Electronic Colloquium on Computational Complexity, Report No. 319 (2026)

Power and Limitations of TFNP Hierarchies

Surendra Ghentiyala* Oliver Korten† Toniann Pitassi‡

November 4, 2026

Abstract

It is well-known that if P = NP, then the seemingly larger class PH also collapses to P; hence, we may base the hardness of NP on that of PH, despite the widespread belief that PH $\not\subseteq$ NP. Although well established in the decision problem setting, few such "conditional collapse theorems" are known for *total* NP search problems. In this work, by building on the ideas of Ghentiyala and Li [GL25] and drawing an analogy with the polynomial hierarchy, we study conditional collapses in TFNP. Our primary focus is on the WeakPigeon hierarchy: the class of problems obtained by *iterating* the WeakPigeon problem via oracles (e.g. $\text{WeakPigeon}, \text{WeakPigeon}^{\text{WeakPigeon}}, \text{WeakPigeon}^{(\text{WeakPigeon}^{\text{WeakPigeon}})}$, etc.).

First, we find that the problem of finding a multicollision of any fixed constant size (in a sufficiently compressing function) is contained in the WeakPigeon hierarchy, making it the first known TFNP problem where iteration non-trivially gives more power. Consequently, if PWPP = FP then the problem of finding multicollisions also collapses to FP. Our proof can be viewed as a reinterpretation of Buzek and Tessaro's "non-black-box" construction of collision resistant hash functions (CRHs) from multicollision-resistant hash functions (MCRHs) [BT24]. Interestingly, in the framework of TFNP problem iteration, this theorem follows from a black-box reduction, leading us to believe that TFNP problem iteration may be a useful explanatory framework for certain non-black-box constructions in cryptography. Our proposed shift in perspective also yields other new conditional collapses. For example, we see that if PPP = promiseUEOPL or PWPP = LOSSY, then the problem of finding multicollisions also collapses to promiseUEOPL or LOSSY respectively.

Having demonstrated that the WeakPigeon hierarchy is surprisingly powerful, we next show that it nonetheless does not contain the TFNP class PPAD (and hence any of its superclasses e.g. PPP, PPA) under black-box reductions. Our separation utilizes the connection between TFNP classes and proof complexity: we prove that bounded-depth Frege has efficient proofs of the tautology asserting the totality of problems in the WeakPigeon hierarchy, while on the other hand the tautology asserting the totality of PPAD is known to be hard for bounded-depth Frege. These insights into the structure of iterated TFNP problems inform us about both the promise and limitations of using our techniques to obtain further conditional collapses in TFNP.

*Cornell University. Email: sg974@cornell.edu. This work was supported by NSF Grant No. CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google. Part of this work was done during a research visit to Columbia University.

†Institute for Advanced Study. korten@ias.edu Oliver Korten is supported by the National Science Foundation under Grant No. DMS-2424441.

‡Columbia University. tp2684@columbia.edu

ISSN 1433-8092

Contents

1 Introduction **1**
- 1.1 Our results . . . 1
- 1.2 Open problems . . . 4
- 1.3 Related work . . . 5

2 Preliminaries **6**
- 2.1 White-Box TFNP . . . 6
- 2.2 Black-Box TFNP . . . 8

3 Defining our hierarchies **9**
- 3.1 Hierarchies within TFNP . . . 9
- 3.2 Hierarchies and consistent subwitnesses . . . 12

4 Conditional collapses **13**
- 4.1 Uniform, non-uniform, and randomized collapses . . . 13
- 4.2 Conditional collapse of the Either(A, B) hierarchy . . . 15

5 The WeakPigeon hierarchy and the consequences of its collapse **16**
- 5.1 Multi-Collisions in the WeakPigeon hierarchy . . . 16
- 5.2 A collapse of the Pigeon hierarchy if PPP = promiseUEOPL . . . 17
- 5.3 A collapse of the WeakPigeon hierarchy if PWPP = LOSSY . . . 20

6 Separations from the WeakPigeon Hierarchy via Proof Complexity **21**

7 Theorems too good to be true **25**

A Lifting some reductions **27**

B Deferred proofs from Section 7 **28**

1 Introduction

Say that one wants to show that solving problems in a complexity class $\mathcal{C}_1$ is at least as hard as solving those in a class $\mathcal{C}_2$. The most basic approach would be to show that $\mathcal{C}_2 \subseteq \mathcal{C}_1$, or more generally that $\mathcal{C}_2 \subseteq \mathsf{P}^{\mathcal{C}_1}$, via a many-one or Turing reduction from a complete problem for $\mathcal{C}_2$ to some problem in $\mathcal{C}_1$. However, it is also possible to show that $\mathcal{C}_1$ is "as hard to solve efficiently" as $\mathcal{C}_2$ *even in the case that we believe* $\mathcal{C}_2 \not\subseteq \mathsf{P}^{\mathcal{C}_1}$ using the upward collapse of some class hierarchy. The most important (and historically first) example of this phenomenon comes from the polynomial hierarchy, where e.g., one can show that if NP collapses to P then so does Σ_2^{P}, despite the fact that $\Sigma_2^{\mathsf{P}} \not\subseteq \mathsf{P}^{\mathsf{NP}}$ is widely believed. Similar phenomena occur in other complexity hierarchies such as the NC and Counting hierarchies [Pip79, Wag86]. Underlying all such upward-collapse theorems is a common principle. Say we have some complexity class $\mathcal{C}$ defined in terms of P (e.g., NP is defined from P by existential quantification). If we find an algorithm to solve problems in $\mathcal{C}$ efficiently, we can imagine we have made P "larger" by expanding it to contain $\mathcal{C}$. Since $\mathcal{C}$ is defined in terms of P, and P has expanded to become more expressive, we have in turn expanded the reach of $\mathcal{C}$ to some larger class $\mathcal{C}'$, which in turn collapses to P as well.

In recent work of Ghentiyala and Li [GL25], possibilities of such upward-collapse phenomena were investigated for the first time in the context of TFNP search problems, which are NP search problems which always have solutions [Pap94]. Consider for concreteness the TFNP problem WeakPigeon (the subject of much of this work) of finding collisions in compressing circuits: given a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^{n-1}$, find $x \neq x' \in \{0,1\}^n$ such that $C(x) = C(x')$. If WeakPigeon $\in \mathsf{FP}$ then there is a polynomial size circuit S which finds such x, x' given a description of C. Hence we may incorporate S into some new WeakPigeon instance $D : \{0,1\}^n \to \{0,1\}^{n-1}$, where on each input x, D uses x to define a WeakPigeon instance C_x, runs $S(C_x)$ to find a collision, and uses that collision to determine its output $D(x)$. Applying the fact that WeakPigeon $\in \mathsf{FP}$ again, we can then find collisions in the circuit D. Does this give us any more power than using a collision-finding oracle on compressing functions which we already unconditionally know have polynomial-size circuits? More generally, for any TFNP problem $\mathcal{C}$, can we identify interesting problems reducible to $\mathcal{C}^{\mathcal{C}}$ which do not seem to reduce to $\mathcal{C}$? In what follows, we will refer to $\mathcal{C}^{\mathcal{C}}$ as the "iteration" of the search problem $\mathcal{C}$. As discovered in the prior work of Ghentiyala and Li [GL25], there is significantly more subtlety in formalizing the notion of iteration for *search problems* in contrast to decision problems; we will discuss these subtleties as they arise in what follows.

