, we define an ordered list of non-special edges that become exposed: Let $\text{List}(\rho)$ consist of all pairs (v, e) with $e \neq e^*$ such that e *first* becomes exposed along ρ at vertex v. By Claim 4, $|\text{List}(\rho)| \leq b$. Also we note that the vertices v may not be distinct as more than one edge could be exposed by the same query vertex v. Assume that $\text{List}(\rho)$ is ordered according to time so the edges that are exposed earlier occur before the edges that are exposed later; when more than one edge is exposed at the same vertex, we order them lexicographically. Thus, for every path ρ, $\text{List}(\rho)$ is an ordered list of length at most b.

Now let ρ be a path, and let (v, e) be one of the pairs in $\text{List}(\rho)$. By definition, this means that at vertex v along ρ, e first becomes exposed. Let ρ_v denote the partial transcript of ρ from the root of T up to and including vertex v. Thus ρ_v fixes exactly $3\tau/2$ h-values in e and its other $\ell - 3\tau/2$ coordinates are still not queried.

Now we can apply Lemma 3 here to obtain:

$$\Pr[(\boldsymbol{h}, e) \in \textsc{Singleton}_G \mid \rho_v] \leq \exp(-\Omega(\tau)).$$

Therefore, conditioned on extending ρ_v, the paths in T for which e is good have total probability mass at most $\exp(-\Omega(\tau))$.

Now consider the entire collection of ordered lists $\{\text{List}(\rho)\}_{\rho \in T}$, over all root-to-leaf paths $\rho \in T$. Fix a position $j \in [b]$; this is a position in all of the ordered lists. Define

$$V_j \coloneqq \{(v, e) \mid \rho \text{ is a path in } T \text{ and } (v, e) \text{ is the } j^{th} \text{ element of the list } \text{List}(\rho)\}.$$

Note that although we have shown that along any *single* path ρ, there are at most b exposed edges, since V_j contains the set of jth exposed edges across *all* ρ's, the number of elements in V_j could be very large. Now for a vertex $v \in T$, define E_v to be the event that h is consistent with ρ_v:

$$E_v \coloneqq \{h \in \mathcal{H} \mid h \text{ extends } \rho_v\}.$$

Claim 5. *For every $j \le b$ and distinct pairs $(v,e),(v',e') \in V_j$, the events E_v and $E_{v'}$ are mutually exclusive, and therefore*

$$\sum_{(v,e)\in V_j} \Pr[E_v] \le 1.$$

Proof. Let $(v,e) \neq (v',e') \in V_j$ where (v,e) is the jth element of the list $\mathrm{List}(\rho)$, and (v',e') is the jth element on the list $\mathrm{List}(\rho')$. We want to show that ρ_v and $\rho_{v'}$ must be inconsistent, and therefore the events $E_v, E_{v'}$ are mutually exclusive. If not then assume without loss of generality that $\rho_{v'}$ extends the partial assignment ρ_v, and therefore ρ_v corresponds to a subpath (possibly equal) of $\rho_{v'}$. But this implies that the first j elements must be identical in both lists $\mathrm{List}(\rho)$ and $\mathrm{List}(\rho')$, contradicting our assumption that they disagree on the jth element. □

Now fix $j \in [b]$. By Claim 5, $\sum_{(v,e)\in V_j} \Pr[E_v] \le 1$, and therefore the probability (over assignments h) that the jth exposed edge on the sampled path exists and is a solution is:

$$\sum_{(v,e)\in V_j} \Pr[E_v] \cdot \Pr[(\boldsymbol{h},e) \in \textsc{Singleton}_G \mid E_v].$$

We can use Lemma 3 to bound each second term by $\exp(-\Omega(\tau))$, and by Claim 5, the sum of the first terms is at most 1. Therefore by a union bound over all $j \in [b]$, the probability that some non-special exposed edge on the sampled path is a solution is at most $b \cdot \exp(-\Omega(\tau))$. □

3.1.1 Proof of the main lemma

Finally, we prove Lemma 3, that we restate here for convenience.

Lemma 3. *Let G, e^*, ℓ, τ be as in Theorem 2. Let ρ be a partial assignment. Fix an edge $e \neq e^*$ and suppose that $|\mathit{fix}(\rho) \cap e| \le 3\tau/2$. Then, given $\ell \ge 2.6\tau$ and $\ell = O(\tau)$, we have*

$$\Pr[(\boldsymbol{h},e) \in \textsc{Singleton}_G \mid \boldsymbol{h} \text{ extends } \rho] \le \exp(-\Omega(\tau)).$$

The key probabilistic fact we need here is McDiarmid's inequality:

Theorem 6 ([McD89])**.** *Let $f\colon \Sigma^m \to \mathbb{R}$ be a function with c-bounded differences, i.e. such that for all $x, x' \in \Sigma^m$ that differ in a single coordinate $|f(x) - f(x')| \le c$. Then*

$$\Pr_{\boldsymbol{x}\sim\Sigma^m}[|f(\boldsymbol{x}) - \mathbb{E}[f(\boldsymbol{x})]| \ge \lambda] \le 2\exp(-\lambda^2/(c^2 m)).$$

Proof of Lemma 3. Let $e = \{u_1, \ldots, u_\ell\}$ and assume wlog that $\boldsymbol{h}(u_1), \ldots, \boldsymbol{h}(u_k)$ are either fixed by ρ or by the initial assignment of $h(1) = \cdots = h(\ell) = \perp$. By assumption, $k \le 2.5\tau$, since at most $3\tau/2$ are fixed by ρ and at most τ by the initial assignment. Denote the assignment ρ extended to $[\ell]$ by ρ'.

Then $\boldsymbol{h}(u_{k+1}), \ldots, \boldsymbol{h}(u_\ell)$ are i.i.d. uniform over $[\tau] \cup \{\perp\}$ conditioned on ρ. Let $D \subseteq [\tau]$ be the set of values d such that there is at most one $i \in [k]$ with $\rho'(u_i) = d$. The size of D is at least $\tau/4$: each element in $[\tau] \setminus D$ has to occur at least twice among the non-$\perp$ $h(u_1), \ldots, h(u_k)$, so $|[\tau] \setminus D| \le 3\tau/4$, since the only fixed non-$\perp$ values are assigned by ρ.

We show that at least one label remains a singleton. Define $f_d(x_{k+1}, \ldots, x_\ell)$ to be 1 if d occurs exactly once in $\rho'(u_1), \ldots, \rho'(u_k), x_{k+1}, \ldots, x_\ell$, and 0 otherwise. Then for $I_d \coloneqq \{i \le k \mid \rho'(u_i) = d\}$ we get

$$\Pr[f_d(\boldsymbol{h}(u_{k+1}), \ldots, \boldsymbol{h}(u_\ell)) = 1] = \begin{cases} (1 - 1/(\tau+1))^{\ell-k} & \text{If } |I_d| = 1; \\ (\ell-k)/(\tau+1) \cdot (1-1/(\tau+1))^{\ell-k-1} & \text{If } I_d = \emptyset. \end{cases}$$

In both cases the probability is $\exp(-\Theta((\ell-k)/\tau)) = \Omega(1)$, since $\ell - k = \Theta(\tau)$.

Defining $f = f_1 + \cdots + f_\tau$ observe that f has 2-bounded differences: changing one value of $h(u_i)$ may create at most two singletons. Thus, since $\mathbb{E}[f(\boldsymbol{h}(u_{k+1}), \ldots, \boldsymbol{h}(u_\ell))] = \Omega(\tau)$, Theorem 6 gives $\Pr[f(\boldsymbol{h}(u_{k+1}), \ldots, \boldsymbol{h}(u_\ell)) = 0] \le \exp(-\Omega((\tau/4)^2/(\ell-k))) = \exp(-\Omega(\tau))$. □

3.2 PPAD containment

Recall that $E \subseteq \binom{[N]}{\ell}$ is r-balanced if there is a signing $\varepsilon \in \{\pm 1\}^E$ such that for every $I \in \binom{[N]}{\leq r}$ we have $\sum_{e \in E_I} \varepsilon_e = 0$ for $E_I \coloneqq \{e \in E \mid e \supseteq I\}$. We say that E is *efficiently* r-balanced if the mapping $e \mapsto \varepsilon_e$ is efficiently computable, and for every I there is a poly-time computable involution $\sigma_I \colon E_I \to E_I$ such that $\varepsilon_{\sigma_I(e)} = -\varepsilon_e$, i.e. given I and e there is a poly-time algorithm that computes $\sigma_I(e)$. Wlog we assume $\varepsilon_{e^*} = 1$.

Theorem 7. *Let $G = ([N], E)$ with $E \subseteq \binom{[N]}{\ell}$ be explicit with $\ell = \mathrm{poly}(\log N)$. Suppose that E is* efficiently *τ-balanced. Then $\textsc{Singleton}_G \in$ PPAD.*

We reduce $\textsc{Singleton}_G$ to a uniform family of unsatisfiable polynomial equations over the reals, with integer coefficients. We show that for τ-balanced graphs G this family has a simple (uniform) unary Nullstellensatz (uNS) refutation. The general theorem from [GHJ+24] then immediately implies $\textsc{Singleton}_G \in$ PPAD in the *black-box* model. Our proof does not formally rely on this theorem. We will show that *efficient* τ-balancedness allows the argument to hold in the uniform setting.

Polynomial encoding of $\textsc{Singleton}_G$. We define an unsatisfiable system of polynomials, $\mathcal{P}_G$. The underlying variables of it are $h_{v,j}$ where $v \in [N]$ and $j \in [w]$, where $w = \lceil \log(\tau + 1) \rceil$. The (semantically boolean) variables $h_{v,1}, \ldots, h_{v,w}$ encode the value of $h(v) \in \{\perp\} \cup [\tau]$: the binary representations of elements in $[\tau]$ encode those elements, all other strings encode $\perp$. Since the encoded function $h \in \mathcal{H}$, we also define $h_{i,j} = 0$ for all $i \in [\ell]$, $j \in [w]$.

We need to represent the fact that $(h, e) \notin \textsc{Singleton}_G$ as a polynomial in $\{h_{v,j}\}_{v \in [N], j \in [w]}$ for a fixed $e = \{v_1, \ldots, v_\ell\}$. We want to count occurrences of each non-$\perp$ value in $h(v_1), \ldots, h(v_\ell)$. That is, for $\beta \in [\tau]$, define

$$n_\beta(e; h) \coloneqq \big|\{v \in e \mid h(v) = \beta\}\big|,$$

$n_\beta(e; h)$ counts the number of times the value β appears in the sequence.

The function $n_\beta(e; h)$ can be expressed as a low degree polynomial in the variables $\{h_{v,j}\}$: writing $[h(v) = \beta]$ to represent the product of literals $h_{v,j}$ expressing $h(v) = \beta$, $n_\beta(e; h)$ is computed by the polynomial $\sum_{v \in e} [h(v) = \beta]$ of degree at most w.

Now we define the following system of polynomial equations:

$$\mathcal{P}_G = \{p_e(h) = 0 \mid e \in E \setminus \{e^*\}\}, \quad \text{where} \quad p_e(h) \coloneqq \prod_{\beta \in [\tau]} \left(1 - \sum_{v \in e} [h(v) = \beta]\right).$$

Note that each p_e is a polynomial of degree τw. Also, with respect to boolean variables $h_{v,j}$, the polynomial $1 - \sum_{v \in e} [h(v) = \beta]$ is zero exactly when β occurs once. Therefore we have:

$$p_e(h) \neq 0 \iff \text{no value in } [\tau] \text{ occurs exactly once.}$$

Unary Nullstellensatz and PPAD. Göös, Hollender, Jain, Maystre, Pires, Robere, and Tao [GHJ+24, Lemma 5] prove that $\mathsf{PPAD}^{\mathrm{dt}}$-inclusion is equivalent to having a *Unary Nullstellensatz* (uNS) proof. More precisely, suppose that $a_1, \ldots, a_M \in \mathbb{Z}[x_1, \ldots, x_N]$ are multilinear polynomials each depending on $\mathrm{poly}(\log N)$ variables. Then a Nullstellensatz refutation is a collection of polynomials $c_i \in \mathbb{Z}[x_1, \ldots, x_N]$ and $d_1, \ldots, d_N \in \mathbb{Z}[x_1, \ldots, x_N]$ such that $\sum_{i \in [M]} a_i \cdot c_i + \sum_{i \in [N]} (x_i^2 - x_i) d_i = 1$. Then Nullstellensatz degree is the maximum degree of $a_i c_i$ and *unary* Nullstellensatz size is the sum of magnitudes of all coefficients in $c_1, \ldots, c_M$.

Lemma 8 ([GHJ+24, Lemma 5])**.** *Let $S \coloneqq \{(x, i) \in \{0,1\}^N \times [M] \mid a_i(x) \neq 0\}$, and suppose there exists a unary Nullstellensatz refutation of $\{a_i\}_{i \in [M]}$ of degree $\mathrm{poly}(\log N)$ and size $2^{\mathrm{poly}(\log N)}$ then $S \in \mathsf{PPAD}^{\mathrm{dt}}$.* [3]

[3] [GHJ+24] only formulate this for CNF search, but it is easy to see that a $\mathrm{poly}(\log N)$-local multilinear polynomial can be converted to a quasipolynomial collection of clauses.

Totality of Singleton$_G$ and PPAD upper bound. First we show that SINGLETON$_G$ is total by proving that the underlying system of polynomial equations, $\mathcal{P}_G$ is unsatisfiable. Then, using the general equivalence between PPAD and unary Nullstellensatz we show that SINGLETON$_G$ has an efficient PPAD formulation.

Lemma 9 (Totality of SINGLETON$_G$)**.** *In the setting of Theorem 7 the system $\mathcal{P}_G$ of polynomial equations is unsatisfiable, equivalently* SINGLETON$_G$ *is total. Furthermore, $\mathcal{P}_G$ has the following simple* uNS *refutation (modulo the boolean axioms $x^2 - x$ for all variables x):*

$$1 = -\sum_{e \in E \setminus \{e^*\}} \varepsilon_e p_e.$$

Proof. Expand the τ factors defining p_e:

$$p_e = \sum_{B \subseteq [\tau]; \kappa \in e^B} (-1)^{|B|} \prod_{\beta \in B} [h(\kappa_\beta) = \beta].$$

That is, each term specifies a subset $B \subseteq [\tau]$, and a list of $|B|$ nodes κ that all belong to e. Now define the set of nodes I, associated with this term:

$$I \coloneqq \{\kappa_\beta \mid \beta \in B\} \subseteq [N], \quad |I| \le |B| \le \tau.$$

If $|I| < |B|$, the coefficient of the term in p_e is 0 (by boolean axioms). If $|I| = |B|$ the coefficient is $(-1)^{|B|}$ if $e \supseteq I$, otherwise it is 0. Therefore, since $|I| \le \tau$, the τ-balancedness implies that the coefficient of the term in $\sum_{e \in E} \varepsilon_e p_e$ is $(-1)^{|B|} \cdot \sum_{e \supseteq I} \varepsilon_e = 0$.

On the other hand, for $e^* = [\ell]$, by definition we have $n_\beta(e^*; h) = 0$ since $h(v) = \perp$ for all $v \in e^*$. Therefore all of its counts $n_\beta(e^*; h)$ are zero, and consequently we have $p_{e^*}(h) = 1$ and $\varepsilon_{e^*} = 1$. Removing this polynomial $p_{e^*}(h)$ from the sum we have $1 = -\sum_{e \in E \setminus \{e^*\}} \varepsilon_e p_e(h)$.

Now this implies that $\mathcal{P}_G = \{p_e(h) \mid e \in E \setminus \{e^*\}\}$ is unsatisfiable: suppose for sake of contradiction that there is an assignment to the underlying variables h such that for every $e \in E \setminus \{e^*\}$, $p_e(h) = 0$. Then $\sum_{e \in E \setminus \{e^*\}} \varepsilon_e p_e(h) = 0$, but this contradicts the fact that the sum is identically 1 (as argued above).

Thus, we have not only shown that $\mathcal{P}_G$ is unsatisfiable, but have given a unary Nullstellensatz refutation $\sum_{e \in E \setminus \{e^*\}} \varepsilon_e p_e = -1$, where each $\varepsilon_e \in \{\pm 1\}$. Therefore, the sum of the absolute values of the coefficients in p_e is at most $(1 + \ell^2)^\tau$ for each $e \in E$, and therefore the refutation has degree $\mathrm{poly}(\ell)$ and unary size $|E|(1 + \ell^2)^\tau \le 2^{O(\ell \log N + \tau \log \ell)} = 2^{\mathrm{polylog} N}$. □

Efficiency of inclusion. By Lemma 8 and Lemma 9, SINGLETON $\in$ PPAD^{dt}. In this section we make the inclusion explicit. Using the efficiency of τ-balancedness we can complete the proof following the one by [GHJ+24] exactly. Let us introduce the canonical PPAD-complete problem:

Definition 2. END-OF-LINE problem is defined with two arrays SUCC, PRED $\in [N]^N$; solutions are

- 1 if PRED$_1 \neq 1$ or SUCC$_1 = 1$ or PRED$_{\mathrm{SUCC}_1} \neq 1$; (distinguished node is not a source)
- $i \in [N] \setminus \{1\}$ if SUCC$_{\mathrm{PRED}_i} \neq i$. (non-canonical source)
- $i \in [N]$ if PRED$_{\mathrm{SUCC}_i} \neq i$. (sink)

Proof of Theorem 7. We instantiate the proof of [GHJ+24, Lemma 5] in our simpler case (the refutation simply adds the axiom polynomials with coefficients $\{\pm 1\}$) and verify that the inclusion is efficient. The nodes of the END-OF-LINE formulation are pairs (e, u) with $e \in E$ and $u \in (\{\perp\} \cup e)^\tau$. Each pair corresponds to an *occurrence* of a monomial $m_u \coloneqq \prod_{i \in [\tau]:\, u_i \neq \perp} [h(u_i) = i]$ in p_e. We do not open up the representations of $[h(u_i) = i]$. For each (e, u) the coefficient of the corresponding monomial in p_e is $c(e, u) \coloneqq (-1)^{|\{i \mid u_i \neq \perp\}|}$. The edges in the END-OF-LINE instance are either *monomial-preserving*, they express the cancellation of two occurrences of the same monomial with opposite signs; or hyperedge-preserving, they express the semantic behavior of the polynomial $p_e(h)$. See Figure 3 for an example.

Monomial-preserving For (e, u) identify the set S of non-$\perp$ elements of u. By τ-balancedness there exists a mapping $\sigma_S \colon E_S \to E_S$ that matches up edges satisfying $e \supseteq S$ with the property $\varepsilon_e = -\varepsilon_{\sigma_S(e)}$. Then we define $A(e, u) \coloneqq (\sigma_S(e), u)$.

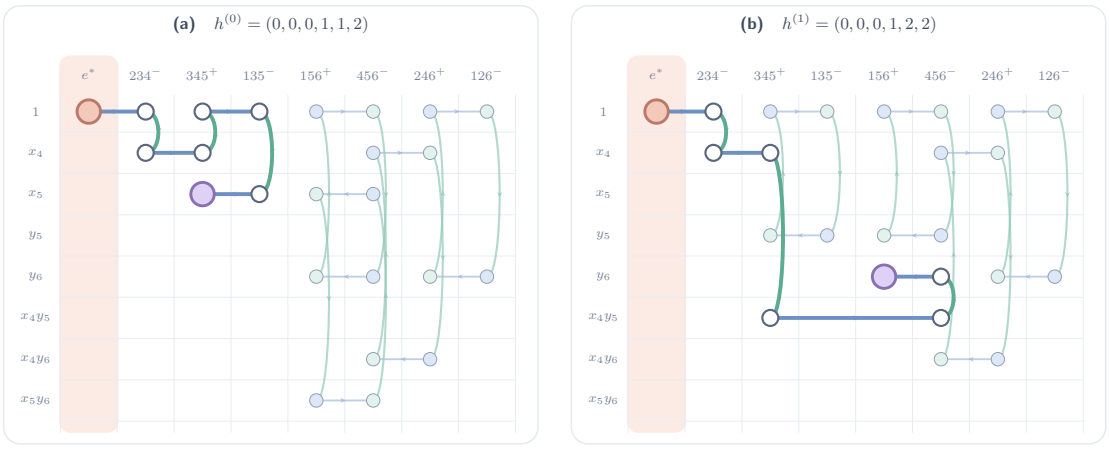

Figure 3: An End-of-Line instance for $G = ([6], E)$ with $E = \{123, 234, 345, 135, 156, 456, 246, 126\}$, $e^* = 123$, and $\varepsilon_e = (-1)^{|e \cap \{4,5,6\}|}$, writing $abc = \{a, b, c\}$. Here $x_v = [h(v) = 1]$, $y_v = [h(v) = 2]$, and 0 denotes $\perp$; only monomial occurrences evaluating to 1 are shown. Blue edges (A) cancel equal monomials across opposite-sign hyperedges; green edges (B_h) match opposite coefficients within a hyperedge. Highlighted paths connect the source $(e^*, \perp^2)$ (peach) to sinks (purple) on non-special solution edges.

Hyperedge-preserving Fix a hyperedge $e \in E$. For all nodes (e, u) if $m_u(h) = 1$ we define $B_h(e, u) \coloneqq (e, u')$ such that m_u and $m_{u'}$ have opposite coefficients in $p_e(h)$ and $B_h(e, u') = (e, u)$. The mapping B_h can be computed efficiently, since the set of u for which monomials $m_u \in p_e$ evaluate to 1 is a simple product set: $\prod_{i \in [\tau]}(\{\perp\} \cup \{v \in e \mid h(v) = i\})$: given a monomial m_u we can compute its rank in the lexicographic order among all monomials with the same coefficient and map it to a monomial with the same rank, but opposite coefficient. If no such monomial exists, define $B_h(e, u) = \perp$. Thus, B_h is not correctly defined iff there are different numbers of monomials with $+1$ and -1 coefficient, i.e. $p_e(h) \neq 0$.

Finally we are ready to define successors and predecessors using the matching A and B_h: if the monomial m_u evaluates to 0 by h, create a self-loop in the node (e, u); otherwise if $\varepsilon_e \cdot c(e, u) \cdot m_u(h) = 1$, take $\textsc{Succ}_{e,u} \coloneqq A(e, u)$ and $\textsc{Pred}_{e,u} \coloneqq B_h(e, u)$, if $B_h(e, u) = \perp$ set $\textsc{Pred}_{e,u} = (e, u)$. If $\varepsilon_e \cdot c(e, u) \cdot m_u(h) = -1$ swap Succ and Pred.

By the definition of the matchings A and B_h, all the nodes (e, u) with $m_u(h) = 1$ are matched, unless $p_e(h) \neq 0$, hence, an unmatched node in the constructed End-of-Line instance reveals a solution. The exception is the node $(e^*, \perp^\tau)$, that we set to be the canonical source. □

3.3 Construction of the graph

In this section, we construct $G = ([N], E)$ that satisfies the requirements of both Theorem 2 and Theorem 7:

Theorem 10. *Suppose ℓ is large enough power of 4. Then there exists $N = 2^{\mathrm{poly}(\ell)}$ and an explicit $G = ([N], E)$ such that $E \subseteq \binom{[N]}{\ell}$ is*

- *$(3\tau/2, \exp((\log N)^{\Omega(1)}), \exp(o(\ell)))$-well-spread;*
- *efficiently τ-balanced;*
- *every $e \in E \setminus \{e^*\}$ satisfies $|e \cap e^*| \leq \tau$;*

where $\tau = 3\ell/8$.

The code. The construction of G comes from a folded Reed-Solomon error correcting code. We identify $[N]$ with a set $[\ell] \times \Gamma$, identifying a codeword $c \in \Gamma^\ell$ with a hyperedge $e_c \coloneqq \{(i, c_i) \mid i \in [\ell]\}$. The zero codeword 0^ℓ corresponds to the planted edge $e^* = e_{0^\ell} = [\ell]$.

Then well-spreadness of E corresponds exactly to list-recovery of the code C: we say that $C \subseteq \Gamma^\ell$ is (δ, a, b)-list recoverable if for every $S_1, \ldots, S_\ell \subset \Gamma$ with $|S_i| \le a$, the number of codewords $(c_1, \ldots, c_\ell) \in C$ with $|\{i \in [\ell] \mid c_i \in S_i\}| \ge (1-\delta)\ell$ is at most b. The following claim is trivial:

Claim 11. *If $C \subseteq \Gamma^\ell$ is (δ, a, b)-list recoverable then $E \coloneqq \{e_c \mid c \in C\}$ is $((1-\delta)\ell, a, b)$-well-spread.*

In order to verify the τ-balancedness we need to fix a specific construction of a list-recoverable C. Given a field $\mathbb{F}$ with $|\mathbb{F}| = n+1$ and a generator $\gamma \in \mathbb{F}^*$ define

$$\mathsf{ReedSolomon}_{\mathbb{F},\gamma,d} \coloneqq \{c \in \mathbb{F}^n \mid c_i = f(\gamma^{i-1}) \text{ where } f \in \mathbb{F}[x] \text{ with } \deg(f) \le d\},$$

and $\mathsf{FoldedRS}^m_{\mathbb{F},\gamma,d} \coloneqq \{((c_1, \ldots, c_m), (c_{m+1}, \ldots, c_{2m}), \ldots, (c_{n-m+1}, \ldots, c_n)) \mid c \in \mathsf{ReedSolomon}_{\mathbb{F},\gamma,d}\}$.

The following fact is a specialization of [Rud07, Section 3.6] used (with slightly different parameters) in [YZ24].

Corollary 12. *Let ℓ be a large enough power of 4 and $\Gamma = \mathbb{F}_{\ell^2}^{\ell-1}$. Then $\mathsf{FoldedRS}^{\ell-1}_{\mathbb{F}_{\ell^2},\gamma,3\ell(\ell-1)/8} \subseteq \Gamma^{\ell+1}$ is $(9/16, \ell^{\ell^{1/4}}, \ell^{2\sqrt{\ell}})$-list recoverable.*

We are now ready to prove the theorem.

Proof of Theorem 10. By applying Corollary 12 we get $\Gamma = \mathbb{F}_{\ell^2}^{\ell-1}$, and $C' \coloneqq \mathsf{FoldedRS}^{\ell-1}_{\mathbb{F}_{\ell^2},\gamma,3\ell(\ell-1)/8} \subseteq \Gamma^{\ell+1}$ that is $(9/16, \ell^{\ell^{1/4}}, \ell^{2\sqrt{\ell}})$-list recoverable. Let $C \coloneqq \{(c_1, \ldots, c_\ell) \mid (c_1, \ldots, c_{\ell+1}) \in C'\}$ be the punctured version of C'. Since the codewords in C' can be uniquely recovered by their marginals from C, the list-recoverability property is retained with a slightly weaker relative parameter: $1 - (1 - 9/16)(\ell+1)/\ell = 9/16 - 7/(16\ell)$. Hence by Claim 11 we have that $E_C \coloneqq \{e_c \mid c \in C\}$ is $(7\ell/16 + O(1), \ell^{\ell^{1/4}}, \ell^{2\sqrt{\ell}})$-well-spread, which trivially implies the required well-spreadness since $3\tau/2 = 9\ell/16 \ge 7\ell/16 + O(1)$ for large enough ℓ.

Let us check the distance property of E_C. For two edges $e \ne e' \in E_C$ let c and c' be the codewords from C corresponding to them. Then $|e \cap e'| \le |\{i \in [\ell] \mid c_i = c'_i\}| \le d/m = \tau$. The last inequality holds, since otherwise two polynomials agree on $> d$ points which implies $c = c'$.

Finally, we establish efficient τ-balancedness of E_C. Take a point $x = \gamma^{m\ell+1} \in \mathbb{F}_{\ell^2}$ and fix a function $\kappa \colon \mathbb{F}_{\ell^2} \to \{\pm 1\}$ that satisfies $\sum_{y \in \mathbb{F}_{\ell^2}} \kappa(y) = 0$. Then for an edge e that corresponds to a codeword defined by the polynomial p_e define $\varepsilon_e \coloneqq \kappa(p_e(x))$. Now fix some $R \in \binom{[\ell]\times\Gamma}{\tau}$ (the case $|R| < \tau$ is handled below). We need to show that $\sum_{e \supseteq R} \varepsilon_e = 0$. If R contains two points whose first coordinates coincide, then no edge contains R and the equality is trivial. Then R has at most one point in each set $\{i\} \times \Gamma$, so the condition $e \supseteq R$ fixes the polynomial p_e in $\tau m = d$ points. Fixing $p_e(x) = y$ to any value $y \in \mathbb{F}_{\ell^2}$ fixes the polynomial p_e, so there exists exactly one such polynomial q_y for every y. Hence $\sum_{e \supseteq R} \varepsilon_e = \sum_{y \in \mathbb{F}_{\ell^2}} \kappa(y) = 0$. In order to show that there is an efficient involution that matches the edges of opposite signs, choose $\kappa(y) \coloneqq (-1)^{\text{the first bit in the representation of } y}$. Then in order to match-up the opposite-sign edges, given e and a valid R compute $y = p_e(x)$ and take y' be the result of flipping the first bit in y. Then let e' be the result of the interpolation of a polynomial with $p_{e'}(x) = y'$ and the d points defined by R.

Now let us verify τ-balancedness for $R \in \binom{[\ell]\times\Gamma}{<\tau}$, again we may assume wlog that R contains at most one point in each set $\{i\} \times \Gamma$, let $I \coloneqq \{i \in [\ell] \mid R \cap \{i\} \times \Gamma \ne \emptyset\}$. Fix a canonical $I' \supseteq I$ of size exactly τ, and define $\sigma_R(e) \coloneqq \sigma_{R'}(e)$ where $R' \coloneqq (I' \times \Gamma) \cap e$. □

Proof of Corollary 12 We use the following fact.

Fact 2 (restatement of [YZ24, Lemma 4.3], originally from [Rud07, Section 3.6])**.** *Suppose m divides $n = |\mathbb{F}| - 1$. Let α, β be positive integers and $s \le m$, $\delta \in (0,1)$ be a real and suppose the following inequalities hold:*

$$\frac{(1-\delta)n}{m} \ge \left(1 + \frac{s}{\beta}\right) \cdot \frac{(n\alpha d^s)^{1/(s+1)}}{m - s + 1} \qquad (\beta + s)\left(\frac{n\alpha}{d}\right)^{1/(s+1)} < |\mathbb{F}|.$$

Then $\mathsf{FoldedRS}^m_{\mathbb{F},\gamma,d}$ is $(\delta, \alpha, |\mathbb{F}|^s)$-list recoverable.

Proof of Corollary 12. Instantiate Fact 2 with the following parameters:

$$|\mathbb{F}| = \ell^2; \quad m = \ell - 1; \quad \tau = 3\ell/8; \quad s = \sqrt{\ell}; \quad \beta = \ell; \quad d = \tau m.$$

Then for $\delta \in (0,1)$, $\alpha \in \mathbb{Z}_{>0}$, it suffices to verify that:

$$(1-\delta) \geq 3/8(1+o(1))\alpha^{1/(1+\sqrt{\ell})} \qquad (1+o(1))(8\alpha/3)^{1/(1+\sqrt{\ell})} < \ell,$$

for the conclusion of Fact 2 to hold. Setting $\alpha = \ell^{\ell^{1/4}}$ and $\delta = 9/16$, we get the claim. □

3.4 Assembling the proof of Theorem 1

Proof of Theorem 1. Let ℓ be a sufficiently large power of 4 and use the graph constructed in Theorem 10, with parameters $N = \ell|\Gamma| = \ell^{2\ell-1}, \tau = 3\ell/8, a = \ell^{\ell^{1/4}}, b = \ell^{2\sqrt{\ell}}$. By Theorem 2, every decision tree of depth at most a solves $\textsc{Singleton}_G$ with probability at most

$$O(b)\exp(-\Omega(\tau)) = \exp\big(O(\sqrt{\ell}\log\ell) - \Omega(\ell)\big) = \exp(-\Omega(\ell)).$$

Since $\log N = \Theta(\ell \log \ell)$, this covers every depth $\exp((\log N)^{o(1)})$ and gives success at most $\exp(-(\log N)^{\Omega(1)})$. Efficient τ-balancedness and $\ell = \mathrm{poly}(\log N)$ give $\textsc{Singleton}_G \in \mathsf{PPAD}$ by Theorem 7.

Set $\Sigma = \{\perp\} \cup [\tau]$ and define $(u, e) \in F_N$ iff $(h_u, e) \in \textsc{Singleton}_G$, where $h_u(i) = \perp$ for $i \in [\ell]$ and $h_u(i) = u_i$ otherwise. Then $F_N \in \mathsf{PPAD}$, and uniform $u \in \Sigma^N$ induces uniform $h_u \in \mathcal{H}$. Fixing the ignored coordinates $u_1, \ldots, u_\ell$ and averaging transfers the same decision-tree lower bound to F_N, proving the theorem. □

3.5 Corollary: EoPL is not in FP

We will use the following claim that allows us to combine two TFNP search problems. The proof is straightforward.

Claim 13. *Let $F = \{F_N \subseteq \Sigma^N \times \mathcal{O}\}$ and $G = \{G_N \subseteq \Sigma^N \times \mathcal{O}'\}$ be two arbitrary TFNP search problems. Define $H = \{H_N \subseteq \Sigma^N \times (\mathcal{O} \cup \mathcal{O}')\}$ by*

$$(u, v) \in H_N \iff (u, v) \in F_N \text{ or } (u, v) \in G_N.$$

Then,

- *H is efficiently reducible to either of F and G.*
- *If any depth-q decision tree solves F and G with probability at most ε_F and ε_G over $\boldsymbol{u} \sim \Sigma^N$, respectively, then any depth-q decision tree solves H with probability at most $\varepsilon_F + \varepsilon_G$ over $\boldsymbol{u} \sim \Sigma^N$.*

The following is a direct corollary from the hardness of PPAD (Theorem 1), the hardness of PLS ([BG20])[4], the collapse $\mathsf{PPAD} \cap \mathsf{PLS} = \mathsf{EoPL}$ ([GHJ+22, Theorem 1]), and Claim 13. We note that $\varepsilon_{\mathsf{PLS}}$ supplied by [BG20, Theorems 36 and 43] is $o(1)$.