Although the conditional-collapse style of reasoning is largely absent from the TFNP literature, we observe in this work that it has appeared implicitly in the cryptography literature. Buzek and Tessaro [BT24] give a non-black-box construction of (infinitely often) CRHs (hash functions in which it is hard to find a pairwise collision) from MCRHs (hash functions in which it is hard to find many collisions at a single output). In Section 1.1.1, we show how one of the key ideas of their proof can in fact be viewed as a reduction from the multicollision problem to the *hierarchy* over WeakPigeon. One of the key benefits of this perspective is that we are able to take the non-black-box style of reasoning used in the cryptographic construction and put it into a unified and familiar framework of collapsing hierarchies. This perspective allows us to extend their results in directions less applicable to cryptography but of greater interest from the TFNP perspective. It will also give us the tools to understand the limits of the non-black-box framework of [BT24] by understanding the limits of hierarchies within the TFNP framework.

1.1 Our results

Our technical results are all centered on the idea of TFNP problems with oracles, that is, A^{B} where A and B are TFNP problems (see Section 3 for a formal definition). As an illustrative example—and a recurring one—consider $\mathrm{WeakPigeon}^{\mathrm{WeakPigeon}}$. The input to this problem is an oracle circuit $C^{\mathrm{WeakPigeon}} : \{0,1\}^n \to \{0,1\}^{n-1}$ and the goal should be to find a collision in $C^{\mathrm{WeakPigeon}}$: distinct $x_1, x_2 \in \{0,1\}^n$ such that $C^{\mathrm{WeakPigeon}}(x_1) = C^{\mathrm{WeakPigeon}}(x_2)$. However, as is, the behavior of $C^{\mathrm{WeakPigeon}}$ is not well-defined since for any $x \in \{0,1\}^n$, $C^{\mathrm{WeakPigeon}}$ may have many possible answers depending on the output of the oracle gates of $C^{\mathrm{WeakPigeon}}$. In accordance with the work of Ghentiyala and Li [GL25], we remedy this by asking that a solution consist not just of a collision x_1, x_2, but also of the answers to the oracle queries

required to evaluate $C^{\text{WeakPigeon}}(x_1), C^{\text{WeakPigeon}}(x_2)$. The answers to these queries fix an evaluation of $C^{\text{WeakPigeon}}(x_1), C^{\text{WeakPigeon}}(x_2)$. Our definition is somewhat weaker than that of [GL25] in that it does not enforce a property they call internal consistency, which has its own advantages and disadvantages (see Section 3.2). In fact, the observant reader may have noticed that we state almost all our results in terms of TFNP problems rather than their corresponding classes—we refer to $\text{WeakPigeon}^{\text{WeakPigeon}}$ rather than $\text{PWPP}^{\text{PWPP}}$. This is a result of the fact that the latter notion seems to make less sense under our definition than it does under the definition of [GL25] (see Section 3.2).

We observe that $\text{WeakPigeon}^{\text{WeakPigeon}}$ is itself a TFNP problem. Therefore, we can define and reason about $\text{WeakPigeon}^{(\text{WeakPigeon}^{\text{WeakPigeon}})}$, which we refer to as the third level of the WeakPigeon hierarchy. This is itself also a TFNP problem, which means we can continue to iterate our definition. So for any TFNP circuit problem A, we obtain a whole hierarchy of such problems, where we refer to A^i as the ith level of the hierarchy (A^0 is an FP-complete problem) and A^* as the class of problems reducible to some level of the hierarchy (see Definition 3.4).

With our definitions of TFNP oracle problems and their hierarchies in place, we establish some theorems about when these hierarchies collapse. These results should be reminiscent of the collapses that occur in other hierarchies (e.g., PH) and will be key to the conditional collapses we prove later.

Observation 4.1. *Let* A *be a* TFNP *circuit problem and i be any nonnegative integer. If* A^{i+1} *reduces to* A^i *under (non-uniform) many-one reductions, then for any $j \geq i$,* A^j *reduces to* A^i *under (non-uniform) many-one reductions.*

Corollary 4.3. *Let* A *be a* TFNP *circuit problem and i be any nonnegative integer. If* A^{i+1} *reduces to* A^i *under randomized many-one reductions, then for any $j \geq i$,* A^j *reduces to* A^i *under randomized many-one reductions.*

Observation 4.1 and Corollary 4.3 should collectively be read as stating that TFNP hierarchies behave as expected: if level i and $i+1$ are equal, then the whole hierarchy "collapses" to level i (and this holds for uniform, non-uniform, and randomized reductions).

We also ask how our hierarchies are related to higher-level structures within TFNP. Here we are particularly concerned with intersection classes of the form $\text{PLS} \cap \text{PPAD}$ or $\text{PPA} \cap \text{PWPP}$ (the latter is of particular interest since it contains the problem of integer factoring [Jeř16]). Therefore, in Section 4.2, we continue the study of when hierarchies of problems collapse by studying hierarchies of problems complete for intersection classes. We have no evidence that the hierarchies of intersection problems do or do not collapse. However, here again, we are able to show conditional collapses.

1.1.1 The WeakPigeon hierarchy and the consequences of its collapse

Having defined our hierarchies and established that collapses behave as expected, we show that these hierarchies contain interesting problems. In particular, our first result shows that finding a 2^ℓ-wise collision in a function that compresses by (approximately) a factor of ℓ is in the ℓth level of the WeakPigeon hierarchy.

Our proof should be viewed as providing a new perspective on the recent work of [BT24] which gives a non-black-box construction of CRHs from MCRHs as follows (we are simplifying greatly). They first construct a plausible CRH f. They argue that if f is indeed collision resistant, we are done. If f is not collision-resistant, then there exists an adversary $\mathcal{A}$ which finds collisions in f. In the latter case, they use $\mathcal{A}$ to construct a CRH g. The use of the adversary $\mathcal{A}$ in this win-win argument is precisely what makes this argument non-black-box. In essence, they are arguing that if one can find collisions in both f and g (where g depends on the collision finder for f), then we can find multicollisions in the original MCRH, thus producing a contradiction. In the language of hierarchies, they are reducing the problem of finding multicollisions in the proposed MCRH to a $\text{WeakPigeon}^{\text{WeakPigeon}}$ instance $C^{\text{WeakPigeon}}$ where oracle gates of the instance correspond to f and the whole $C^{\text{WeakPigeon}}$ corresponds to g. They then make the following argument: either solving the internal WeakPigeon oracle gate instances is hard, in which case CRHs exist, or it is easy, in which case the $\text{WeakPigeon}^{\text{WeakPigeon}}$ instance reduces to $\text{WeakPigeon}^{\text{FP}} = \text{WeakPigeon}$, which must now be a CRH.

Adapting this reasoning to TFNP yields the following containment for the WeakPigeon hierarchy. Here, t-Pigeon(M, N) denotes the problem of finding a t-wise collision in $C : [M] \to [N]$ (see Definition 2.11).

Theorem 5.1. *For every positive integer ℓ, 2^ℓ-Pigeon$(2^{\ell n}, 2^{n-1})$ reduces to* WeakPigeon$^\ell$.

Theorem 5.1 stands in contrast to the fact that finding even a 3-wise collision in a hash function with nontrivial compression is not black-box reducible to Pigeon (let alone WeakPigeon) [JLRX24]. This indicates that the WeakPigeon hierarchy is much more powerful than WeakPigeon.

We next combine Theorem 5.1 with our observations from Section 4.1 about when hierarchies collapse. In particular, we can conclude that a collapse of WeakPigeon to a sufficiently small class would collapse all of WeakPigeon* to the same class. This would allow us to conclude that the problem of finding multicollisions collapses to that same class.

Corollary 5.3. *Let $\mathcal{C} \in \{\mathsf{FP}, \mathsf{TFZPP}, \mathsf{FP/poly}\}$ and $\ell \in \mathbb{N}$.* WeakPigeon $\in \mathcal{C}$ *if and only if* 2^ℓ-Pigeon$(2^{\ell n}, 2^{n-1}) \in \mathcal{C}$.

In comparison to [BT24], Theorem 5.1 has the added benefit that it is completely black-box, while the proof in [BT24] requires reasoning about a possible adversary. The only part of our reasoning which may be viewed as "gray-box" is the reasoning required to go from Theorem 5.1 to Corollary 5.3 where we argue that the WeakPigeon hierarchy collapses if WeakPigeon $\in \mathcal{C}$. Therefore, the non-black-box construction of [BT24] (and previously [RV24]) becomes black-box in our framework. This allows us to study these cryptographic constructions in a systematic framework through the lens of TFNP problem iteration and hierarchies: we see when we expect similar techniques to yield more results and what fundamental barriers these techniques encounter (see Section 1.1.2 and Section 1.1.3).

Furthermore, Theorem 5.1 gives us some insight into what makes Pigeon and WeakPigeon different from other TFNP problems. [GL25] showed that PPA, PLS, and LOSSY are *self-low*, meaning that iterating their complete search problems does not increase their complexity. However, Theorem 5.1 tells us that this is likely not the case for WeakPigeon:

Corollary 5.5. *There is no black-box reduction from* WeakPigeon^2 *to* Pigeon.

Viewing Theorem 5.1 from the perspective of a hierarchy collapse rather than the non-black-box construction perspective of [BT24] yields other, new conditional collapses. For example, we find that if PPP $\subseteq$ promiseUEOPL, then the whole Pigeon hierarchy collapses to promiseUEOPL. And since, by Theorem 5.1, Pigeon* contains the problem of finding multicollisions, if PPP $\subseteq$ promiseUEOPL, the problem of finding multicollisions in a sufficiently compressing function is also in promiseUEOPL. We then do the same for a conditional collapse of the WeakPigeon hierarchy in the case that PWPP = LOSSY. This kind of reasoning (as far as we can tell) only makes sense in the TFNP hierarchies style of reasoning rather than the cryptographic constructions perspective.

Corollary 5.9. *If* PPP $\subseteq$ promiseUEOPL*, then* Pigeon* $\subseteq$ promiseUEOPL.

Corollary 5.10. *If* PPP = promiseUEOPL*, then for any positive integer ℓ,* 2^ℓ-Pigeon$(2^{\ell n}, 2^{n-1}) \in$ promiseUEOPL.

Corollary 5.12. PWPP = LOSSY *if and only if for all positive integers i,* WeakPigeon^i *is equivalent to* Lossy *under many-one reductions. Furthermore,* PWPP = LOSSY *if and only if for every positive integer ℓ,* 2^ℓ-Pigeon$(2^{\ell n}, 2^{n-1})$ *reduces to* Lossy.

Corollary 5.9 is proven using an expanded notion of TFNP problems with oracles which emphasizes oracles for search problems with unique solutions (see Section 5.2.1), which may be of independent interest. We find Corollaries 5.10 and 5.12 somewhat surprising since the problem of finding multicollisions is not related to any of PPP, UEOPL, PWPP or LOSSY by black-box reductions [JLRX24]. However, we are able to show that if PPP $\subseteq$ promiseUEOPL or PWPP = LOSSY, then the set of problems involving finding multicollisions (which is orthogonal to the standard TFNP classes) also collapses down to promiseUEOPL or LOSSY, respectively. Corollary 5.10 can therefore be seen as additional and rather strong evidence that PPP $\not\subseteq$ promiseUEOPL; not

only would PPP $\subseteq$ promiseUEOPL be a surprising collapse in its own right, but it would also imply that the problem of finding multicollisions is also in promiseUEOPL. Similarly, Corollary 5.12 can be seen as additional evidence that PWPP $\neq$ LOSSY, since equality would imply that the seemingly unrelated problem of finding multicollisions is also in LOSSY.

1.1.2 Black-box iteration from the perspective of proof complexity

Results in the previous section indicate the nontrivial power of iterating the WeakPigeon search problem, and the question naturally arises whether other TFNP problems which are known to lie outside of PWPP nonetheless fall inside the WeakPigeon hierarchy. In Section 6 we answer this question in the negative for the classes PPAD, PPP, and PPA.

Theorem 6.6. *In the black-box model, the complete problem for* PPAD *is not many-one reducible to* WeakPigeon^i *for any constant* i*. Consequently, the same holds for any superclass of* PPAD*, e.g.,* PPP*,* PPA*.*

Theorem 6.6 is established by showing that any black-box search problem which is efficiently reducible to the WeakPigeon hierarchy has an efficient proof of totality in bounded-depth Frege. Combined with the known fact that PPAD, PPP, PPA contain problems which are not provably total in subexpontial size bounded-depth Frege, this yields the desired separation in Theorem 6.6. This fact is derived by combining the classical bounded depth proof of the weak pigeonhole principle of Paris, Wilkie and Woods [PWW88] with the following general result.

Theorem 6.4 (Informal)**.** *If* $\mathcal{S}$ *is a total search problem, then bounded depth Frege can efficiently prove the totality of* $\text{WeakPigeon}^{\mathcal{S}}$ *using the totality of* $\mathcal{S}$ *as an axiom.*

The above implies that if $\mathcal{S}$ has efficient bounded depth proofs, then so does every problem black-box reducible to $\text{WeakPigeon}^{\mathcal{S}}$. Repeatedly applying this theorem with $\mathcal{S} = \text{WeakPigeon}, \mathcal{S} = \text{WeakPigeon}^2, \ldots,$ we conclude that bounded depth Frege can prove the totality of any fixed level of the WeakPigeon hierarchy.

1.1.3 Barriers against further conditional collapses

Theorem 5.1 is a containment and leads to a rather surprising conditional collapse (Corollary 5.3). Upon seeing such a collapse, one might hope for similar conditional collapses across other TFNP hierarchies. For example, if PPADS is contained in the EoL hierarchy, then we would have the dream theorem that if PPAD $\subseteq$ FP, then PPADS $\subseteq$ FP. We show that, unfortunately, such theorems seem unlikely.