Corollary 14. *There exists a TFNP search problem $F = \{F_N \subseteq \Sigma^N \times \mathcal{O}\}$ in EoPL satisfying that: every decision tree T with depth at most $\exp\left((\log N)^{o(1)}\right)$ solves F with probability at most*

$$\Pr_{\boldsymbol{u} \sim \Sigma^N}\left[(\boldsymbol{u}, T(\boldsymbol{u})) \in F_N\right] = o(1).$$

In particular, $\mathsf{EoPL} \not\subseteq_{\mathrm{ROM}} \mathsf{FP}$.

[4] [BG20] proved the separation in the random oracle model. However, it can be verified that their proof implies a decision-tree lower bound in the black-box setting.

4 Efficient search for UEoPL in the black-box ROM

In this section we show that UEoPL is easy in the random oracle model: every instance that is generated from the oracle by shallow decision trees can be solved, with high probability over the oracle, by a shallow decision tree. We use the definition of UEoPL from [GHJ+24, Section 3.2], which we reproduce here.

Definition 3 (UEoPL$_N$ [GHJ+24, Section 3.2]). Let $N \geq 2$. As input we are given a successor $\textsc{Succ}_u \in [N] \cup \{\bot\}$ for each $u \in [N] \times [N]$ and a predecessor $\textsc{Pred}_u \in [N] \cup \{\bot\}$ for each $u \in \{2, \dots, N\} \times [N]$. A node $(i, j) \in [N-1] \times [N]$ is *active* if $\textsc{Succ}_{ij} = k \neq \bot$ and $\textsc{Pred}_{i+1,k} = j$, otherwise it is *inactive*; a node $(i, j) \in \{N\} \times [N]$ is active if $\textsc{Succ}_{ij} \neq \bot$ and inactive otherwise. A node u is a *proper sink* if u is inactive but some active node has u as a successor. A node (i, j) is a *proper source* if it is active, $i < N$, and no active node has (i, j) as a successor. The goal is to output any of the following

1. $(1, 1)$, if $(1, 1)$ is inactive, (inactive distinguished source)
2. (N, j), if (N, j) is active, (active sink)
3. (i, j), if (i, j) is a proper sink, (proper sink)
4. (i, j), if $(i, j) \neq (1, 1)$ and (i, j) is a proper source, (proper source)
5. (i, j) and (i, j'), if $j \neq j'$ and both nodes are active. (two parallel lines)

Formally, UEoPL$_N$ is the set of all pairs (x, z) such that x is an input as above and z is one of the outputs listed for x. We call x an *instance* and z a *solution* of x.

Here the successor $\textsc{Succ}_v = k \neq \bot$ of a node $v = (i, j)$ with $i < N$ is interpreted as naming the node $(i + 1, k)$ on the next row, and we also call this node the successor of v. A solution always exists: if $(1, 1)$ is active, then following successors from $(1, 1)$ traces a path that advances by one row with every step and ends in a proper sink or in an active node of the last row.

A proper sink $u = (i, j)$ is locally verified by checking that $i > 1$, $\textsc{Pred}_u = p \neq \bot$, $\textsc{Succ}_{i-1,p} = j$, and u is inactive. The two pointer equalities certify an incoming edge from an active node, and inactivity is checked with at most two further pointer evaluations.

Instances in the random oracle model. An instance is generated from the oracle $u \in \{0, 1\}^{\mathbb{N}}$ by a decision forest ω: every successor $\textsc{Succ}_v$ and predecessor $\textsc{Pred}_v$ of the instance $\omega(u)$ is computed by a decision tree that queries bits of u and outputs an element of $[N] \cup \{\bot\}$. We write q for the maximum depth of the decision trees in ω.

Theorem 15. *Let $N \geq 2$ and $q \geq 1$, and let ω be a decision forest that generates instances of UEoPL$_N$ and computes every pointer in depth at most q. Then there is a decision tree T of depth $\mathrm{poly}(q) \cdot \log N$ such that*

$$\Pr_{\boldsymbol{u} \sim \{0,1\}^{\mathbb{N}}} \left[(\omega(\boldsymbol{u}), T(\boldsymbol{u})) \in \textit{UEoPL}_N \right] \geq \frac{2}{3}.$$

Moreover, for every $u \in \{0, 1\}^{\mathbb{N}}$, the output $T(u)$ is either a solution of $\omega(u)$ or the failure symbol fail.

Remark on upper bounds in the decision tree setting. Theorem 15 bounds the number of oracle queries, not the computation: the tree T is shown to exist, and nothing is claimed about computing it. This is likely unavoidable, since UEoPL is hard in the white-box random oracle model under a complexity assumption [CHK+19a].

Restrictions. A *restriction* ρ assigns values to finitely many bits of the oracle, and $\mathsf{cube}(\rho) \subseteq \{0, 1\}^{\mathbb{N}}$ is the set of oracles that agree with ρ. We write $\Pr_\rho$ and $\mathbb{E}_\rho$ for probability and expectation over $\boldsymbol{u} \sim \mathsf{cube}(\rho)$. A decision tree on $\mathsf{cube}(\rho)$ queries only bits that are not fixed by ρ.

Decision tree families. Throughout, a *family of decision trees* $f = (f_y)_{y \in Y}$ with $f_y \colon \{0, 1\}^{\mathbb{N}} \to \{0, 1\}$ *accepts* the label $y \in Y$ on u if $f_y(u) = 1$; the failure symbol fail is never accepted. We say that a decision tree $D \colon \{0, 1\}^{\mathbb{N}} \to Y \cup \{\mathsf{fail}\}$ *certifies* the label $y \in Y$ on u if, before outputting y, it has queried all the bits

fixed by some accepting leaf of f_y that u reaches; in that case, f accepts y on every oracle that agrees with the queried bits.

The proof of Theorem 15 relies on the following lemma, which does not refer to UEoPL.

Lemma 16. *For every $q \geq 1$ there are $a = O(q^2)$ and $\eta = \Theta(1/q)$ such that the following holds. Let $f = (f_y)_{y \in Y}$ be a finite family of decision trees $f_y \colon \{0,1\}^{\mathbb{N}} \to \{0,1\}$ of depth at most q and let $D \colon \{0,1\}^{\mathbb{N}} \to Y \cup \{\mathsf{fail}\}$ be a decision tree of depth t such that*

$$\Pr_{\boldsymbol{u} \sim \{0,1\}^{\mathbb{N}}} \left[f \text{ accepts } D(\boldsymbol{u}) \text{ on } \boldsymbol{u} \right] \geq 1 - \eta.$$

Then at least one of the following holds.

1. *There is a decision tree $T_1 \colon \{0,1\}^{\mathbb{N}} \to Y \cup \{\mathsf{fail}\}$ of depth at most a such that $\Pr_{\boldsymbol{u}}[T_1(\boldsymbol{u}) \neq \mathsf{fail}] \geq 1/2$, and T_1 certifies $T_1(u)$ on u whenever $T_1(u) \neq \mathsf{fail}$.*
2. *There is a decision tree $T_2 \colon \{0,1\}^{\mathbb{N}} \to \binom{Y}{2} \cup \{\mathsf{fail}\}$ of depth $O(t+a)$ such that $\Pr_{\boldsymbol{u}}[T_2(\boldsymbol{u}) \neq \mathsf{fail}] \geq \eta$, and T_2 certifies both labels of $T_2(u)$ on u whenever $T_2(u) \neq \mathsf{fail}$.*

In both cases the tree queries only bits that D or some f_y queries.

Remark 1. Lemma 16 is a decision-tree form of [BCG26, Corollary 5.4]. The idea that a search problem with (almost) unique solutions cannot be hard relative to a random oracle goes back to Rudich [Rud88]; his conjecture was proved by Kahn, Saks, and Smyth [KSS10] via the dual BKR inequality, which [GRY25] and then [BCG26] applied to verifiable delay functions and proofs of work. The proof below does not use the BKR inequality.

Intuition behind Lemma 16. In Section 4.1 we prove our UEoPL upper bound (Theorem 15) by applying Lemma 16 recursively, where Y will correspond to the set of nodes in a row of the UEoPL instance, and the decision tree family $(f_y)_{y \in Y}$ will be the depth-$2q$ decision trees checking whether node y is active. Therefore "f accepts $y \in Y$ on $\boldsymbol{u}$" means that node y in this row is active on input $\boldsymbol{u}$, and "D certifies y on $\boldsymbol{u}$" means that D outputs y after reading a certificate that y is active. In this context, Lemma 16 gives a win-win dichotomy applied to this row of nodes. Informally it states: Let D be any decision tree (of potentially very large depth $t \gg q$) that outputs an active node in this row with probability at least $1 - \eta$. Then at least one of the following holds: (1) There is a shallow decision tree T_1 (of depth at most $O(q^2)$) such that $T_1(\boldsymbol{u})$ outputs an active node in this row with probability at least $1/2$, and furthermore $T_1(\boldsymbol{u})$ always outputs either an active node or fail; or (2) There is a decision tree T_2 of depth $O(t+a)$ such that $T_2(\boldsymbol{u})$ outputs *two* distinct active nodes in this row with probability at least η, and furthermore $T_2(\boldsymbol{u})$ always outputs either 2 active nodes or fail.

Why should we expect such a win-win situation? Think of each accepting leaf of an activity tree f_y as a certificate: a small restriction ρ that guarantees that y is active for all u consistent with ρ. The tree D (however deep it is) is assumed to almost always output an active node, so almost every oracle u is active for at least one node, namely the one that $D(u)$ outputs. Now there are two cases: either (1) the certificates corresponding to accepting leaves of $(f_y)_{y \in Y}$ are almost disjoint, or (2) they overlap substantially. In case (1), D was never needed: we can construct a shallow tree T_1 that finds the unique active node in this row by applying a robust version of the classic result (due to Nisan) showing that any Boolean function represented by a disjoint q-DNF can be transformed into a depth $O(q^2)$ decision tree. In case (2) since the certificates overlap, on many input oracles u, $D(u)$ outputs one active node while u is also consistent with a different certificate. Such u's have at least 2 active nodes, which in UEoPL is a solution! In this second case using a sensitivity argument it is shown that the tree T_2 obtained by running D twice, once on u and once on a slightly perturbed copy of u, tends to output two active nodes. In this second case even though the depth of T_2 is close to that of D, we get a big win since T_2 outputs an actual solution. As we explain below, we avoid incurring the expensive cost of running this high depth tree recursively by delaying running it until the end where it is only run once.

4.1 Proof of Theorem 15

Fix ω and q as in Theorem 15. Throughout the algorithm, ρ denotes the restriction that records the answers to all queries made so far. It is a single global object: every subroutine reads it, and every query that any

subroutine makes extends it. For an algorithm A that queries the oracle, we write $\mathsf{Q}(A)$ for the number of queries that A makes, and call $\mathbb{E}_\rho[\mathsf{Q}(A)]$ the expected query complexity of A on $\mathsf{cube}(\rho)$. A property of the instance $\omega(u)$ *holds on* $\mathsf{cube}(\rho)$ if it holds for every $u \in \mathsf{cube}(\rho)$.

Activity trees. For a node v, let $\mathsf{act}_v(u) \in \{0,1\}$ indicate whether v is active in $\omega(u)$. This is computed by a decision tree of depth at most $2q$, which evaluates $\textsc{Succ}_v$ and then, if v is not in the last row and $\textsc{Succ}_v \neq \bot$, the predecessor of the successor of v. For a row $b \in [N]$, let $\mathsf{act}^b := (\mathsf{act}_{(b,j)})_{j\in[N]}$ be the family of these trees for the nodes of row b. Throughout the proof, $a = O(q^2)$ and $\eta = \Theta(1/q)$ are the parameters that Lemma 16 provides for verifier depth $2q$.

We apply Lemma 16 to a tree D on $\mathsf{cube}(\rho)$ and to the trees act^b with the bits of ρ fixed, which does not increase their depth. These trees, and hence T_1 and T_2, query no bit that ρ fixes, so for every event E determined by their outputs, $\Pr_{u\sim\mathsf{cube}(\rho)}[E] = \Pr_{u\sim\{0,1\}^{\mathbb{N}}}[E]$ after renaming the free bits. In particular, a label that T_1 or T_2 certifies is accepted on the cube of the restriction obtained after running the tree; for the activity trees this means that the node is active on that cube.

High-level description of the algorithm. The backbone of the algorithm is the recursive subroutine $\textsc{Jump}(v, r)$, given in Algorithm 1, which jumps from a node $v = (i, j)$ in row i to the target row $b := \min\{i + 2^r, N\}$. It is called only on nodes that are *active on* $\mathsf{cube}(\rho)$, and it returns one of three things: an active node of row b, a solution, or a decision tree that finds a solution with probability at least η (Claim 18). The third kind of output is explained below.

For $r = 0$ the subroutine reads the successor of v and tests whether it is active, which gives either a proper sink or an active node of the next row. For $r \geq 1$, the obvious way to reach row b is by two jumps of 2^{r-1} rows (the solid arrows in Figure 4): let C be the decision tree that runs $\textsc{Jump}(v, r-1)$ and then, from the node v' that this returns, $\textsc{Jump}(v', r-1)$. But running C at every level would double the number of queries with each level, and $\log N$ levels would bring us back to $\Theta(N)$ queries. So, as a rule, we do not run C. Instead we use it as the tree D of Lemma 16, applied with the activity trees of row b as verifiers: since C almost always outputs an active node of row b, the lemma provides one of two trees.

- In the compression case it provides a tree T_1 of depth a, which does not depend on r, that outputs an active node of row b with probability at least $1/2$ (the dashed arrow in Figure 4). We then jump by running T_1 instead of C. Since T_1 outputs fail on some oracles, this may fail; but then we have learned at least one new bit, so we repeat the loop with the enlarged restriction.
- In the collision case it provides a tree T_2 that outputs two active nodes of row b, which is a solution, with probability at least η. This is as good as a jump (in fact better), except for its cost: the depth of T_2 is of the order of $1/\eta$ times the query complexity of C. Running it inside the recursion would multiply the cost of every level by $1/\eta$, which compounds over the $\log N$ levels. So $\textsc{Jump}$ does not run T_2, and instead it returns the tree T_2 itself as its output, and every enclosing call returns it as well: a run of C that outputs a tree ends the loop of the next level in line 8 which returns the tree, and so on up to Algorithm 2. Thus the whole recursion unwinds at once, with no further queries, and Algorithm 2 runs T_2 a single time. If that run fails, Algorithm 2 restarts $\textsc{Jump}((1,1), r^*)$ from the enlarged restriction. In this way the factor $1/\eta$ is paid once, at the top level, rather than once per level.

The only situation in which C itself is run is when it is likely, with probability at least $\eta/2$, to output a solution or a tree (line 8). Either output ends the entire computation of $\textsc{Jump}(v, r)$ as just described, so this happens $O(1/\eta)$ times per level in expectation (Claim 19). In every other situation C outputs an active node of row b with probability more than $1 - \eta/2$; cutting it off after $t = \lceil 2\,\mathbb{E}_\rho[\mathsf{Q}(C)]/\eta \rceil$ queries (line 10) costs another $\eta/2$ by Markov's inequality and yields the tree D of bounded depth that Lemma 16 needs.

It is important for the accounting that lines 7, 9, 10 and 11 of Algorithm 1 make no oracle queries: they define trees and compute probabilities and expectations on $\mathsf{cube}(\rho)$. The recursion is executed only in line 8, and queries are made only in lines 4, 8 and 12.

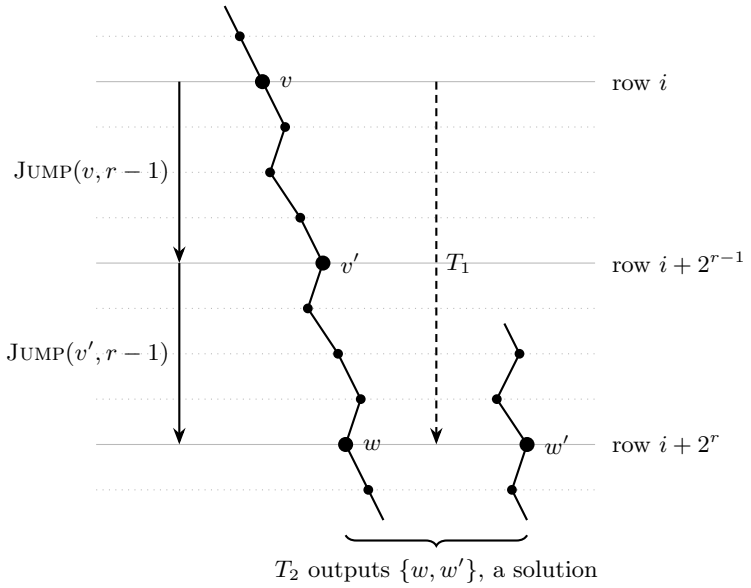

Figure 4: One iteration of the loop in Algorithm 1: the two jumps on the left form C, T_1 (dashed) jumps directly, and T_2 outputs two active nodes w, w' of row $i + 2^r$. The dots indicate active nodes at each layer.