Theorem 7.1. *If for some nonnegative integer* i*,* PPADS *has a many-one reduction to* EoL^i*, then* PPADS $\subseteq$ PPA*.*

Similarly, one might hope for a theorem of the form PPAD is contained in the CLO (Continuous-Local-Opt) hierarchy, which would give us the dream theorem that CLS = FP $\implies$ PPAD = FP. Or one might hope for a theorem like PLS is contained in the CLO (Continuous-Local-Opt) hierarchy, which would give us the dream theorem that CLS = FP $\implies$ PLS = FP. Unfortunately, theorems of either form would imply collapses of TFNP classes we have good evidence to believe are separate [Mor01, BOM04].

Theorem 7.2. *If for some nonnegative integer* i*,* PPAD *has a many-one reduction to* CLO^i*, then* PPAD $\subseteq$ PLS*. Similarly, if for some nonnegative integer* i*,* PLS *has a many-one reduction to* CLO^i*, then* PLS $\subseteq$ PPA*.*

1.2 Open problems

Black-box TFNP and proof complexity The connection between proof complexity and TFNP has yielded insights in both areas. Here we initiate the study of TFNP problem iteration in the black-box setting (see Section 3 and Section 6) and in particular show in Section 6 that sufficiently strong proof systems can reason about the iteration of $\text{WeakPigeon}^{\text{dt}}$ at essentially zero cost. We use this to show that PPAD^{dt} is not contained in the $\text{WeakPigeon}^{\text{dt}}$ hierarchy. Our proof method, by design, cannot separate from the

WeakPigeon hierarchy any class $\mathcal{C}$ whose complete tautologies have efficient bounded-depth Frege proofs. In the standard landscape of natural TFNP classes, this leaves out PLS and all its subclasses, and we highlight the open problem of showing that PLS (or more strongly, one of its subclasses) is not contained in the WeakPigeon hierarchy. We also leave it as an open question if one can separate all levels of the WeakPigeon hierarchy (we have separated the first and second levels in Corollary 5.5). Finally, we ask if there is any kind of characterization of the self-low TFNP subclasses in terms of their corresponding proof systems. Notably, the two interesting examples of classes which we know to not be self-low are PPP and PWPP which also happen to be two of the only standard TFNP classes which do not seem to correspond to any natural proof system. Furthermore, a look at Theorem 6.4 also suggests that one may be able to prove a similar theorem where WeakPigeon is substituted with $\mathcal{S}$. This would imply that if $\mathcal{S}$ corresponds to a sufficiently strong proof system, then $\mathcal{S}^{\mathcal{S}}$ is provable in bounded depth Frege using $\mathcal{S}$ as a substitution axiom, and therefore reduces to $\mathcal{S}$. Can one show that sufficiently strong proof systems correspond to TFNP principles which are self-low? Currently, we do not know of any formal property of proof systems which will predict in general whether their corresponding TFNP class is self-low or not.

A relativization barrier. In one sense, our result that if $\mathsf{PWPP} = \mathsf{FP}$, then 2^{ℓ}-Pigeon$(2^{\ell n}, 2^{n-1}) \in \mathsf{FP}$ bypasses the black-box barrier of [JLRX24] which implicitly states that there exists an oracle O such that $2^{\ell}\text{-Pigeon}(2^{\ell n}, 2^{n-1})^O \notin \mathsf{PWPP}^O$. We note that almost all black-box barriers in the TFNP landscape are of this form: we have two likely intractable classes A, B within TFNP and show the existence of an oracle O such that $\mathsf{A}^O \not\subseteq \mathsf{B}^O$. We show that hierarchy theorems allow us to circumvent these types of barriers. However, our proofs run into a relativization barrier. It is not hard to see that relative to all oracles O, if $\mathsf{PWPP}^O = \mathsf{FP}^O$, then $2^{\ell}\text{-Pigeon}(2^{\ell n}, 2^{n-1})^O \in \mathsf{FP}^O$, meaning our proof relativizes. Therefore, to rule out that the hierarchies style of reasoning can be used in a relativizing way to relate the collapse of A to the collapse of B, it would suffice to construct an oracle O relative to which $\mathsf{A}^O = \mathsf{FP}^O$ but $\mathsf{B}^O \neq \mathsf{FP}^O$. Note that this is different from showing an oracle O such that $\mathsf{A}^O \not\subseteq \mathsf{B}^O$, as is standard in the TFNP literature. It would therefore be interesting to show, for all the standard TFNP classes A, B (e.g., PPAD, PLS, etc.) which are believed to be distinct, the existence of oracles O such that $\mathsf{A}^O = \mathsf{FP}^O$ and $\mathsf{B}^O \neq \mathsf{FP}^O$.

Closure properties. One of the nice things about the definition of [GL25] is that it enforces nice closure properties (see Section 3.2). In particular, under their definition, if A, B, C are TFNP circuit problems and A reduces to B under black-box reductions, then A^{C} reduces to B^{C} (under FP^{C} reductions). We ask if such closure properties hold for our Definition 3.3.

1.3 Related work

[GL25] first introduced the concept of hierarchies within TFNP. Their work uses a stricter definition of TFNP oracle classes and hierarchies than we do (see Section 3.2). In particular, our definitions do not require that subwitnesses are internally consistent, which allows us to prove theorems of the form $\text{WeakPigeon}^{\text{Lossy}}$ reduces to WeakPigeon and lets us derive one of our key conditional collapses.

[GL25] considers the self-lowness of a TFNP problem A a desirable property, since it means that one can always assume the existence of oracle gates to A when reducing to A, which makes reductions to A much easier (and possibly even tractable for the first time). Here we take the opposite view: we are *pleased* to find that a TFNP problem A is *not* self-low since it lets us conclude that $\mathrm{A} \in \mathsf{FP}$ has the non-trivial consequence that $\mathrm{A}^{\mathrm{A}} \in \mathsf{FP}$.

Though [GL25] is the first work to give a systematic treatment of TFNP oracle classes, the idea of giving TFNP circuit problems oracle access to other TFNP problems dates back much further. [Kor22] showed that the problem of generating large primes is in $\text{Lossy}^{\text{Factor}}$ and [LLR24] implicitly used $\mathsf{LOSSY}^{\mathsf{PLS}}$ to study the complexity of proving lower bounds on resolution proofs. One can even view the fact that the set of languages low for NP is exactly $\mathsf{NP} \cap \mathsf{coNP}$ [KT+15] as the decision version analogue of [GL25]. Specifically, [GL25] showed that, for any TFNP circuit problem A and TFNP problem B, $\mathrm{A}^{\mathrm{B}} \in \mathsf{TFNP}$. So in some sense, TFNP (which is the search analogue of $\mathsf{NP} \cap \mathsf{coNP}$) is the set of search problems which are low for TFNP.

Our Section 5.1 should be seen as a translation of the non-black-box method of Buzek and Tessaro of constructing collision resistant hash functions (CRH) from multicollision resistant hash functions (MCRH) [BT24]. In fact, our translation is significantly easier to reason about. We are not concerned with average case complexity, which is important for cryptography. And our proof is completely black-box since we do not need to reason about the adversary finding collisions in f but only show a black-box reduction from the problem of finding multicollisions to $\text{WeakPigeon}^{\text{WeakPigeon}}$ and then argue that if WeakPigeon $\in \mathsf{FP}$, then the whole hierarchy collapses. In some sense, the whole non-black-box part of the reasoning in [BT24] is isolated to reasoning about what happens if the hierarchy collapses to FP and the reduction itself remains black-box.

The work of Buzek and Tessaro is itself partially inspired by a prior non-black-box construction of CRHs from MCRHs by Rothblum and Vasudevan [RV24] which uses similar ideas. [RV24] builds an infinitely often CRH from an MCRH which is originally 3-wise or 4-wise collision resistant and sufficiently compressing. [BT24] extended these ideas to build infinitely often CRHs from MCRHs which are k-wise collision resistant and compress by a $(1+\varepsilon)$ factor for any constant k. The ideas of Rothblum and Vasudevan were subsequently translated into the language of TFNP and conditional collapses in TFNP in [BGSD25], though not through the unifying language of hierarchies that we use here.

Finally, TFNP problem iteration and self-lowness are closely related to Turing-closure. Indeed, for any problem $\mathrm{A} \in \mathsf{TFNP}$, one can view its Turing-closure FP^{A} as TFNP iteration where the base class is FP. Furthermore, the fact that several well known TFNP subclasses are Turing-closed is crucial for several of our proofs [BJ12, LPT24] (see Lemma 2.16). Importantly, non-self-lowness of several TFNP classes can be inferred from known results about Turing-closure. A work by Fleming, Grosser, Pitassi, and Robere showed that PPP is not Turing-closed under black-box reductions [FGPR24], implying that $\text{Pigeon}^{\text{Pigeon}}$ does not reduce to Pigeon under black-box reductions. Later work by Hubáček [Hub26] showed that PWPP is not Turing-closed under black-box reductions, implying that $\text{WeakPigeon}^{\text{WeakPigeon}}$ does not reduce to WeakPigeon under black-box reductions. Interestingly, the NestedCollision problem that Hubáček uses to separate PWPP from $\mathsf{FP}^{\mathsf{PWPP}}$ bears a striking resemblance to $\text{WeakPigeon}^{\text{WeakPigeon}}$, though we do not know of any formal reduction.

2 Preliminaries

2.1 White-Box TFNP

2.1.1 Basics

We are interested in search problems, which are formally defined as relations. TFNP problems are total, efficiently verifiable relations.

Definition 2.1. *A search problem is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. We say y is a solution to R on input x if $(x,y) \in R$.*

Definition 2.2. *A relation R is in TFNP if the following conditions hold.*

1. ***Totality:** for all $x \in \{0,1\}^n$, there exists a $y \in \{0,1\}^{\mathrm{poly}(n)}$ such that $(x,y) \in R$.*
2. ***Verifiability:** there exists a polynomial time algorithm $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that $V(x,y) = 1$ iff $(x,y) \in R$.*

For $R, Q \in \mathsf{TFNP}$, a reduction from R to Q consists of two efficient functions f and g. f transforms the input to R into an input to Q, and g transforms any solution to a Q instance into a proposed solution to the original R instance.

Definition 2.3. *For $R, Q \in \mathsf{TFNP}$, a many-one reduction from R to Q consists of two polynomial time algorithms (f,g) such that for all $x \in \{0,1\}^n, o \in \{0,1\}^{\mathrm{poly}(n)}$*

$$(x, g(x,o)) \in R \Longleftarrow (f(x), o) \in Q.$$

Definition 2.4. *For $R, Q \in$ TFNP, a randomized many-one reduction from R to Q consists of two randomized polynomial time algorithms (f, g) such that, for all $x \in \{0,1\}^n$, with probability at least $1/2$, for all $o \in \{0,1\}^{\mathrm{poly}(n)}$,*

$$(x, g(x, o)) \in R \Longleftarrow (f(x), o) \in Q.$$

2.1.2 Classes

Many of the TFNP subclasses we will be concerned with are defined with respect to a complete problem which has some sequential structure on an implicitly defined exponentially large graph. Take Definition 2.5 for example. One should think of the input circuits P, S as defining the edge set of a directed graph G with 2^n vertices. In particular, G has a directed edge (a, b) if $S(a) = b$ and $P(b) = a$. By the fact that P, S have exactly one output, each vertex of G has at most one incoming and outgoing edge. Furthermore, $P(0) = 0$ implies that 0 has no incoming edge. Therefore, G has a line starting at 0. A solution to an EoL instance is the last vertex in a line (end of line), or the start of a new line. We will not cover the intuition behind all the TFNP classes we use here, but we encourage the reader to read the foundational works defining these classes. We note that many of the classes we use have a similar structure to PPAD in that their complete problem should be thought of as implicitly defining an exponentially sized graph.

Definition 2.5 ([Pap94])**.** *The problem* EndOfLine *(abbreviated* EoL*) is defined as follows. The input is circuits $P, S : \{0,1\}^n \to \{0,1\}^n$ such that $S(0) \neq 0$ and $P(0) = 0$. A solution is any $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$ or any non-zero $x \in \{0,1\}^n$ such that $S(P(x)) \neq x$.* PPAD *is defined as all* TFNP *problems which are many-one reducible to* EoL*.*

Definition 2.6. *The problem* SinkOfLine *(abbreviated* SoL*) is defined as follows. The input is circuits $P, S : \{0,1\}^n \to \{0,1\}^n$ such that $S(0) \neq 0$ and $P(0) = 0$. A solution is any $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$.* PPADS *is defined as all* TFNP *problems which are many-one reducible to* SoL*.*

Definition 2.7 ([JPY88])**.** *The problem* SinkOfDAG *(abbreviated* SoD*) is defined as follows. Given two circuits $S : \{0,1\}^n \to \{0,1\}^n$ such that $S(0) \neq 0$ and $V : \{0,1\}^n \to \{1, \ldots, 2^m - 1\}$, find $v \in \{0,1\}^n$ such that either 1) $v \neq S(v)$ but $S(v) = S(S(v))$ or 2) $V(S(v)) \leq V(v)$.* PLS *is defined as all* TFNP *problems which are many-one reducible to* SoD*.*

Definition 2.8 ([FGMS20])**.** *The problem* UEOPL *is defined as follows. Given Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$, and a Boolean circuit $V : \{0,1\}^n \to \{0, 1, \ldots, 2^m - 1\}$ such that $V(0^n) = 0$, find one of the following:*

(U1) A vertex $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$.

(UV1) A vertex $x \in \{0,1\}^n$ such that $x \neq S(x)$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$.

(UV2) A vertex $x \in \{0,1\}^n$ such that $S(P(x)) \neq x \neq 0^n$.

(UV3) Two vertices $x, y \in \{0,1\}^n$ such that $x \neq y$, $x \neq S(x)$, $y \neq S(y)$, and either $V(x) = V(y)$ or $V(x) < V(y) < V(S(x))$.

UEOPL *is defined as all* TFNP *problems many-one reducible to* UEOPL*.*

In Definition 2.8, U1 is the type of solution in which we are interested, and all other solutions are violations that demonstrate the instance in question is not an EoPL (EndOfPotentialLine) instance which has exactly one line, along which the value function V is strictly increasing. We will mostly be interested in the case where that promise is satisfied, for which [FGMS20] introduce the class promiseUEOPL.