Algorithm 1 Subroutine JUMP in Algorithm 2.

Input: a node $v = (i, j)$ that is active on $\mathsf{cube}(\rho)$, and $r \geq 0$
Output: an active node of row $\min\{i + 2^r, N\}$, a solution, or a decision tree

if $i = N$ **then return** the node v.
Set $b \coloneqq \min\{i + 2^r, N\}$.
if $r = 0$ **then**
    Query $\textsc{Succ}_v$ and $\mathsf{act}_{v'}$, where $v' \coloneqq (i + 1, \textsc{Succ}_v)$ is the successor of v.
    return the solution v' if v' is inactive; the node v' otherwise.
loop
    Let C be the decision tree on $\mathsf{cube}(\rho)$ that runs $\textsc{Jump}(v, r-1)$ and, if it outputs a node v', runs $\textsc{Jump}(v', r-1)$; C outputs the output of the last run.
    if $\Pr_\rho[C$ outputs a solution or a tree on $\boldsymbol{u}] \geq \eta/2$ **then** run C and **return** its output.
    Set $t \coloneqq \lceil 2\,\mathbb{E}_\rho[\mathsf{Q}(C)]/\eta \rceil$.
    Let D be C cut off after t queries, with output fail if C is cut off or does not output a node.
    Apply Lemma 16 on $\mathsf{cube}(\rho)$ to act^b and D. **if** it provides no tree T_1 **then return** the tree T_2.
    Run T_1. **if** T_1 outputs a node v' **then return** the node v'.

Algorithm 2 The main algorithm.

Input: $u \in \{0,1\}^{\mathbb{N}}$
Output: a solution of $\omega(u)$

Query $\mathsf{act}_{(1,1)}$. **if** $(1,1)$ is inactive **then return** $(1,1)$.
Set $r^* \coloneqq \lceil \log_2 N \rceil$.
loop
    Run $\textsc{Jump}((1,1), r^*)$.
    - If it outputs a node or a solution z: **return** z.
    - If it outputs a decision tree G: run G, and **return** its output if it is a solution.

Let $r \geq 0$, let ρ be a restriction, and let $v = (i, j)$ be a node that is active on $\mathsf{cube}(\rho)$. The following

four claims are about $\textsc{Jump}(v, r)$ started from ρ. Claims 17 and 18 are proved together by induction on r; Claims 19 and 20 then follow.

Claim 17. $\textsc{Jump}(v, r)$ *halts on every* $u \in \mathsf{cube}(\rho)$.

Claim 18. $\textsc{Jump}(v, r)$ *outputs one of the following, where* ρ' *is the restriction right before it halts:*

- *a node of its target row that is active on* $\mathsf{cube}(\rho')$*;*
- *a solution on* $\mathsf{cube}(\rho')$*;*
- *a decision tree* G *on* $\mathsf{cube}(\rho')$ *such that* $G(u)$ *is a solution of* $\omega(u)$ *or* fail *for every* $u \in \mathsf{cube}(\rho')$*, and* $\Pr_{\rho'}[G(\boldsymbol{u}) \neq \mathsf{fail}] \geq \eta$.

Proof of Claim 17. The cases $i = N$ and $r = 0$ are immediate, so let $i < N$ and $r \geq 1$. By Claims 17 and 18 for level $r - 1$,

> the tree C of line 7 halts, and it outputs a solution or a tree as in Claim 18, or a node of row b that is active on the cube of the restriction at that time. (1)

Consider an iteration of the loop, at a restriction ρ, that reaches line 9. Then C outputs a solution or a tree with probability less than $\eta/2$ over $\mathsf{cube}(\rho)$, and by Markov's inequality it makes more than t queries with probability less than $\eta/2$. So, by (1), the tree D of line 10 has depth at most t and outputs an active node of row b with probability more than $1 - \eta$, and Lemma 16 applies on $\mathsf{cube}(\rho)$ to act^b and D. If $\textsc{Jump}(v, r)$ does not halt in this iteration, then T_1 was run and output fail. Since $\Pr_\rho[T_1(\boldsymbol{u}) \neq \mathsf{fail}] \geq 1/2$, the tree T_1 cannot output fail without making a query, so the iteration fixes at least one more bit of the oracle. The instance $\omega(u)$ depends on only finitely many bits of u, and no other bit ever needs to be queried, so there are only finitely many iterations. □

Proof of Claim 18. If $i = N$, then the output is the active node v of row $b = N$. If $i < N$ and $r = 0$, the query to $\textsc{Succ}_v$ fixes its successor v' in row b on $\mathsf{cube}(\rho')$. The node v remains active on $\mathsf{cube}(\rho')$, since ρ' extends the starting restriction. If the activity test finds v' inactive, its queried transcript certifies inactivity on $\mathsf{cube}(\rho')$, and the active node v witnesses an incoming edge to v'. Thus v' itself is a proper sink, and hence a solution, throughout $\mathsf{cube}(\rho')$. Otherwise the activity test certifies that v' is active on that cube. If $i < N$ and $r \geq 1$, then $\textsc{Jump}(v, r)$ halts in line 8, 11 or 12.

- In line 8 it outputs the output of C, which is as claimed by (1).
- In line 11 it outputs T_2, which by Lemma 16 outputs a pair of distinct active nodes of row b, that is, a solution, with probability at least η, and fail otherwise.
- In line 12 it outputs a node of row b that T_1 certifies by Lemma 16, hence active on $\mathsf{cube}(\rho')$. □

Claim 19. $\mathbb{E}_\rho[\mathsf{Q}(\textsc{Jump}(v, r))] = O(a(r + 1)/\eta)$.

Proof. For $k \leq r$, let $\boldsymbol{N}_k$ be the number of times that $\textsc{Jump}(\cdot, k)$ is run during $\textsc{Jump}(v, r)$, and let $\boldsymbol{M}_k \leq \boldsymbol{N}_k$ be the number of those times in which C is run in line 8. Then $\boldsymbol{N}_r = 1$, and $\boldsymbol{N}_{k-1} \leq 2\boldsymbol{M}_k$ for $k \in [r]$, because $\textsc{Jump}(\cdot, k - 1)$ is run only inside such a C, and at most twice there. By the test in line 8, each time C is run it outputs a solution or a tree with probability at least $\eta/2$, and then $\textsc{Jump}(v, r)$ halts without running anything further. Hence $\mathbb{E}_\rho[\boldsymbol{M}_k] = O(1/\eta)$ for every $k \in [r]$, and so $\mathbb{E}_\rho[\sum_{k \leq r} \boldsymbol{N}_k] = O(1 + r/\eta)$.

Each time $\textsc{Jump}(\cdot, k)$ is run, the queries that it makes outside of line 8 are the $O(q) = O(a)$ queries of line 4 and the queries of T_1 in line 12. Each time T_1 is run, it makes at most a queries and, with probability at least $1/2$, outputs a node, upon which $\textsc{Jump}(\cdot, k)$ halts. So these are $O(a)$ queries in expectation, and $\mathbb{E}_\rho[\mathsf{Q}(\textsc{Jump}(v, r))] = O(a) \cdot O(1 + r/\eta) = O(a(r + 1)/\eta)$. □

Claim 20. *Every tree that* $\textsc{Jump}(v, r)$ *outputs has depth* $O(a(r + 1)/\eta^2)$.

Proof. A tree that $\textsc{Jump}(v, r)$ outputs is the tree T_2 in line 11 of $\textsc{Jump}(\cdot, k)$ for some $k \in [r]$. Its depth is $O(t + a)$, where $t = O(\mathbb{E}_\rho[\mathsf{Q}(C)]/\eta) = O(ak/\eta^2)$ by Claim 19. □

Proof of Theorem 15. Consider an iteration of the loop in Algorithm 2 that starts from a restriction ρ. By Claims 17–20, $\textsc{Jump}((1, 1), r^*)$ makes $O(ar^*/\eta)$ queries in expectation, and it outputs a solution or a tree G of depth $O(ar^*/\eta^2)$ that outputs a solution with probability at least η. Hence the iteration makes $O(ar^*/\eta^2)$

queries in expectation and returns with probability at least η over $\mathsf{cube}(\rho)$. So whenever Algorithm 2 halts on u, its output is a solution of $\omega(u)$, and its expected query complexity is at most

$$B = O(q) + \frac{1}{\eta} \cdot O\Big(\frac{ar^*}{\eta^2}\Big) = \mathrm{poly}(q) \cdot \log N.$$

Let T be Algorithm 2 stopped after $3B$ queries, with output fail if it has not halted by then. This is a decision tree of depth at most $3B$, because an iteration that does not return makes at least one query, as it returns with positive probability over $\mathsf{cube}(\rho)$. By Markov's inequality, T outputs fail with probability at most $1/3$. □

4.2 Proof of Lemma 16

Let f and D be as in Lemma 16; the parameters a and η are chosen in the proof of Lemma 16 below. The trees f_y and D query only finitely many bits, which we rename to $1, \dots, K$. Below $\boldsymbol{u}$ is uniform over $\{0,1\}^K$, all trees query only these bits, and for a set $A \subseteq \{0,1\}^K$ we write $\Pr[A]$ for $\Pr[\boldsymbol{u} \in A]$. Let

$$\alpha \coloneqq \Pr\big[f \text{ does not accept } D(\boldsymbol{u}) \text{ on } \boldsymbol{u}\big], \qquad \varepsilon \coloneqq \max_E \Pr\big[f \text{ accepts } E(\boldsymbol{u}) \text{ on } \boldsymbol{u}, \text{ and } E(\boldsymbol{u}) \neq D(\boldsymbol{u})\big],$$

where the maximum is over all decision trees $E\colon \{0,1\}^K \to Y \cup \{\mathsf{fail}\}$ of depth at most t.

Let Λ be the set of all accepting leaves of the trees f_y that are reached on some input. For $\tau \in \Lambda$ we write y_τ for the label of the tree that τ belongs to, and $B_\tau \subseteq \{0,1\}^K$ for the subcube of inputs on which f_{y_τ} reaches τ; it fixes at most q bits.

Claim 21. *There is a set $\Lambda' \subseteq \Lambda$ such that $m(u) \coloneqq |\{\tau \in \Lambda' : u \in B_\tau\}|$ satisfies*

$$\mathbb{E}\,|m(\boldsymbol{u}) - 1| \leq \alpha + O(q\varepsilon).$$

Claim 22. *There is an absolute constant $c > 0$ such that the following holds. Let $(B_\tau)_{\tau \in \Lambda'}$ be a finite family of subcubes of $\{0,1\}^K$, each fixing at most q bits, and let $m(u) \coloneqq |\{\tau \in \Lambda' : u \in B_\tau\}|$. If $\mathbb{E}\,|m(\boldsymbol{u}) - 1| \leq c$, then there is a decision tree of depth $O(q^2)$ whose output on u is either fail or some τ with $u \in B_\tau$, and which outputs fail with probability at most $1/2$.*

We prove Claim 21 in Section 4.3 and Claim 22 in Section 4.4.

Proof of Lemma 16. Let c be the constant of Claim 22. We choose $\eta = \Theta(1/q)$ small enough that the bound $\alpha + O(q\varepsilon)$ of Claim 21 is at most c whenever $\alpha \leq \eta$ and $\varepsilon \leq 2\eta$, and we let $a = O(q^2)$ be the depth bound of Claim 22, which we may assume to be at least $2q$. By the hypothesis on D, we have $\alpha \leq \eta$.

Suppose first that $\varepsilon \leq 2\eta$. By Claim 21, there is a set Λ' of accepting leaves with $\mathbb{E}\,|m(\boldsymbol{u}) - 1| \leq c$. Let T_1 run the decision tree of Claim 22 for this family, and output y_τ if that tree outputs τ. Then T_1 has depth at most a and outputs fail with probability at most $1/2$. If it outputs y_τ, then it has queried all bits fixed by B_τ and $u \in B_\tau$, so T_1 certifies y_τ on u.

Suppose now that $\varepsilon > 2\eta$, and let E be a tree that attains the maximum in the definition of ε. Let T_2 run D and E; if $D(u)$ and $E(u)$ are distinct labels in Y, it then runs $f_{D(u)}$ and $f_{E(u)}$ and outputs the pair $\{D(u), E(u)\}$ if both trees accept; in every other case it outputs fail. Then T_2 certifies both labels whenever it does not output fail, it has depth at most $2t + 2q = O(t + a)$, and

$$\Pr[T_2(\boldsymbol{u}) \neq \mathsf{fail}] \geq \varepsilon - \alpha > \eta.$$ □

4.3 Proof of Claim 21

For $\tau \in \Lambda$, let

$$A_\tau \coloneqq \{u \in B_\tau : D(u) = y_\tau\}, \qquad \theta_\tau \coloneqq \Pr[A_\tau]/\Pr[B_\tau], \qquad \Gamma \coloneqq \sum_{\tau \in \Lambda} \Pr[A_\tau](1 - \theta_\tau).$$

The sets A_τ are pairwise disjoint, since $u \in A_\tau$ means that τ is the leaf that the tree $f_{D(u)}$ reaches on u. Their union is the event that f accepts $D(\boldsymbol{u})$, and hence $\sum_\tau \Pr[A_\tau] = 1 - \alpha$.

Definition 4. Let $\beta \in [0,1]$ and $u \in \{0,1\}^K$. A *β-noisy copy* of u is the random string $\boldsymbol{x}$ obtained from u by independently replacing each bit, with probability β, by a fresh uniform bit.

Claim 23. *Let $\boldsymbol{x}$ be a β-noisy copy of $\boldsymbol{u}$. Then for every $A \subseteq \{0,1\}^K$ with $p := \Pr[A]$,*

$$\Pr[\boldsymbol{u} \in A,\ \boldsymbol{x} \notin A] \geq \beta p(1-p).$$

Proof. Let g be the indicator of A, with Fourier coefficients $\widehat{g}(J)$ for $J \subseteq [K]$. Then $\widehat{g}(\varnothing) = p$ and $\sum_J \widehat{g}(J)^2 = p$, so

$$\Pr[\boldsymbol{u} \in A,\ \boldsymbol{x} \in A] = \sum_J (1-\beta)^{|J|} \widehat{g}(J)^2 \leq p^2 + (1-\beta)(p - p^2).$$

Subtracting this from p gives the claim. □

Claim 24. $\Gamma = O(q\varepsilon)$.

Proof. Let $\boldsymbol{x}$ be a β-noisy copy of $\boldsymbol{u}$ for $\beta := 1/(2q)$, and run D on $\boldsymbol{x}$. The pairs $(\boldsymbol{u}, \boldsymbol{x})$ and $(\boldsymbol{x}, \boldsymbol{u})$ have the same distribution. Fix $\tau \in \Lambda$, and let H_τ be the event that the noise replaces none of the at most q bits fixed by B_τ, so that $\Pr[H_\tau] \geq 1 - q\beta = 1/2$. Conditioned on H_τ and on $\boldsymbol{u} \in B_\tau$, we have $\boldsymbol{x} \in B_\tau$, and on the bits not fixed by B_τ the pair $(\boldsymbol{u}, \boldsymbol{x})$ is distributed as before. Applying Claim 23 inside the subcube B_τ to the set A_τ, with the roles of $\boldsymbol{u}$ and $\boldsymbol{x}$ exchanged, we get

$$\Pr[\boldsymbol{x} \in A_\tau,\ \boldsymbol{u} \in B_\tau \setminus A_\tau] \geq \Pr[B_\tau] \cdot \Pr[H_\tau] \cdot \beta\theta_\tau(1-\theta_\tau) \geq \frac{\beta}{2} \Pr[A_\tau](1-\theta_\tau).$$

The events on the left-hand side are disjoint for different τ, because the sets A_τ are. On each of them, the label $D(\boldsymbol{x}) = y_\tau$ is accepted on $\boldsymbol{u}$ and differs from $D(\boldsymbol{u})$. Therefore

$$\Pr\left[f \text{ accepts } D(\boldsymbol{x}) \text{ on } \boldsymbol{u}, \text{ and } D(\boldsymbol{x}) \neq D(\boldsymbol{u})\right] \geq \frac{\beta\Gamma}{2}.$$

Conditioned on which bits are replaced and on their new values, $D(\boldsymbol{x})$ is computed from $\boldsymbol{u}$ by a decision tree of depth at most t, so this probability is at most ε. Hence $\Gamma \leq 2\varepsilon/\beta = O(q\varepsilon)$. □

Proof of Claim 21. Let $\Lambda' := \{\tau \in \Lambda : \theta_\tau \geq 1/2\}$. If $m(u) = 0$, then either u lies in no set A_τ, or it lies in A_τ for some $\tau \notin \Lambda'$. Since $1 \leq 2(1-\theta_\tau)$ for $\tau \notin \Lambda'$,

$$\Pr[m(\boldsymbol{u}) = 0] \leq \alpha + \sum_{\tau \notin \Lambda'} \Pr[A_\tau] \leq \alpha + 2 \sum_{\tau \notin \Lambda'} \Pr[A_\tau](1-\theta_\tau).$$

Since every u lies in at most one of the sets A_τ, we have $\max\{m(u) - 1, 0\} \leq |\{\tau \in \Lambda' : u \in B_\tau \setminus A_\tau\}|$. Hence, using $\Pr[B_\tau] = \Pr[A_\tau]/\theta_\tau \leq 2\Pr[A_\tau]$ for $\tau \in \Lambda'$,

$$\mathbb{E} \max\{m(\boldsymbol{u}) - 1, 0\} \leq \sum_{\tau \in \Lambda'} \Pr[B_\tau](1-\theta_\tau) \leq 2 \sum_{\tau \in \Lambda'} \Pr[A_\tau](1-\theta_\tau).$$

Adding the two bounds gives $\mathbb{E}\,|m(\boldsymbol{u}) - 1| \leq \alpha + 2\Gamma$, which is $\alpha + O(q\varepsilon)$ by Claim 24. □

4.4 Proof of Claim 22

We prove Claim 22 with $c = 1/64$. Let

$$W(u) := \mathbf{1}[m(u) = 0] + m(u) \cdot \mathbf{1}[m(u) \geq 2].$$

Since $W \leq 2|m - 1|$ pointwise, we have $\mathbb{E}[W(\boldsymbol{u})] \leq 1/32$. The decision tree is given in Algorithm 3.