Definition 2.9 ([FGMS20])**.** promiseUEOPL *is defined to be the promise version of* UEOPL *in which it is promised that there is a unique line in the graph defined by P, S, and* promiseUEOPL *to be the class of promise problems that can be reduced in polynomial time to* promiseUEOPL*.*

Definition 2.10 ([DP11])**.** *The problem* ContinuousLocalOpt *(abbreviated* CLO*) is defined as follows. Given two arithmetic circuits computing functions $f : [0,1]^3 \to [0,1]^3$ and $p : [0,1]^3 \to [0,1]$, and constants $\varepsilon, \lambda > 0$, find a point x such that $p(f(x)) \geq p(x) - \varepsilon$, or two points that violate the λ-continuity of either f or p.* CLS *is defined as all* TFNP *problems which are many-one reducible to* CLO*.*

The totality of the following three related problems follows from the pigeonhole principle.

Definition 2.11 ([JLRX24, Jeř16])**.** *For any positive integer n, let $t(n)$, $M(n)$, and $N(n)$ be integer parameters satisfying $M > (t-1)N$. The problem* t-Pigeon(M, N) *is defined as follows. Given as input n in unary and a* poly(n) *size circuit $C : [M] \to [N]$, output distinct $x_1, \ldots, x_t \in [M]$ such that $C(x_1) = \cdots = C(x_t)$*[1]*.* WeakPigeon *is defined as* 2-Pigeon$(2^n, 2^{n-1})$*, and* PWPP *is defined as all* TFNP *problems which are many-one reducible to* WeakPigeon*.*

Definition 2.12 ([Pap94])**.** *The problem* Pigeon *is defined as follows. Given a circuit $C : \{0,1\}^n \to \{0,1\}^n$, output either 1) x such that $C(x) = 0$ or 2) distinct x_1, x_2 such that $C(x_1) = C(x_2)$.* PPP *is defined as all* TFNP *problems which are many-one reducible to* Pigeon*.*

Definition 2.13 ([Kor22])**.** *Let $f(n) < n$ be an efficiently computable function. $f(n)$-*Lossy *is defined as follows. Given two circuits $C : \{0,1\}^n \to \{0,1\}^{f(n)}$ and $D : \{0,1\}^{f(n)} \to \{0,1\}^n$, output $x \in \{0,1\}^n$ such that $D(C(x)) \neq x$.* Lossy *is defined as* $(n-1)$-Lossy *and* LOSSY *is defined as all* TFNP *problems which are many-one reducible to* Lossy*.*

Definition 2.14. *Let* A *and* B *be* TFNP *problems.* Either(A, B) *is defined as follows. Given (a, b) where a is an instance of* A *and b is an instance of* B*, output y such that either y is a solution to a or y is a solution to b.*

It is not too hard to see that if A, B are complete for TFNP classes A, B, respectively, then Either(A, B) is complete for $\mathsf{A} \cap \mathsf{B}$. Finally, we define TFZPP, which is the total function analogue of ZPP.

Definition 2.15. TFZPP *is the class of* TFNP *relations R such that there exists a randomized, polynomial time algorithm $\mathcal{A}(\cdot)$ where for all $x \in \{0,1\}^n$, $(x, \mathcal{A}(x)) \in R$ with probability at least $1/2$ (over the coins of $\mathcal{A}$).*

2.1.3 Some useful reductions

We say that a TFNP class A is Turing-closed if a Turing reduction to A implies a many-one reduction to A.

Lemma 2.16 ([BJ12])**.** PPA*,* PPAD*,* PPADS*, and* PLS *are Turing-closed.*

Lemma 2.17 ([LPT24])**.** LOSSY *is Turing-closed.*

We note that the compression factor used to define LOSSY is mostly irrelevant.

Lemma 2.18 ([Kor22])**.** *Fix a constant $\varepsilon \in (0,1]$ and let $n^\varepsilon \leq f(n) < n$ be any efficiently computable function. $f(n)$-*Lossy *is equivalent to* Lossy *under many-one reductions.*

2.2 Black-Box TFNP

Definition 2.19. *A total search problem in the decision tree model is a sequence of relations $R^{\mathrm{dt}} = \{R_N \subseteq \{0,1\}^N \times O_N\}$, where O_N is a finite set of solutions, such that for every input $x \in \{0,1\}^N$, there exists a solution $o \in O_N$ such that $(x, o) \in R_N$.*

A total search problem R^{dt} belongs to $\mathsf{TFNP}^{\mathrm{dt}}$ *if, for each solution $o \in O_N$, there is a decision tree V_o of depth* poly$(\log N)$ *such that for every input $x \in \{0,1\}^N$, $V_o(x) = 1$ iff $(x, o) \in R_N$.*

[1] We are using different notation from both [JLRX24] and [BGSD25] for multicollision problems as we wish to reserve the use of a superscript to indicate TFNP problem iteration.

Definition 2.20. *Let $M = 2^{\mathrm{polylog}(N)}$. A decision tree reduction from a relation $R^{\mathrm{dt}} = \{R_N \subseteq \{0,1\}^N \times O_N\}$ to $Q^{\mathrm{dt}} = \{Q_M \subseteq \{0,1\}^M \times O'_M\}$ is a set of* polylog(N)*-depth decision trees $f_i : \{0,1\}^N \to \{0,1\}$ for each $i \in [M]$ and $g_o : \{0,1\}^N \to O_N$ for each $o \in O'_M$ such that for each $x \in \{0,1\}^N$,*

$$(x, g_o(x)) \in R \Longleftarrow (f(x), o) \in Q.$$

We identify all TFNP problems defined in Section 2.1.2 with their natural black-box analogues and note that almost all reductions that work in the white-box setting also work in the black-box setting.

3 Defining our hierarchies

To define hierarchies of TFNP problems, we must first define what it means for a circuit TFNP problem, A, to be given access to an oracle for a search problem O. The reader will hopefully find our definition (which matches that of [GL25] up to the difference noted in Section 3.2) natural: we simply allow the input circuit to A to have O oracle gates.

3.1 Hierarchies within TFNP

3.1.1 White-box hierarchies

When defining A^O, we will require that A is a circuit problem. Intuitively, this means that the input has as one of its parts a circuit C. Actually, some problems like SoD have multiple circuits, but for the sake of simplicity, we treat each such problem a taking in one circuit which is the concatenation of all of its circuit inputs.

Definition 3.1. *We say that a TFNP problem $R \subseteq \{0,1\}^n \times \{0,1\}^{\mathrm{poly}(n)}$ is a circuit problem if its input is a* poly(n) *size circuit $C : \{0,1\}^n \to \{0,1\}^{\mathrm{poly}(n)}$ along with an optional input $a \in \{0,1\}^{\mathrm{poly}(n)}$, and any potential solution can be verified by making oracle queries to C. We assume without loss of generality that R takes as input only one circuit by thinking of any TFNP problem which takes in multiple circuits $C_1, \ldots, C_t$ as one which only takes in one circuit $C' = C_1 || \ldots || C_t$ where the ith block of n input bits and the ith block of* poly(n) *output bits represent the behavior of C_i.*

C^{O} should *not* be viewed as a function since the behavior of a circuit C^{O} is not well-defined when O is a search problem with multiple possible outputs. In particular, $C^{\mathrm{O}}(x)$ depends on what subwitnesses are used to answer the O oracle gate queries made while evaluating $C^{\mathrm{O}}(x)$. To rectify this, for every C^{O}, we define a *simulation* circuit, C_{Eval} [2], for C^{O}. C_{Eval} takes as input x plus auxiliary inputs that serve as solutions to C^{O}'s oracle gates.