Algorithm 3 The decision tree of Claim 22.

Input: $u \in \{0,1\}^K$
Output: $\tau \in \Lambda'$ with $u \in B_\tau$, or fail
1: Initialize σ as the empty restriction.
2: **for** $k \in [8(q+1)]$ **do**
3: **if** $\mathbb{E}_\sigma[W(\boldsymbol{u})] > 1/8$ **then return** fail.
4: Set $\tau^* := \arg\min_\tau \Pr_\sigma[m(\boldsymbol{u}) \geq 2 \mid \boldsymbol{u} \in B_\tau]$, where τ ranges over Λ' with $\Pr_\sigma[B_\tau] > 0$.
5: Query all bits that are fixed by B_{τ^*} but not by σ, and record the answers in σ.
6: **if** $u \in B_{\tau^*}$ **then return** τ^*.
7: **return** fail.

Algorithm 3 makes at most $8(q+1) \cdot q = O(q^2)$ queries, and if it outputs τ then $u \in B_\tau$. There are two cases where it outputs fail:

1. Algorithm 3 halts in line 3.
2. Algorithm 3 halts in line 7.

We denote these two events by $\mathcal{E}_1$ and $\mathcal{E}_2$ respectively.

Claim 25. $\Pr[\mathcal{E}_1] \leq 1/4$.

Proof. The restrictions σ at which Algorithm 3 halts in line 3 have disjoint cubes and satisfy $\mathbb{E}_\sigma W > 1/8$. As $W \geq 0$,

$$\frac{1}{32} \geq \mathbb{E}\, W(\boldsymbol{u}) \geq \sum_\sigma \Pr[\mathsf{cube}(\sigma)] \cdot \mathbb{E}_\sigma W \geq \frac{1}{8} \Pr[\mathcal{E}_1],$$

where the sum is over these restrictions. □

Claim 26. *Suppose that an iteration of Algorithm 3 reaches line 4 at a restriction σ. Then τ^* exists, and $B^* := B_{\tau^*}$ satisfies $\Pr_\sigma[m \geq 2 \mid B^*] \leq 1/7$.*

Proof. Let $e := \mathbb{E}_\sigma W \leq 1/8$. Since $W \geq \mathbf{1}[m \neq 1]$, we have $\Pr_\sigma[m = 1] \geq 1 - e$. Hence

$$\sum_\tau \Pr_\sigma[B_\tau] = \mathbb{E}_\sigma[m] \geq 1 - e \qquad \text{and} \qquad \sum_\tau \Pr_\sigma[B_\tau] \cdot \Pr_\sigma[m \geq 2 \mid B_\tau] = \mathbb{E}_\sigma\left[m \cdot \mathbf{1}[m \geq 2]\right] \leq e,$$

where the second sum is over τ with $\Pr_\sigma[B_\tau] > 0$. So τ^* exists, and $\Pr_\sigma[m \geq 2 \mid B^*] \leq e/(1-e) \leq 1/7$. □

For u with $m(u) = 1$, let $B(u)$ be the unique member of the family that contains u. We say that an iteration of Algorithm 3 *makes progress on* u if $m(u) = 1$ and it either outputs τ^* or queries a bit that is fixed by $B(u)$. As $B(u)$ fixes at most q bits, at most $q+1$ iterations make progress on any given u.

Claim 27. *An iteration of Algorithm 3 that reaches line 4 at a restriction σ makes progress with probability at least $1/2$ over $\boldsymbol{u} \sim \mathsf{cube}(\sigma)$.*

Proof. Let $\mathcal{N}$ be the set of $u \in \mathsf{cube}(\sigma)$ such that $m(u) = 1$, the member $B(u)$ is not B^*, and no bit is fixed to different values by $B(u)$ and B^*. For $u \in \mathsf{cube}(\sigma)$, let x be obtained from u by overwriting the bits fixed by B^* with the values that B^* prescribes. If $\boldsymbol{u} \sim \mathsf{cube}(\sigma)$, then $\boldsymbol{x}$ is uniform over $\mathsf{cube}(\sigma) \cap B^*$. If $u \in \mathcal{N}$, then x lies in both B^* and $B(u)$, and so $m(x) \geq 2$. By Claim 26,

$$\Pr_\sigma[\mathcal{N}] \leq \Pr_\sigma[m(\boldsymbol{x}) \geq 2] = \Pr_\sigma[m \geq 2 \mid B^*] \leq \frac{1}{7}.$$

Now let $u \in \mathsf{cube}(\sigma) \setminus \mathcal{N}$ with $m(u) = 1$. If $B(u) = B^*$, then the iteration outputs τ^*. Otherwise some bit is fixed to different values by $B(u)$ and B^*. This bit is not fixed by σ, because both $B(u)$ and B^* intersect $\mathsf{cube}(\sigma)$, so the iteration queries it. Since $\mathbb{E}_\sigma W \leq 1/8$ and $W \geq \mathbf{1}[m \neq 1]$, the iteration makes progress with probability at least $\Pr_\sigma[m = 1] - \Pr_\sigma[\mathcal{N}] \geq 7/8 - 1/7 > 1/2$. □

Claim 28. $\Pr[\mathcal{E}_2] \leq 1/4$.

Proof. For $k \le 8(q+1)$, let $\mathcal{R}_k$ be the event that the k-th iteration reaches line 4, and let $\mathcal{P}_k \subseteq \mathcal{R}_k$ be the event that it makes progress. Then $\Pr[\mathcal{P}_k] \ge \Pr[\mathcal{R}_k]/2$ by Claim 27, and $\sum_k \Pr[\mathcal{P}_k] \le q+1$. If $\mathcal{E}_2$ occurs, then all the events $\mathcal{R}_k$ occur. Hence

$$8(q+1) \cdot \Pr[\mathcal{E}_2] \le \sum_k \Pr[\mathcal{R}_k] \le 2 \sum_k \Pr[\mathcal{P}_k] \le 2(q+1). \qquad \square$$

Given Claims 25 and 28, Algorithm 3 outputs fail with probability at most 1/2, which proves Claim 22.

5 PLS separations from proof complexity

In this section we prove separations between TFNP classes in the random oracle model. We will work with the following notion of a ROM reduction.

Public reduction. Let $F = \{F_n\}, R = \{R_m\}$ be TFNP problems, where $F_n \subseteq \Sigma^n \times \mathcal{O}$ and $R_m \subseteq \Sigma^m \times \mathcal{O}'$. A *public (ROM) reduction* from F_n to R_m is given by a family of decision trees (ω, f, g) satisfying the following properties.

1. The first family of decision trees, $\omega = \{\omega_1, \dots, \omega_n\}$ query variables $u_1, \dots, u_t$, and together they map a random string $\boldsymbol{u} \in \Sigma^t$ to an instance of F_n. That is, for each random string u, $\omega(\boldsymbol{u}) = \omega_1(\boldsymbol{u}) \dots \omega_n(\boldsymbol{u}) \in \Sigma^n$ defines an input to F_n, and thus ω defines a "locally-samplable" distribution over instances of F_n.
2. The second family of decision trees, $f = \{f_1, \dots, f_m\}$ query $\boldsymbol{u}$ and together they define a reduction from an instance of F_n to an instance of R_m.
3. The third family of decision trees $g = \{g_o \mid o \in \mathcal{O}'\}$ queries $\boldsymbol{u}$, and if o is a legal solution to the instance $f(\boldsymbol{u})$ of R_m, the decision tree $g_o(\boldsymbol{u})$ outputs a candidate solution to F_n on instance $\omega(\boldsymbol{u})$.

The *depth* $d(\omega, f, g)$ of a reduction (f, g) is the maximum depth of any of the decision trees, and the *complexity* is $c(\omega, f, g) \coloneqq d(\omega, f, g) + \log m$. The *error* of the reduction is

$$\operatorname{err}(\omega, f, g) \coloneqq \Pr_{\boldsymbol{u} \sim \Sigma^t} \Big[\exists o \in \mathcal{O}' : (f(\boldsymbol{u}), o) \in R_m \wedge (\omega(\boldsymbol{u}), g_o(\boldsymbol{u})) \notin F_n \Big].$$

We say that F *public reduces* to R if for every sufficiently large n there is a polylog(n)-complexity reduction for *every* locally-samplable distribution. That is, for every polylog(n)-depth ω there exists f, g such that $c(\omega, f, g) = \text{polylog}(n)$ and $\operatorname{err}(\omega, f, g) \le 1/3$.

Note that up to a blow-up of $d(\omega, f, g)$ this is equivalent to allowing f and g to depend on $\omega(\boldsymbol{u})$ as well, since each query to a bit of $\omega(\boldsymbol{u})$ can be simulated by querying the values of $\boldsymbol{u}$ that ω queries for the value of that bit. As well, note that if $\operatorname{err}(\omega, f, g) = 0$, then we recover the standard definition of a black-box reduction between TFNP problems.

5.1 Separations in the ROM from average-case proof complexity bounds

Our TFNP separations will proceed by a reduction to proof complexity lower bounds. Proof complexity and TFNP are intimately connected in the black-box model. Characterization theorems show that black-box TFNP subclasses correspond exactly to the search problems whose totality can be proven efficiently in a corresponding proof system [GKRS19, BFI23]. For example, recalling Definition 1 of a resolution refutation, the following result shows that resolution is exactly the search problems whose totality can be proven by PLS-style arguments.

Resolution = PLS ([BKT14, Kam19]). For any $F \in \mathsf{TFNP}^{\mathrm{dt}}$, $F \in \mathsf{PLS}^{\mathrm{dt}}$ iff there is a width-poly$(\log n)$ pair $(\mathcal{F}, S)$ that solves F. Quantitatively, a zero-error reduction of complexity t to the canonical PLS problem yields a resolution refutation of width $O(t + v)$, where v bounds the source-verifier depth; see [Kam19, Theorem 8.18(1)]. We use the same characterization for alphabet-valued decision trees, replacing Boolean literals by cell equalities $[u_i = a]$.

In this section, we extend this connection to ROM reductions. Let (ω, f, g) be a public reduction from a TFNP problem $F \subseteq \Sigma^n \times \mathcal{O}$ to a TFNP problem $R \subseteq \Sigma^m \times \mathcal{O}'$. All our lower bounds will be against the uniform distribution $\omega(u) = u$, and henceforth we restrict attention to this case and omit ω. The main difference between standard black-box reductions and ROM reductions is that the latter allows for error. We will capture this error by a low-width DNF B whose probability of being satisfied over the uniform distribution is $\mathrm{err}(\omega, f, g)$. A width-$w$ *DNF* over an alphabet Σ is a disjunction of predicates $B = \bigvee_{i\in[t]} B_i$ such that each B_i is a term depending on at most w variables – there is a single assignment to the variables of B_i satisfying it. This will give us a new search problem which we denote by $F \vee B$: on a given input, find either a solution to F or a satisfied term of B.

We now define B. For each target answer $o \in \mathcal{O}'$, let H_o be the decision tree that first verifies $(f(u), o) \in R$, then runs the decoder g_o, and finally checks that $(u, g_o(u)) \notin F$. Let $\mathrm{viol}(f, g)$ be the set of accepting root-to-leaf transcripts of these trees, viewed as partial assignments to u. Define the violation DNF for the reduction as

$$B_{f,g} \coloneqq \bigvee_{\rho\in\mathrm{viol}(f,g)} \bigwedge_{i\in\mathsf{fix}(\rho)} [u_i = \rho_i].$$

Thus $B_{f,g}(u) = 1$ exactly when some valid target answer decodes to an invalid source answer. If the reduction has depth d and the target and source verifiers have depths v_R and v_F, respectively, each term has width at most $dv_R + d + v_F$. $B_{f,g}$ captures the allowable error of the public reduction, and will be given to the proof system as "advice" to help prove F. The key property of $B_{f,g}$ imposed by the reduction is that

$$\Pr_{\boldsymbol{u}\sim\Sigma^n}\left[B_{f,g}(\boldsymbol{u}) = 1\right] = \mathrm{err}(\omega, f, g).$$

Therefore, if the reduction has low error, then $B_{f,g}$ is almost never satisfied and intuitively cannot be too helpful in proving F. Abusing notation, let $F \vee B \subseteq \Sigma^n \times (\mathcal{O} \cup \{\rho \mid \rho \in B\})$ be the search problem of returning a solution to F or a satisfied term of B.

Lemma 29. *Let $F \subseteq \Sigma^n \times \mathcal{O}$ have solution verifiers of depth at most w. Any public reduction (id, f, g) from F to the canonical PLS problem of complexity at most w yields a DNF B of width $O(w)$ such that*

$$\Pr_{\boldsymbol{u}\sim\Sigma^n}[B(\boldsymbol{u}) = 1] = \mathrm{err}(\mathrm{id}, f, g),$$

and $F \vee B$ has a resolution refutation of width $O(w)$. More precisely, if the reduction has depth d and the source verifiers have depth v_F, the DNF B has width at most $3d + v_F$.

In particular, if $F \in \mathsf{TFNP}^{\mathrm{dt}}$ and $F \vee B$ has no polylog$n$-width resolution refutation for every width-polylog$n$ DNF B satisfying $\Pr[B(\boldsymbol{u}) = 1] \le 1/3$, then there is no public reduction from F to PLS. This is a lower bound on the randomized resolution width of F from [BKT14] specialized to the uniform distribution.

Proof. For each target answer $o \in \mathcal{O}'$, the tree H_o defined above accepts exactly when o is valid for $f(u)$ but $g_o(u)$ is invalid for F. Its target-verification stage uses at most $2d$ source queries, its decoder uses at most d, and its source verifier uses at most v_F. Taking the disjunction of all accepting transcripts gives $B = B_{f,g}$ of width at most $3d + v_F \le 4w$, with satisfaction probability exactly $\mathrm{err}(\mathrm{id}, f, g)$.

Modify the decoder to run the three stages of H_o. If the target check fails, return an arbitrary value. Otherwise, return $g_o(u)$ when the source verifier accepts, or the accepting transcript of H_o, a satisfied term of B, when it rejects. This is a zero-error reduction from $F \vee B$ to the same PLS instance, of complexity $O(w)$. Solutions to $F \vee B$ are verified in depth $O(w)$, so the quantitative PLS–resolution characterization [Kam19, Theorem 8.18(1)] supplies the width-$O(w)$ refutation. □

5.2 Robust strategies and average-case resolution

In this section, we develop tools for proving average case resolution lower bounds – that is, $F \vee B$ for every B satisfying the conditions of Lemma 29. We begin by describing the dual object (known as a *Delayer strategy*) for resolution proofs, and later extend it to the average case. For a partial assignment $\rho \in (\Sigma \cup \{\star\})^m$, denote by $\mathsf{fix}(\rho) \coloneqq \{i \in [m] \mid \rho_i \neq \star\}$, and let $\mathsf{free}(\rho) \coloneqq [m] \setminus \mathsf{fix}(\rho)$. As well, let $\mathsf{cube}(\rho) \coloneqq \{x \in \Sigma^m \mid \forall i \notin \mathsf{free}(\rho), x_i = \rho_i\}$ be the sub-cube of total assignments which agree with ρ.