The auxiliary inputs allow C_{Eval} to simulate C^{O} on x. A minor technical detail we will mostly elide is that by the nature of C^O, each oracle gate has a fixed input/output length which must be known when C^O is constructed. TFNP problems generally do not have a fixed input/output length (in particular, the output length can depend on the input). We solve this discrepancy by padding all inputs and outputs so they have the appropriate bit-lengths (see [GL25] for a detailed discussion of this subtlety). We will also fix a canonical gate evaluation order in any circuit C so that we can safely say something like "gate i of C^O".

Definition 3.2. *Let $C^{\mathrm{O}} : \{0,1\}^n \to \{0,1\}^m$ be an oracle circuit where* O *is a TFNP problem, C^{O} contains t oracle gates, and the i^{th} oracle gate of C^{O} has an s_i-bit output. We now define the simulation circuit C_{Eval} for C^{O}:*

$$C_{\mathrm{Eval}} : \{0,1\}^n \times \prod_{i=1}^{t} \{0,1\}^{s_i} \to \{0,1\}^m \cup \{\bot\} \ .$$

[2] [GL25] refer to this as C^*, but to avoid conflicting with the $*$ used to indicate hierarchy in A^*, we prefer C_{Eval}.

C_{Eval} on input $(x, w_1, \ldots, w_t)$ simulates the evaluation of C^{O} on x. At the i^{th} instance when C^{O} must evaluate an oracle gate, say on input u, we check if w_i is a valid solution to the i^{th} oracle call to O on input u. If it is not, then $C_{\text{Eval}}(x, w_1, \ldots, w_t) = \perp$ and the simulation terminates; otherwise, the simulation of C^{O} continues by using w_i as the output of oracle gate i. If the simulation does not terminate prematurely and obtains a simulated m-bit output of $C^{\text{O}}(x)$, $C_{\text{Eval}}(x, w_1, \ldots, w_t)$ returns that output.

Now that we know that providing answers to gate evaluations (what we call subwitnesses) is enough to fix the evaluation of the circuit C^{O} on some input x, the definition of a solution to A^{B} follows naturally. It simply involves lifting the definition of a solution to A to include the subwitnesses needed in the evaluations of C^{O} to confirm a solution to A. Crucially, Definition 3.3 is different from Definition 3.7 (which is used in [GL25]) in that it does not enforce a property they call internal consistency (see Section 3.2 for a discussion of this fact).

Definition 3.3. *Let A and B be TFNP problems and A be a circuit problem with a black-box verifier $V : \{0,1\}^n \times \{0,1\}^{\text{poly}(n)} \to \{0,1\}$, one which only queries the circuit input to A in a black-box manner. The input to A^{B} is defined the same as A except the circuit input C^{B} of A contains t B oracle gates. The verifier for A^{B}, $V' : \{0,1\}^n \times \{0,1\}^{\text{poly}(n)} \to \{0,1\}$, is defined as follows. V' takes as input $(C^{\text{B}}, a), (y, w_{1,1}, \ldots, w_{\text{poly}(n),t})$. V' simulates the computation of $V((C^{\text{B}}, a), y)$ and on the i^{th} evaluation of C^{B}, say on input x, the black-box verifier V' receives $C_{\text{Eval}}(x, w_{i,1}, w_{i,2}, \ldots, w_{i,t})$. If, at any point of the simulation, $C_{\text{Eval}}(x, w_{i,1}, w_{i,2}, \ldots, w_{i,t}) = \perp$, V' outputs 0. Otherwise, V' outputs the result of V after the simulation of V terminates.*

If the reader wishes to see a concrete instantiation of Definition 3.3, they should now skip to Section 3.1.2. It is not too hard to see that if $\text{A}, \text{B} \in \mathsf{TFNP}$, then A^{B} is itself a TFNP problem. We can then iterate this construction to obtain hierarchies of problems within TFNP.

Definition 3.4. *Let A be a TFNP circuit problem. We define $\text{A}^1 = \text{A}$ and $\text{A}^{i+1} = \text{A}^{(\text{A}^i)}$ (for convenience we will also let A^0 be some FP-complete problem such as the circuit evaluation problem). A^* is defined as the class of all TFNP problems which have a many-one reduction to A^i for some fixed nonnegative integer i. We say that A is self-low if A^{A} has a many-one reduction to A.*

We use the notation A^* rather than A^* to encourage the reader to note the choice of complete problem may matter. In particular, under our definition, even if A and B are equivalent, it is not clear that A^* and B^* are the same [3] ([GL25] use the stricter Definition 3.7 to force these to be equivalent under sufficiently strong reductions).

We will make use of the following natural lemma.

Lemma 3.5 ([GL25])**.** *Let A be a TFNP circuit problem and let B_1, B_2 be TFNP problems. If B_1 is (non-uniformly) many-one reducible to B_2, then A^{B_1} has a (non-uniform) many-one reduction to A^{B_2} that preserves the number of oracle gates [4].*

3.1.2 An example

Definition 3.3 may be hard to digest. As such, we will give an example of a $\text{WeakPigeon}^{\text{Bertrand}}$ instance. Here Bertrand is the following problem: given as input 1^n with $n > 1$, output a prime p such that $2^{n-1} < p < 2^n$. The totality of Bertrand follows from Bertrand's postulate. Consider the example family of $\text{WeakPigeon}^{\text{Bertrand}}$ instances in Figure 1.

[3] We make no claim that they are not the same; rather, we only note that we find no easy proof of such a closure property.

[4] Technically, [GL25] showed that A^{B_1} reduces to A^{B_2} under Definition 3.7 of A^{B} and only for uniform reductions. We observe that the same proof goes through under Definition 3.3, goes through for non-uniform reductions, and also preserves the number of oracle gates. The reduction is quite simple: replace each B_1 gate in the input to A^{B_1} with the B_2 oracle circuit which simulates the reduction from B_1 to B_2.

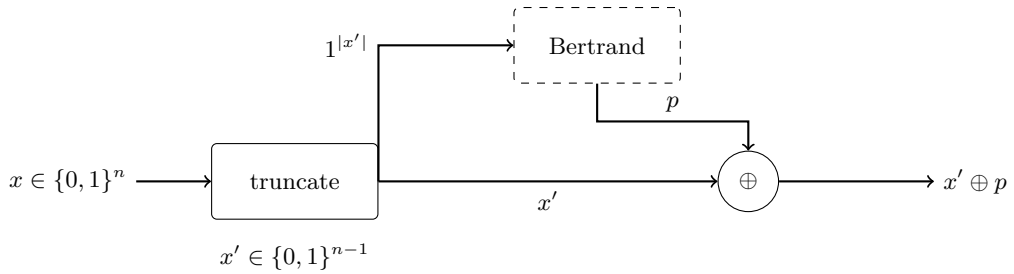


Figure 1: An example family of WeakPigeon$^{\text{Bertrand}}$ instances. An input $x \in \{0,1\}^n$ is first truncated to compute $x' = x_{[1,n-1]}$. We then feed $1^{|x'|} = 1^{n-1}$ to our Bertrand gate to get back an $n-1$ bit prime p, and then output $p \oplus x'$.