Strategy. Let $F \subseteq \Sigma^m \times \mathcal{O}$. A width-$w$ *strategy* for F is a non-empty downward-closed set $\mathcal{F} \subseteq (\Sigma \cup \{\star\})^m$ such that for every $\rho \in \mathcal{F}$ with $|\mathsf{fix}(\rho)| \leq w$ the following hold:

- For every $o \in \mathcal{O}$, there is an $x \in \mathsf{cube}(\rho)$ such that $(x, o) \notin F$
- *Extension Property:* If $|\mathsf{fix}(\rho)| < w$ and $i \in \mathsf{free}(\rho)$ there is $a \in \Sigma$ such that $\rho \cup \{x_i = a\} \in \mathcal{F}$.

This is the dual object for resolution proofs:

Lemma 30 ([AD08])**.** *There is a width-w resolution refutation of F iff there is no width-w strategy.*

Hence, to separate F from PLS it suffices to show that there is a polylog$n$-width strategy for $F \vee B$ for all suitable B. We show that this is implied by constructing a suitably *robust* strategy for F.

Robust strategy. A width-w strategy for F is k-robust if it also satisfies:

- *Robust Extension Property:* For every $\rho \in \mathcal{F}$ with $|\mathsf{fix}(\rho)| < w$ and every $i \in [m] \setminus \mathsf{fix}(\rho)$ there exist at least $|\Sigma| - k$ values $a \in \Sigma$ such that $\rho \cup \{x_i = a\} \in \mathcal{F}$.

Lemma 31. *Suppose that $F \subseteq \Sigma^m \times \mathcal{O}$ has a width-w k-robust strategy. Then for every width-w DNF B such that $\Pr_{\boldsymbol{u} \sim \Sigma^m}[B(\boldsymbol{u}) = 1] < (1 - k/|\Sigma|)^w$ there is an (ordinary) strategy of width w for $F \vee B$.*

The remainder of this section is devoted to proving this lemma. Let us identify $\Sigma = [n]$ for simplicity. We aim to extract an ordinary strategy for $F \vee B$ from a k-robust strategy for F. The naive approach would be to take the k-robust strategy $\mathcal{F}$ for F and keep only $\rho \in \mathcal{F}$ such that

$$\mathsf{cube}(\rho) \not\subseteq B^{-1}(1). \tag{2}$$

Then no $\rho \in \mathcal{F}$ that meets (2) can falsify any clause of F or B. The issue with such a naive restriction is with the extension property. Let $x_1, \ldots, x_m$ be the variables of F. Suppose ρ satisfies (2) and $n - k$ extensions of the form $\rho \cup \{x_i = j\}$ are in $\mathcal{F}$. If we show that at least one of the extensions also satisfies (2), then this proves the extension property. If $k = 0$, this would be trivial:

$$\mathsf{cube}(\rho) \setminus B^{-1}(1) = \bigsqcup_{j \in [n]} (\mathsf{cube}(\rho \cup \{x_i = j\}) \setminus B^{-1}(1)), \tag{3}$$

so if the left-hand side is non-empty, then at least one of the terms on the right-hand is non-empty. Hence, ρ satisfies the extension property. This becomes false even for $k = 1$, as the difference on the left-hand side may be completely contained in one of the parts of the right-hand side.

We resolve this by using the linear algebraic embedding of Razborov [Raz03] developed for the pseudo-width method. The high-level idea is to forget about which holes are occupied by pigeons under ρ and instead only track the number of holes that are occupied. To do so, we associate each pigeon $i \in [m]$ with a vector space $V_i = \mathbb{R}^{n-k}$ and choose vectors $v_{i,1}, \ldots, v_{i,n}$ such that every $n - k$ of them span the space—this will have the effect of only counting the number of occupied holes. Random vectors or Vandermonde vectors have this property. Then we set $V \coloneqq \bigotimes_{i \in [m]} V_i$ and define $L \colon [n]^m \to V$ as mapping $x \in \Sigma^m$ to $\otimes_{i \in [m]} v_{i,x_i}$. For a set $A \subseteq [n]^m$ extend $L(A) \coloneqq \mathrm{span}\{L(x) \mid x \in A\}$ and shorthand $L(\rho) \coloneqq L(\mathsf{cube}(\rho))$. We then claim that

$$\mathcal{F}' \coloneqq \{\rho \in \mathcal{F} \mid L(\rho) \not\subseteq L(B^{-1}(1))\}$$

is a width-w strategy for $F \vee B$. Indeed, the key property of this embedding is that now if we want to extend ρ by mapping x_i then all we need to know is that there are $n-k$-many values in Σ where x_i can be mapped to. This is formalized in the next lemma as the following "saturated counting" version of (3), stating that *any* $n - k$ extensions of ρ are sufficient to span $L(\rho)$.

Lemma 32. *For every $M \in \binom{\Sigma}{n-k}$ and $i \notin \mathsf{fix}(\rho)$ we have $L(\rho) = \sum_{j \in M} L(\rho \cup \{x_i = j\})$.*

Proof. By definition we have:

$$\begin{aligned} L(\rho) &= \bigotimes_{t \in \mathsf{fix}(\rho)} \mathrm{span}(v_{t,\rho(t)}) \otimes V_i \otimes \bigotimes_{\ell \in [m] \setminus (\mathsf{fix}(\rho) \cup \{i\})} V_\ell \\ &= \bigotimes_{t \in \mathsf{fix}(\rho)} \mathrm{span}(v_{t,\rho(t)}) \otimes \mathrm{span}\{v_{i,j} \mid j \in M\} \otimes \bigotimes_{\ell \in [m] \setminus (\mathsf{fix}(\rho) \cup \{i\})} V_\ell. \end{aligned}$$

Then by the property of tensor products we can rewrite

$$L(\rho) = \sum_{j\in M} \bigotimes_{t\in \mathsf{fix}(\rho)} \operatorname{span}(v_{t,\rho(t)}) \otimes \operatorname{span}(v_{i,j}) \otimes \bigotimes_{\ell\in[m]\setminus(\mathsf{fix}(\rho)\cup\{i\})} V_\ell = \sum_{j\in M} L(\rho \cup \{x_i = j\}). \qquad \square$$

It follows that $\mathcal{F}'$ satisfies the extension property: suppose that $\rho \in \mathcal{F}'$, so $L(\rho) \not\subseteq L(B^{-1}(1))$, $i \in [m] \setminus \mathsf{fix}(\rho)$. Since $\rho \in \mathcal{F}$ by the k-robustness of $\mathcal{F}$ we have that the size of $M = \{j \in [n] \mid \rho \cup \{x_i = j\} \in \mathcal{F}\}$ is at least $n-k$. If for all $j \in M$ we have $L(\rho\cup\{x_i = j\}) \subseteq L(B^{-1}(1))$, then by Lemma 32 $\sum_{j\in M} L(\rho\cup\{x_i = j\}) = L(\rho)$, and since $L(B^{-1}(1))$ is a subspace, we have $L(\rho) \subseteq L(B^{-1}(1))$, a contradiction.

The remaining issue is the non-triviality of $\mathcal{F}'$; indeed, the extension property is satisfied by the empty strategy. Thus, we need to show that $\star^m \in \mathcal{F}'$, equivalently $V = L(\star^m) \not\subseteq L(B^{-1}(1))$. We show that for simple small sets A, $\dim(L(A))$ is also small:

Lemma 33. *For every w-DNF B we have*

$$\frac{\dim(L(B^{-1}(1)))}{\dim(V)} \le (1-k/n)^{-w} \cdot \Pr_{\boldsymbol{u}\sim[n]^m}[B(\boldsymbol{u}) = 1].$$

Given Lemma 33, we can conclude the proof of Lemma 31: $\Pr[B(\boldsymbol{u}) = 1] < (1-k/n)^w$, so $\dim(L(B^{-1}(1))) < \dim(V)$, and $\mathcal{F}' \ni \star^m$.

Proof of Lemma 33. Let $B_1, \dots, B_h$ be the terms of B and let $B_{\le i} = \bigvee_{j\le i} B_j$. By induction on i, we will bound how much the dimension can increase from $B_{\le i-1}$ to $B_{\le i}$. Denote $Z_i \coloneqq B_i^{-1}(1)$ and $Z_{\le i} \coloneqq \bigcup_{j\le i} Z_j$, $Z \coloneqq B^{-1}(1)$. We will prove that the following holds:

$$\dim(L(Z_{\le i})) - \dim(L(Z_{<i})) \le \left(\frac{n-k}{n}\right)^{m-w} \left|Z_i \setminus Z_{<i}\right|. \tag{4}$$

We postpone the proof of this inequality and first complete the proof of the lemma. Summing over all i,

$$\dim(L(Z)) \le \left(\frac{n-k}{n}\right)^{m-w} \left(\sum_i |Z_i \setminus Z_{<i}|\right) = \left(\frac{n-k}{n}\right)^{m-w} |Z|.$$

Hence,

$$\frac{\dim(L(Z))}{\dim(V)} \le \frac{((n-k)/n)^{m-w}}{(n-k)^m}|Z| = \left(\frac{n-k}{n}\right)^{-w}|Z|n^{-m} = (1-k/n)^{-w} \cdot \Pr_{\boldsymbol{u}\sim[n]^m}[B(\boldsymbol{u}) = 1].$$

To prove Equation 4 we will use the following claim which bounds the dimension of a set as a function of its size.

Claim 34. *For any partial assignment $\rho\colon [m] \to [n] \cup \{\star\}$ and any $A \subseteq \mathsf{cube}(\rho)$,*

$$\dim(L(A)) \ge \left(\frac{n-k}{n}\right)^{m-|\mathsf{fix}(\rho)|}|A|.$$

In particular, $\dim(L(Z_i)) \ge ((n-k)/n)^{m-w}$.

Proof. First, we prove that for any $A \subseteq [n]^m$, $\dim(L(A)) \ge |A|((n-k)/n)^m$. For each $i \in [m]$ choose $\boldsymbol{B}_i \sim \binom{[n]}{n-k}$ and note that by our embedding, the vectors in $L(\boldsymbol{B}_i)$ are all linearly independent. Let $\boldsymbol{B} = \prod_{i\in[m]} \boldsymbol{B}_i$. Then,

$$\dim(L(A \cap \boldsymbol{B})) = |A \cap \boldsymbol{B}|,$$

by linear independence. That is, $\boldsymbol{B}_i$ is restricting the i-th variable to have a value only in $\boldsymbol{B}_i$. Hence

$$\dim(L(A)) \ge \mathbb{E}[\dim(L(A \cap \boldsymbol{B}))] = \mathbb{E}[|A \cap \boldsymbol{B}|] = |A|\left(\frac{n-k}{n}\right)^m,$$

where the final equality is the probability that each element in $\boldsymbol{B}$ is in A. By the same argument, for any partial restriction ρ and $A \subseteq \mathsf{cube}(\rho)$,

$$\dim(L(A)) \ge \left(\frac{n-k}{n}\right)^{m-|\mathsf{fix}(\rho)|}|A|. \qquad \square$$

We now prove Equation 4. First note that without loss of generality we can assume every term in B has width exactly w: a term B_i of width t can be transformed into an equivalent DNF of width $t+1$ by taking a variable x_j not mentioned in B_i and defining $\bigvee_{a\in[n]} B_i \wedge [\![x_j = a]\!]$. With this simplification,

$$\begin{aligned}
\left(\frac{n-k}{n}\right)^{m-w} |Z_i \setminus Z_{<i}| &= \left(\frac{n-k}{n}\right)^{m-w} (|Z_i| - |Z_i \cap Z_{<i}|) \\
&= \left(\frac{n-k}{n}\right)^{m-w} (n^{m-w} - |Z_i \cap Z_{<i}|) \\
&\geq (n-k)^{m-w} - \dim(L(Z_i \cap Z_{<i})) && \text{(Applying Claim 34)} \\
&= \dim(L(Z_i)) - \dim(L(Z_i \cap Z_{<i})) \\
&\geq \dim(L(Z_{\leq i})) - \dim(L(Z_{<i})),
\end{aligned}$$

where the final inequality follows by applying the next Lemma 35 with $B = Z_i \cap Z_{<i}$, $A = Z_i \setminus Z_{<i}$, and $C = Z_{<i} \setminus Z_i$. □

Lemma 35. *For any disjoint sets A, B, C,*

$$\dim(L(A \cup B)) - \dim(L(B)) \geq \dim(L(A \cup B \cup C)) - \dim(L(B \cup C)).$$

Proof. Let A, B, C be disjoint sets. Then,

$$\begin{aligned}
\dim(L(A \cup B)) - \dim(L(B)) &= \dim(L(A)) - \dim(L(A) \cap L(B)) \\
&\geq \dim(L(A)) - \dim(L(A) \cap L(B \cup C)) \\
&= \dim(L(A \cup B \cup C)) - \dim(L(B \cup C)),
\end{aligned}$$

where the final line follows by adding $B \cup C$ to both sides. □

5.3 PWPP is not in PLS

In this section we show that PWPP is not contained within PLS under public reductions.

Theorem 36. *Let (ω, f, g), with $\omega(\boldsymbol{u}) = \boldsymbol{u}$, be any public reduction of complexity $w \leq n^{1/3}$ from* Pigeon *on $[n]^{2n}$ to the canonical PLS-complete problem of Section A. Then,* $\mathrm{err}(\omega, f, g) = 1 - n^{-\Omega(1)}$.

Consider the problem Pigeon $\subseteq [n]^m \times \binom{[m]}{2}$, for $m > n$, of finding a collision $x_i = x_j$ in a given $x \in [n]^m$. For $m = 2n$ this is PWPP-complete. Let (ω, f, g), with $\omega(\boldsymbol{u}) = \boldsymbol{u}$, be a public reduction from Pigeon to the canonical PLS problem, of depth d and complexity $w = d + \log M$, where $M \geq 4$ is the number of target pointers. A collision is verified with two source queries. Thus Lemma 29 gives a DNF B of width at most $3d + 2 \leq 3w$ and a resolution refutation of Pigeon $\vee B$ of width $O(w)$, such that $\Pr_{\boldsymbol{u}\sim[n]^m}[B(\boldsymbol{u}) = 1] = \mathrm{err}(\omega, f, g)$. Choose an absolute constant C such that both widths are at most $W = \lceil Cw \rceil$. The following lemma forces

$$\mathrm{err}(\omega, f, g) \geq (1 - W/n)^W \geq 1 - W^2/n = 1 - O(w^2/n),$$

which proves Theorem 36 for $w \leq n^{1/3}$ and sufficiently large n.

Lemma 37. *If B is a width-w DNF such that $\Pr_{\boldsymbol{u}\sim[n]^m}[B(\boldsymbol{u}) = 1] < (1 - w/n)^w$ then there is a width-w strategy for* Pigeon $\vee B$.

Proof. By Lemma 31 it suffices to show that Pigeon has a w-robust strategy of width w. Define $\mathcal{F}$ to be the set of all partial injections of width at most w. $\mathcal{F}$ is downward-closed by definition and no injection falsifies Pigeon, so it remains to show the query property. Take $\rho \in \mathcal{F}$ and arbitrary $i \in \mathsf{free}(\rho)$. For every $j \in [n] \setminus \mathsf{range}(\rho)$ (there are $n - w$ of those) the partial assignment $\rho \cup \{p_i = j\}$ is a partial injection, so if $|\mathsf{fix}(\rho)| < w$, $\rho \cup \{p_i = j\} \in \mathcal{F}$ as required. □

As a consequence, we can also obtain average-case resolution *size* lower bounds. In terms of Definition 1 the size is $|\mathcal{F}|$. Notice that we can relax Definition 1 so either ρ fixes a solution or $i(\rho)$ is defined, regardless of the width of ρ. Then the terminal elements of $\mathcal{F}$ do not need to have the same width. The following corollary's proof is a variant of [PT19, Proposition 3.7].

Corollary 38. *For any DNF B satisfying $\Pr_{\boldsymbol{u}\sim[n]^m}[B(\boldsymbol{u})=1] < (1-w/n)^w - 1/n$, any resolution refutation of $\textsc{Pigeon} \vee B$ has size at least n^w.*

Proof. Suppose $\mathcal{F}$ is the (arbitrary width) refutation of $\textsc{Pigeon} \vee B$ of size $< n^w$. Remove all elements of $\mathcal{F}$ of width $> w$ from $\mathcal{F}$ and form a DNF B'. Then $\Pr[B'(\boldsymbol{u})=1] < n^w \cdot n^{-(w+1)} = 1/n$, so there is a width-$(w+1)$ resolution refutation of $\textsc{Pigeon} \vee B \vee B'$, which contradicts Lemma 37. □

5.4 PPA is not in PLS

The preceding PPAD $\notin$ FP separation already implies PPA $\notin$ FP. Here we strengthen the conclusion: we construct a PPA search problem that has no low-depth average-case reduction to PLS.

The problem is again to find an edge in the hypergraph $G = ([N], E)$ supplied by Theorem 10: given a labeling $h\colon [N] \to \{0,1\}^r$ find $e \in E$ such that $\oplus_{v\in e} h(v) = 0^r$. We again plant a special edge $e^* = [\ell]$ and set $h(1) = \cdots = h(\ell) = 0^r$, denoting $\mathcal{H}$ the set of such functions, and forbid e^* as a solution:

$$\textsc{ZeroEdge}_G := \{(h, e) \in \mathcal{H} \times (E \setminus \{e^*\});\ \oplus_{v\in e} h(v) = 0^r\}.$$

The reason for this change is in order to make $\textsc{ZeroEdge}_G$ hard even for PLS.

Theorem 39. *There exists $G = ([N], E)$ such that $\textsc{ZeroEdge}_G \in$ PPA and there is no public reduction from $\textsc{ZeroEdge}_G$ to PLS of reduction complexity $\exp((\log N)^{o(1)})$.*

The theorem immediately follows from Lemmas 40 and 41 below:

Lemma 40. *Suppose $G = ([N], E)$ and $E \subseteq \binom{[N]}{\ell}$ is $(\ell, w+\ell, b)$-well-spread. Then there exists a b-robust width-w strategy for $\textsc{ZeroEdge}_G$.*

Lemma 41. *If $G = ([N], E)$ with $E \subseteq \binom{[N]}{\ell}$ is explicit and efficiently r-balanced, with $\ell = \mathrm{poly}(\log N)$, then $\textsc{ZeroEdge}_G \in$ PPA.*

Proof of Theorem 39. The graph supplied by Theorem 10 satisfies the preconditions of Lemmas 40 and 41 simultaneously. Then by Lemma 31 and the robust strategy given by Lemma 40 we get that for every DNF B of width $w = \exp((\log N)^{o(1)})$ and $\Pr[B(\boldsymbol{u})=1] \le 1/3$ there is no width-w refutation for $\textsc{ZeroEdge}_G \vee B$. Hence by Lemma 29, no public reduction from $\textsc{ZeroEdge}_G$ to PLS exists. □

PLS hardness: proof of Lemma 40

Let $\mathcal{L}$ be the set of all partial assignments $\rho\colon [N] \setminus [\ell] \to \{0,1\}^r \cup \{\star\}$. We say that $\rho \in \mathcal{L}$ *reveals an edge* $e \in E$ if $\rho(v) \neq \star$ for all $v \in e \setminus [\ell]$ and $\oplus_{v\in e\setminus[\ell]} \rho(v) = 0$. Then we simply define

$$\mathcal{F} := \{\rho \in \mathcal{L} \mid |\mathsf{fix}(\rho)| \le w \text{ and } \rho \text{ does not reveal any edge } e \in E \setminus \{e^*\}\}.$$

Take any $\rho \in \mathcal{F}$ of width less than w and any $v \in ([N] \setminus [\ell]) \setminus \mathsf{fix}(\rho)$. By well-spreadness we get that at most b edges $F \subseteq E$ intersect $\mathsf{fix}(\rho) \cup [\ell] \cup \{v\}$ in at least ℓ nodes. By the fact that ρ does not reveal any edge, the sum of labels in all edges that do not contain v is non-zero. For each edge $e \in F$ such that $v \in e$ let $f_e := \oplus_{z\in e\setminus[\ell]\setminus\{v\}} \rho(z)$ be the value of $h(v)$ that makes e a solution to $\textsc{ZeroEdge}_G$. Then for $\alpha \in \{0,1\}^r \setminus \{f_e \mid e \in F; v \in e\}$ the assignment $\rho \cup \{h(v) = \alpha\} \in \mathcal{F}$, so since $|F| \le b$ the extension property of a b-robust strategy is satisfied.

PPA containment: proof of Lemma 41

We follow exactly the same strategy as in the PPAD case: encode ZeroEdge as a system of polynomials which is unsatisfiable over the Boolean cube and show that it has a short $\mathbb{F}_2$-Nullstellensatz proof, which implies that it lies within PPA by [GKRS19]. For every $a \in [N] \setminus [\ell]$, and $j \in [r]$, introduce one Boolean variable $h_{a,j}$. Write

$$h(a) = (h_{a,1}, \ldots, h_{a,r}) \in \{0,1\}^r.$$

Set $h_{a,j} \coloneqq 0$ for all $a \in [\ell]$ and $j \in [r]$; these are constants, not variables. Thus $h(1) = \cdots = h(\ell) = 0^r$. Let $h = \{h_{a,j} \mid a \in [N] \setminus [\ell],\ j \in [r]\}$. Then the number of Boolean variables in h is $n \coloneqq (N-\ell)r$.

For an edge $e \in E$, define

$$F_e(h) = \sum_{v \in e} h(v) \in \mathbb{F}_2^r, \qquad F_e(h)_j = \sum_{v \in e} h_{v,j}.$$

where the first sum is coordinate-wise XOR and the second sum is over $\mathbb{F}_2$. Define

$$\mathcal{P}_G = \{p_e(h) \mid e \in E \setminus \{e^*\}\}, \qquad p_e(h) = \prod_{j=1}^{r} \left(1 + \sum_{v \in e} h_{v,j}\right). \tag{5}$$

Every polynomial is over $\mathbb{F}_2$. The set of polynomials in $\mathcal{P}_G$ will be our system of (unsatisfiable) polynomial equations. On a Boolean assignment,

$$p_e(h) = 1 \quad \Longleftrightarrow \quad F_e(h) = 0^r.$$

The next lemma shows that if E is r-balanced, then the system of polynomial equations ($\mathcal{P}_G$) is unsatisfiable, and therefore ZeroEdge$_G$ is total. Secondly, we exhibit a simple NS_2 refutation for $\mathcal{P}_G$, which by [GKRS19] implies a *black-box* containment ZeroEdge$_G \in \mathsf{PPA}^{\mathrm{dt}}$.

Lemma 42 (NS_2 refutation of $\mathcal{P}_G$)**.** *The system $\mathcal{P}_G$ is unsatisfiable and has the NS_2 refutation*

$$1 = \sum_{e \in E \setminus \{e^*\}} p_e(h) \mod 2.$$

Furthermore, the NS_2 certificate, $\sum_{e \in E \setminus \{e^*\}} p_e$ has degree r and at most $|E|(\ell+1)^r$ monomial occurrences before cancellation.

Proof. Let $p_{e^*}(h)$ be defined as above (although it isn't one of the equations in $\mathcal{P}_G$); that is, $p_{e^*}(h) = \prod_{j \in [r]}(1 + F_{e^*}(h)_j)$. We first show that by the balancedness of E we have: $\sum_{e \in E} p_e(h) = 0$ over $\mathbb{F}_2$. To see this first note that when we expand $p_e(h)$ we get the sum of terms of the form

$$h_{a_1,j_1} h_{a_2,j_2} \ldots h_{a_s,j_s},$$

where $s \le r$, all j_k's are distinct, and $a_1, \ldots, a_s \in e$. Now, summing over all e and expanding each $p_e(h)$ we want to show that every term appears an even number of times in $\sum_{e \in E} p_e(h)$. Each term appears exactly once in each $p_e(h)$ where e contains the set of nodes mentioned in it. By the r-balancedness the number of edges that contain a fixed subset of size at most r is even.

On the other hand, since $h(1) = \cdots = h(\ell) = 0$, the polynomial $p_{e^*}(h) = p_{[\ell]}(h)$ is equal to 1:

$$p_{e^*}(h) = \prod_{j \in [r]} (1 + F_{e^*}(h)_j) = \prod_{j \in [r]} (1 + \sum_{i \in [\ell]} h(i)_j) = \prod_{j \in [r]} (1 + 0) = 1.$$

Therefore since $p_{e^*}(h) = 1$ and $\sum_{e \in E} p_e(h) = 0$, it follows that $\sum_{e \in E \setminus \{e^*\}} p_e(h) = 1$ over $\mathbb{F}_2$. □

We have shown not only that $\mathcal{P}_G$ is unsatisfiable, but that there is a simple NS_2 refutation, $\sum_e p_e q_e = 1$, (where the q_e's are all 1) obtained by simply taking the sum of all polynomials in $\mathcal{P}_G$. Lastly, each nonconstant p_e has degree r, and there are at most $(\ell+1)^r$ terms in each p_e. Therefore, the proof size (number of monomials before cancellation) is at most $|E|(\ell+1)^r = 2^{\mathrm{poly}(\ell)}$.

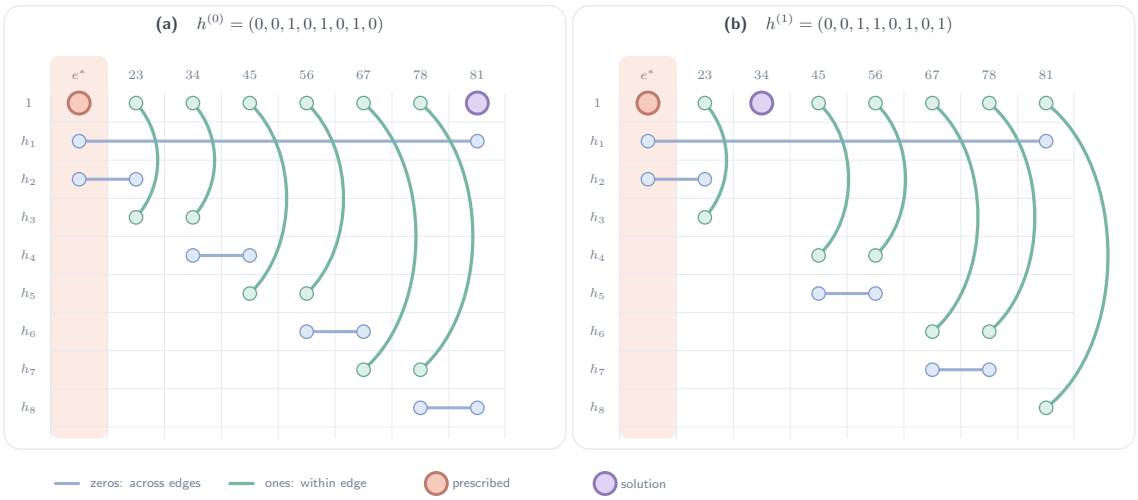

Figure 5: This picture illustrates the translation with $r = 1$ and G being the ordinary 8-node cycle with $e^* = \{1, 2\}$. The problem ZeroEdge$_G$ in this case is given a boolean labeling $h \in \{0,1\}^8$ with $h(1) = h(2) = 0$ to find an edge with equal nodes labels. One can view this instance of ZeroEdge$_G$ as a Tseitin formula for the dual of G, so the NS_2-refutation is simply $\sum_{uv \in E}(1 + h_u + h_v) = 0$. Each cell of the table corresponds to some pair edge-monomial (monomials have degree ≤ 1 here), but only the cells (e, u) with $u \in \{\bot\} \cup e$ are nodes of the PPA instance.

Translating from NS_2 to PPA. To finish the proof of PPA inclusion, we instantiate the conversion of a $\mathbb{F}_2$-Nullstellensatz refutation to a PPA instance by [GKRS19, Lemma 13]. Their generic proof does not necessarily imply an efficient instance. We use the fact that G is *efficiently* r-balanced. Recall that it means that for every set $S \subseteq [N]$ of size $\leq r$ the edges $e \supseteq S$ can be efficiently paired-up.

First recall the canonical PPA-complete problem.

Definition 5. Lonely problem is defined on a single array $\textsc{Mate} \in ([2N+1])^{2N+1}$

- $i \in [2N+1]$ if $\textsc{Mate}_{\textsc{Mate}_i} \neq i$, (mismatch)
- $i \in [2N+1]$ if $\textsc{Mate}_i = i$. (lonely node)

Every monomial *occurrence* in the refutation $\sum_{e \in E \setminus \{e^*\}} p_e(h) + 1 = 0$ can be encoded by a pair (e, u) with $e \in E$ and $u \in (e \cup \{\bot\})^r$, let V be the set of these encodings, excluding $(e^*, \bot^r)$. Then $|V| = |E|(\ell+1)^r - 1$ is odd by balancedness for the empty set. A pair (e, u) corresponds to $m_u := \prod_{i \in [r]\colon u_i \neq \bot} h_{u_i,i}$ (if $u_i \in e^*$, then $h_{u_i,i}$ is a constant 0). We define the instance $\textsc{Mate}\colon \mathcal{H} \to V^V$ to reflect the cancellation of the different occurrences of the same monomial. Take some (e, u) such that $m_u(h) = 0$, i.e. the corresponding monomial evaluates to 0 in the refutation. Such monomials are not useful for finding a solution to ZeroEdge$_G$, so we match them up using *syntactic* properties of NS_2 refutation. Let $S \subseteq e$ be the set of non-$\bot$ coordinates in u. By efficient r-balancedness there exists an involution $\sigma_S\colon E_S \to E_S$ that matches-up the edges e containing the set S. Then we define $\textsc{Mate}_{e,u}(h) = (\sigma_S(e), u)$. Now we need to match the monomial occurrences that evaluate to 1 in the refutation. Take $e \in E \setminus \{e^*\}$. If $p_e(h) = 1$, we just define $\textsc{Mate}_{e,\bot^r}(h) = (e, \bot^r)$ and pair-up all the other nodes corresponding to monomials evaluating to 1 on h. If any such node is a solution to the PPA-instance, $(h, e) \in$ ZeroEdge$_G$ by definition. If $p_e(h) = 0$, the number of monomials in p_e that evaluate to 1 is even, so we can pair them arbitrarily. To make the pairings above efficient observe that the set of tuples $u \in (e \cup \{\bot\})^r$ such that $m_u(h) = 1$ is a product set $\prod_{i \in [r]}(\{\bot\} \cup \{v \in e \mid h_{v,i} = 1\})$, so we can efficiently pair the elements up in the lexicographic order (excluding $\bot^r$ in the case $p_e(h) = 1$). Then all solutions correspond to solutions of ZeroEdge$_G$ as required.

Acknowledgements

Ziyi Guan is funded by the Swiss National Science Foundation (SNSF) through a Postdoc.Mobility fellowship. Oliver Korten is supported by the National Science Foundation under Grant No. DMS-2424441. Artur

Riazanov is supported by the Simons Foundation as a Junior Fellow in the Simons Society of Fellows (20599). Noah Fleming is supported by the Swedish Research Council under grant number 2025-06762. Mika Göös and Weiqiang Yuan are supported by the Swiss State Secretariat for Education, Research, and Innovation (SERI) under contract number MB22.00026.

The authors thank Nir Bitansky, Tal Malkin, Yaroslav Alekseev, Gilbert Maystre, Eylon Yogev, and Yiding Zhang for discussions related to this project.

AI disclosure

The separation from PWPP $\not\subseteq$ PLS was obtained by the human authors, using ChatGPT 5.5 for assistance with writing the proof. The human authors obtained the collapse of UEoPL by asking ChatGPT 6 to refine their initial recursive algorithm using results from [BCG26]. The separations PPA $\not\subseteq$ PLS and PPAD $\not\subseteq$ FP were obtained by ChatGPT 6 supplied with the previous two results. In the generated proof, the use of Nullstellensatz equivalence was hidden. This paper is written by humans; all proofs were significantly simplified.

References

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008. URL: https://doi.org/10.1016/j.jcss.2007.06.025, doi:10.1016/J.JCSS.2007.06.025.

[BCE+98] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998. doi:10.1006/jcss.1998.1575.

[BCG26] Benedikt Bünz, Jessica Chen, and Ziyi Guan. It takes two: Proofs of work for Fiat–Shamir. Cryptology ePrint Archive, Paper 2026/2326, 2026. URL: https://eprint.iacr.org/2026/2326.

[BCH+22] Nir Bitansky, Arka Rai Choudhuri, Justin Holmgren, Chethan Kamath, Alex Lombardi, Omer Paneth, and Ron D. Rothblum. PPAD is as hard as LWE and iterated squaring. In *Theory of Cryptography Conference (TCC 2022), Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 593–622. Springer, 2022. doi:10.1007/978-3-031-22365-5_21.

[BDY26] Shany Ben-David and Eylon Yogev. The power of permutable PRPs: Towards trapdoor permutations with full key and message domains. Cryptology ePrint Archive, Paper 2026/2333, 2026. URL: https://eprint.iacr.org/2026/2333.

[BFI23] 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, MIT, Cambridge, Massachusetts, USA, January 10-13, 2023*, volume 251 of *LIPIcs*, pages 30:1–30:40. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.ITCS.2023.30, doi:10.4230/LIPICS.ITCS.2023.30.

[BG20] Nir Bitansky and Idan Gerichter. On the cryptographic hardness of local search. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *LIPIcs*, pages 6:1–6:29. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.6.

[BGIP01] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *Journal of Computer and System Sciences*, 62(2):267–289, 2001. doi:10.1006/jcss.2000.1726.

[BIK+96] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proceedings of the London Mathematical Society*, 73(1):1–26, 1996. Preliminary version in FOCS 1994. doi:10.1112/plms/s3-73.1.1.

[BIK+97] Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiří Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6(3):256–298, 1997. doi:10.1007/BF01294258.

[BKT14] Samuel R. Buss, Leszek Aleksander Kolodziejczyk, and Neil Thapen. Fragments of approximate counting. *J. Symb. Log.*, 79(2):496–525, 2014. URL: https://doi.org/10.1017/jsl.2013.37, doi:10.1017/JSL.2013.37.

[BOM04] Joshua Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004)*, pages 54–67, 2004. doi:10.1109/CCC.2004.1313795.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, pages 1480–1498, 2015. doi:10.1109/FOCS.2015.94.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS 1993)*, pages 62–73, 1993. doi:10.1145/168588.168596.

[BT22] Ilario Bonacina and Neil Thapen. A separation of PLS from PPP. Technical Report TR22-089, Electronic Colloquium on Computational Complexity, 2022. URL: https://eccc.weizmann.ac.il/report/2022/089/.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM*, 56(3):14:1–14:57, 2009. doi:10.1145/1516512.1516516.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC 1996)*, pages 174–183, 1996. doi:10.1145/237814.237860.

[CFJW25] Liyan Chen, Cody Freitag, Zhengzhong Jin, and Daniel Wichs. Unambiguous SNARGs for P from LWE with applications to PPAD hardness. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC 2025)*, pages 944–954, 2025. doi:10.1145/3717823.3718159.

[CHK+19a] Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a Nash equilibrium is no easier than breaking Fiat-Shamir. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019)*, pages 1103–1114, 2019. doi:10.1145/3313276.3316400.

[CHK+19b] Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. PPAD-hardness via iterated squaring modulo a composite. Cryptology ePrint Archive, Paper 2019/667, 2019. URL: https://eprint.iacr.org/2019/667.

[CLRS16] Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. Approximate constraint satisfaction requires large LP relaxations. *Journal of the ACM*, 63(4):34:1–34:22, 2016. Preliminary version in FOCS 2013. doi:10.1145/2811255.

[CP18] Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 451–467. Springer, 2018. doi:10.1007/978-3-319-78375-8_15.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. doi:10.1137/070699652.

[FGH+24] Lukáš Folwarczný, Mika Göös, Pavel Hubáček, Gilbert Maystre, and Weiqiang Yuan. One-Way Functions vs. TFNP: Simpler and Improved. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2024.50.

[FGHS23] John Fearnley, Paul W. Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS = PPAD ∩ PLS. *Journal of the ACM*, 70(1):Article 7, 2023. URL: https://arxiv.org/abs/2011.01929v4, doi:10.1145/3568163.

[FGJ+26] Noah Fleming, Stefan Grosser, Siddhartha Jain, Jiawei Li, Hanlin Ren, Morgan Shirley, and Weiqiang Yuan. Total search problems in ZPP. In Shubhangi Saraf, editor, *17th Innovations in Theoretical Computer Science Conference, ITCS 2026, Bocconi University, Milan, Italy, January 27-30, 2026*, volume 362 of *LIPIcs*, pages 60:1–60:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. URL: https://doi.org/10.4230/LIPIcs.ITCS.2026.60, doi:10.4230/LIPICS.ITCS.2026.60.

[FGMS20] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020. Preliminary version in ICALP 2019. doi:10.1016/j.jcss.2020.05.007.

[GHJ+22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. In *37th Computational Complexity Conference (CCC 2022)*, volume 234 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. URL: https://arxiv.org/abs/2202.07761, doi:10.4230/LIPIcs.CCC.2022.33.

[GHJ+24] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. *Journal of the ACM*, 71(4):Article 26, 2024. URL: https://arxiv.org/abs/2205.02168v3, doi:10.1145/3663758.

[GKRS19] 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)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2019.38, doi:10.4230/LIPIcs.ITCS.2019.38.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a Nash equilibrium. In *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 579–604. Springer, 2016. doi:10.1007/978-3-662-53008-5_20.

[GRY25] Ziyi Guan, Artur Riazanov, and Weiqiang Yuan. Breaking verifiable delay functions in the random oracle model. In *Advances in Cryptology – CRYPTO 2025*, 2025. URL: https://eprint.iacr.org/2024/766.

[HKKS20] Pavel Hubáček, Chethan Kamath, Karel Král, and Veronika Slívová. On average-case hardness in TFNP from one-way functions. In *Theory of Cryptography Conference (TCC 2020), Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 614–638. Springer, 2020. doi:10.1007/978-3-030-64381-2_22.

[HLS+26] Jordan Horacsek, Chin Ho Lee, Igor Shinkar, Emanuele Viola, and Renfei Zhou. Local samplers for product distributions. In Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis, editors, *53rd International Colloquium on Automata, Languages, and*

Programming, ICALP 2026, Royal Holloway, University of London, Egham, United Kingdom, July 7-10, 2026, volume 374 of *LIPIcs*, pages 109:1–109:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. URL: https://doi.org/10.4230/LIPIcs.ICALP.2026.109, doi:10.4230/LIPICS.ICALP.2026.109.

[HNY17] Pavel Hubáček, Moni Naor, and Eylon Yogev. The journey from NP to TFNP hardness. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *LIPIcs*, pages 60:1–60:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.60.

[HY20] Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. *SIAM Journal on Computing*, 49(6):1128–1172, 2020. Preliminary version in SODA 2017. doi:10.1137/17M1118014.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT 1995)*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999. doi:10.1007/s000370050024.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC 1989)*, pages 44–61, 1989. doi:10.1145/73007.73012.

[Jeř16] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016. doi:10.1016/j.jcss.2015.08.001.

[JKKZ21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, pages 708–721, 2021. doi:10.1145/3406325.3451055.

[JLRX24] Siddhartha Jain, Jiawei Li, Robert Robere, and Zhiyang Xun. On pigeonhole principles and Ramsey in TFNP. In *65th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2024)*, pages 406–428, 2024. doi:10.1109/FOCS61266.2024.00033.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.

[Kam19] Pritish Kamath. *Some hardness escalation results in computational complexity theory*. PhD thesis, Massachusetts Institute of Technology, 2019. URL: https://pritishkamath.github.io/files/phd-thesis.pdf.

[Kis26] Fedor Kiselev. Randomized separations in black-box TFNP. arXiv:2606.04697, 2026. URL: https://arxiv.org/abs/2606.04697.

[KLV23] Yael Tauman Kalai, Alex Lombardi, and Vinod Vaikuntanathan. SNARGs and PPAD hardness from the decisional Diffie-Hellman assumption. In *Advances in Cryptology – EUROCRYPT 2023, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 470–498. Springer, 2023. doi:10.1007/978-3-031-30617-4_16.

[KMR17] Pravesh K. Kothari, Raghu Meka, and Prasad Raghavendra. Approximating rectangles by juntas and weakly-exponential lower bounds for LP relaxations of CSPs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 590–603, 2017. doi:10.1145/3055399.3055438.

[KNY18] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 162–194. Springer, 2018. doi:10.1007/978-3-319-78375-8_6.

[KNY19] Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *Journal of the ACM*, 66(5), 2019. Preliminary version in FOCS 2017. doi:10.1145/3341106.

[KPY20] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. Delegation with updatable unambiguous proofs and PPAD-hardness. In *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 652–673. Springer, 2020. doi:10.1007/978-3-030-56877-1_23.

[KSS10] Jeff D. Kahn, Michael E. Saks, and Clifford D. Smyth. The dual bkr inequality and rudich's conjecture. *Combinatorics, Probability and Computing*, 20:257 – 266, 2010. URL: https://api.semanticscholar.org/CorpusID:39276849.

[KT22] Leszek Aleksander Kołodziejczyk and Neil Thapen. Approximate counting and NP search problems. *Journal of Mathematical Logic*, 22(3):2250012, 2022. doi:10.1142/S021906132250012X.

[Li24] Jiawei Li. Total NP search problems with abundant solutions. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287 of *LIPIcs*, pages 75:1–75:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ITCS.2024.75.

[LRS15] James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 567–576, 2015. doi:10.1145/2746539.2746599.

[LV20] Alex Lombardi and Vinod Vaikuntanathan. Fiat-shamir for repeated squaring with applications to PPAD-hardness and VDFs. In *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 632–651. Springer, 2020. doi:10.1007/978-3-030-56877-1_22.

[McD89] Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics, 1989*, volume 141 of *London Mathematical Society Lecture Note Series*, pages 148–188. Cambridge University Press, 1989. doi:10.1017/CBO9781107359949.008.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. doi:10.1137/S0097539795284959.

[Mor01] Tsuyoshi Morioka. Classification of search problems and their definability in bounded arithmetic. Master's thesis, University of Toronto, 2001. ECCC TR01-082.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

[PT19] Pavel Pudlák and Neil Thapen. Random resolution refutations. *Comput. Complex.*, 28(2):185–239, 2019. URL: https://doi.org/10.1007/s00037-019-00182-7, doi:10.1007/S00037-019-00182-7.

[Raz87] 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. Translated from Matematicheskie Zametki 41(4):598–607. doi:10.1007/BF01137685.

[Raz98] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998. doi:10.1007/s000370050013.

[Raz03] Alexander A. Razborov. Resolution lower bounds for the weak functional pigeonhole principle. *Theoretical Computer Science*, 303(1):233–243, 2003. Logic and Complexity in Computer Science. URL: https://www.sciencedirect.com/science/article/pii/S030439750200453X, doi:10.1016/S0304-3975(02)00453-X.

[RSS21] Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? *Journal of Cryptology*, 34(1):8, 2021. Preliminary version in TCC 2017. doi:10.1007/s00145-020-09369-6.

[Rud88] Steven Rudich. *Limits on the Provable Consequences of One-way Functions*. PhD thesis, EECS Department, University of California, Berkeley, Dec 1988. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1988/6060.html.

[Rud07] Atri Rudra. *List Decoding and Property Testing of Error Correcting Codes*. PhD thesis, University of Washington, 2007.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC 1987)*, pages 77–82, 1987. doi:10.1145/28395.28404.

[Tse70] Grigori S. Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part II*, volume 8 of *Seminars in Mathematics, Steklov Mathematical Institute*, pages 115–125. 1970.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.

[YZ24] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. *Journal of the ACM*, 71(3):Article 20, 2024. URL: https://arxiv.org/abs/2204.02063v3, doi:10.1145/3658665.

A TFNP glossary

In this section, we give formal definitions of all TFNP classes we use. With the exception of the entire class TFNP and the smallest subclass (total) FP that contains efficiently solvable total relations, all subclasses are defined by canonical complete problems. Then if F is a complete relation for some class $\mathcal{C}$ we say that $F' \in \mathcal{C}$ iff there exists a zero-error public reduction (see Section 5) from F' to F.

PPP (Pigeon). We are given $\textsc{Pigeon} \in [N]^{N+1}$; a solution is

- $i \neq j \in [N+1]$ such that $\textsc{Pigeon}_i = \textsc{Pigeon}_j$. (collision)

PWPP (WeakPigeon). We are given $\textsc{Pigeon} \in [N]^{2N}$; a solution is

- $i \neq j \in [2N]$ such that $\textsc{Pigeon}_i = \textsc{Pigeon}_j$. (collision)

PPA (Lonely). We are given $\textsc{Mate} \in [2N+1]^{2N+1}$; solutions are

- $i \in [2N+1]$ such that $\textsc{Mate}_i = i$; (lonely node)
- $i \in [2N+1]$ such that $\textsc{Mate}_{\textsc{Mate}_i} \neq i$. (mismatch)

PPAD (End of Line). We are given $\textsc{Succ}, \textsc{Pred} \in [N]^N$; solutions are

- 1 if $\textsc{Pred}_1 \neq 1$ or $\textsc{Succ}_1 = 1$ or $\textsc{Pred}_{\textsc{Succ}_1} \neq 1$; (distinguished node is not a source)
- $i \in [N] \setminus \{1\}$ if $\textsc{Succ}_{\textsc{Pred}_i} \neq i$. (non-canonical source)
- $i \in [N]$ if $\textsc{Pred}_{\textsc{Succ}_i} \neq i$. (active sink)

A common grid formulation. For PLS, EoPL, and UEoPL, use the $N \times N$ grid, $N \geq 2$, of Definition 3. Every node has a successor $\textsc{Succ}_{ij} \in [N] \cup \{\bot\}$; for $i < N$, a non-$\bot$ value k names $(i+1, k)$. For PLS a node is *active* if $\textsc{Succ}_{ij} \neq \bot$ (for EoPL and UEoPL we restrict the activity requirement). A *proper sink* is an inactive node reached from an active node.

PLS (Sink of Dag). Solutions are

- $(1,1)$ if it is inactive; (no canonical source)
- (N, j) if it is active; (sink)
- (i, j) with $i < N$ if it is active, but $(i+1, \textsc{Succ}_{ij})$ is inactive. (predecessor of a proper sink)

In the third case we output the active predecessor, which witnesses an incoming edge to the inactive sink. This makes the solution locally checkable even though PLS has no predecessor pointers, as in [GHJ+24, Section 3.2].

EoPL (End of Potential Line). Add pointers $\textsc{Pred}_{ij} \in [N] \cup \{\bot\}$ for rows $2, \ldots, N$, and **redefine** activity throughout: for $i < N$, node (i, j) is active iff $\textsc{Succ}_{ij} = k \neq \bot$ and $\textsc{Pred}_{i+1,k} = j$; in row N, activity still means $\textsc{Succ}_{Nj} \neq \bot$. Keep the first two PLS solution types with this activity definition, replace the third by the proper sink itself, and add proper sources:

- (i, j) if it is a proper sink; (proper sink)
- $(i, j) \neq (1, 1)$ with $i < N$ if it is active and no active node has it as a successor. (proper source)

In particular, every active node in row 1 except $(1, 1)$ is a proper source. A proper sink itself is locally checkable here because its predecessor pointer identifies the only possible active predecessor; see Definition 3.

UEoPL (Unique End of Potential Line). Use the same instance, activity, and four solution types as EoPL, and add just one further type:

- (i, j) and (i, j') with $j \neq j'$ both active. (parallel active nodes)

B TFNP separations in the ROM

In this section, we will give a formal definition of TFNP problems and reductions under the random oracle model. Let $\Gamma_{a(n),b(n)}$ be the set of all Boolean functions $\gamma \colon \{0,1\}^{a(n)} \to \{0,1\}^{b(n)}$.

Definition 6 (Search problem in an oracle model). A search problem in the oracle-model of $\Gamma_{a(n),b(n)}$ is a family of relations $\mathcal{R} = \{R_n\}_{n \geq 1}$ where $R_n \subseteq \{0,1\}^n \times \Gamma_{a(n),b(n)} \times \{0,1\}^{s(n)}$. $\mathcal{R}$ is total if for all n, $x \in \{0,1\}^n$ and $\gamma_n \in \Gamma_{a(n),b(n)}$ there exists y such that $(x, \gamma_n, y) \in R_n$.

We say that an algorithm A average-case solves some oracle search problem $\mathcal{R}^{(\cdot)}$ in the random oracle model if for all sufficiently large n, when γ_n is sampled uniformly over all functions from $a(n)$ bits to $b(n)$ bits,

$$\Pr_{\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{\gamma}_n \sim \Gamma_{a(n),b(n)}} [A^{\boldsymbol{\gamma}_n}(\boldsymbol{x}) \text{ outputs } y \text{ such that } (\boldsymbol{x}, \boldsymbol{\gamma}_n, y) \in R_n] \geq \frac{2}{3}.$$

We say that an oracle TFNP search problem $\mathcal{R}^{(\cdot)}$ is *efficiently reducible* to $\mathcal{S}^{(\cdot)}$ *in the ROM*, if there are oracle algorithms $f^{(\cdot)}, g^{(\cdot)}$ in $\mathrm{poly}(n)$ time, such that

$$\Pr_{\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{\gamma}_n \sim \Gamma_{a(n),b(n)}} [\forall y,\ (f^{\boldsymbol{\gamma}_n}(1^n, \boldsymbol{x}), \gamma_n, y) \in \mathcal{S}^{(\cdot)} \implies (\boldsymbol{x}, \boldsymbol{\gamma}_n, g^{\boldsymbol{\gamma}_n}(1^n, \boldsymbol{x}, y)) \in \mathcal{R}^{(\cdot)}] \geq \frac{2}{3}.$$

One may assume that $a(n), b(n), s(n)$ are all bounded by polynomials. For all the *circuit TFNP problems* we mentioned in this paper, i.e., TFNP problems, where the input is the description of a circuit whose truth table encodes an object of exponential length, it is natural to define the corresponding oracle TFNP problem by letting the input be descriptions of *oracle circuits*. For example, the oracle search problem $\mathsf{PWPP}^{(\cdot)}$ has the

input x being the description of some oracle circuit $C^{(\cdot)}\colon \{0,1\}^m \to \{0,1\}^{m-1}$, and $(x, \gamma_n, (y_1, y_2)) \in \mathsf{PWPP}^{(\cdot)}$ if and only if $C^{\gamma_n}(y_1) = C^{\gamma_n}(y_2)$ for $y_1 \neq y_2$.

Note that any of the circuit TFNP problems we considered in this paper is equivalent to *another* problem where the input oracle circuit is promised to be the canonical circuit $\widehat{C}^{\gamma}(x) \coloneqq \gamma(x)$. It is not hard to see that reductions between oracle versions of the TFNP problems in the ROM can be viewed as *uniform* versions of average-case reductions between the corresponding black-box problems, where each node of the decision trees can be locally printed by a uniform algorithm in $\mathrm{polylog}(N)$ time.