Let C^{Bertrand} be the circuit from Figure 1 where $n = 4$. We will also freely switch between bit-level evaluation of C^{Bertrand} and treating those values as integers with the confidence that the reader can perform these translations for themselves. Let us begin by probing C_{Eval} on a few examples to see what it does. As previously noted, C^{Bertrand} as currently defined is not a well-defined function since its output depends on the answer provided by its Bertrand oracle gate. In particular, $C^{\text{Bertrand}}(1111)$ could be either $101 \oplus 111 = 010$ in the case that $p = 5$ or $111 \oplus 111 = 000$ in the case that $p = 7$. As defined in Definition 3.2, C_{Eval} lets us lock in one of these evaluations since it takes (in this case) both x and p as input. In particular, $C_{\text{Eval}}(1111, 5) = 010$ and $C_{\text{Eval}}(1111, 7) = 000$. In the case that the candidate prime p provided as a subwitness to $C_{\text{Eval}}(x, p)$ is not a valid answer to the Bertrand oracle query 1^{n-1}, $C_{\text{Eval}}(x, p) = \perp$. For example, $C_{\text{Eval}}(1111, 6) = \perp$ since 6 is not prime.

Now, let us consider what an answer to WeakPigeon$^{\text{Bertrand}}$ looks like under Definition 3.3. Essentially, Definition 3.3 says that it has the same form as an answer to WeakPigeon but also comes with the subwitnesses needed to evaluate an answer. An answer to WeakPigeon consists of distinct, colliding $x, x' \in \{0,1\}^n$. Therefore, an answer to the C^{Bertrand} instance in Figure 1 consists of $x, x' \in \{0,1\}^n$ and $p, p' \in \{0,1\}^{n-1}$ where x, x' form the collision in C^{Bertrand}, p is a subwitness that lets us evaluate $C^{\text{Bertrand}}(x)$ and p' is a subwitness that lets us evaluate $C^{\text{Bertrand}}(x')$. Formally, a solution consists of distinct $x, x' \in \{0,1\}^n$ and $p, p' \in \{0,1\}^{n-1}$ such that $C_{\text{Eval}}(x, p) = C_{\text{Eval}}(x', p') \neq \perp$. For example, $x = 1111, x' = 1011$ and $p = 5, p' = 7$ is a solution to WeakPigeon$^{\text{Bertrand}}$ on C^{Bertrand} since $C_{\text{Eval}}(1111, 5) = 010 = C_{\text{Eval}}(1011, 7)$.

3.1.3 Unrolling oracle circuits

The following shift of perspective will be helpful in understanding TFNP problem iteration and will prove crucial to defining iteration in the black-box setting (and therefore for our black-box separation in Section 6). When B is itself a circuit problem, instead of viewing the input to A^{B} as a single circuit with oracle gates, we can equivalently view it as several circuits. Consider a simple example where the input to WeakPigeon$^{\text{B}}$ is $C^{\text{B}} : \{0,1\}^n \to \{0,1\}^{n-1}$ which has two oracle gates. Let us use D_x to denote the circuit queried to the first B oracle gate when C^{B} is queried on x, $D_{x,o}$ to denote the circuit queried to the second B oracle gate when C^{B} is queried on x and the first oracle gate outputs o as its proposed solution to B on D_x. Note that there exists an efficient algorithm to compute $D_x(y)$ given x, y, to compute $D_{x,o}(y)$ given x, o, y, and to compute $C_{\text{Eval}}(x, o_1, o_2)$ given x, o_1, o_2. Using Definition 3.3, one can therefore rephrase solving WeakPigeon$^{\text{B}}$ as finding distinct x, x' and $(o_1, o_2), (o_1', o_2')$ such that the following hold.

1. o_1 is a solution to B on D_x and o_2 is a solution to B on D_{x,o_1}
2. o_1' is a solution to B on $D_{x'}'$ and o_2' is a solution to B on $D_{x',o_1'}$
3. $C_{\text{Eval}}(x, o_1, o_2) = C_{\text{Eval}}(x', o_1', o_2')$

This equivalent definition abstracts away the internal logic of C^{B} so that we can now reason simply at the level of circuits. We have designed an alternative formulation of A^{B} such that its input does not have oracle gates but rather unfolds the behavior of those oracle gates into multiple circuits. Let us consider what this looks like for WeakPigeon^2.

Definition 3.6. WeakPigeon^2 *is defined as follows. We are given as input* n *circuits* $C_1, \ldots, C_n$*, where*

$$C_i : \{0,1\}^n \times \prod_{j=1}^{i-1} (\{0,1\}^n \times \{0,1\}^n) \times \{0,1\}^n \to \{0,1\}^{n-1} \ ,$$

along with a final circuit,

$$C_{last} : \{0,1\}^n \times \prod_{j=1}^{n} (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^{n-1} \ .$$

The goal is to output $x, (t_1, t_1'), \ldots, (t_n, t_n')$ *where* $x, t_i, t_i' \in \{0,1\}^n$*, and* $y, (u_1, u_1'), \ldots, (u_n, u_n')$ *where* $y, u_i, u_i' \in \{0,1\}^n$ *such that the following hold.*

1. *For all* $i \in [1, n]$*,* $t_i \neq t_i'$ *and* $C_i(x, t_1, t_1', \ldots, t_{i-1}, t_{i-1}', t_i) = C_i(x, t_1, t_1', \ldots, t_{i-1}, t_{i-1}', t_i')$*.*
2. *For all* $i \in [1, n]$*,* $u_i \neq u_i'$ *and* $C_i(y, u_1, u_1', \ldots, u_{i-1}, u_{i-1}', u_i) = C_i(y, u_1, u_1', \ldots, u_{i-1}, u_{i-1}', u_i')$*.*
3. $x \neq y$ *and* $C_{last}(x, t_1, t_1', \ldots, t_n, t_n') = C_{last}(y, u_1, u_1', \ldots, u_n, u_n')$

Definition 3.6 is the unrolled version of some WeakPigeon^2 instance $T^{\text{WeakPigeon}}$ which has an n bit input, an $n-1$ bit output, and n oracle gates, where each oracle gate is queried on a circuit which has an n bit input and an $n-1$ bit output. We will justify these assumptions shortly, but we wish to first analyze the syntax of Definition 3.6. C_i represents oracle gate i of $T^{\text{WeakPigeon}}$. In particular, $C_i(x, o_1, \ldots, o_{i-1}, \cdot)$ is the WeakPigeon instance queried by oracle gate i assuming that the input to $T^{\text{WeakPigeon}}$ was x and oracle gate queries $1, 2, \ldots, i-1$ were answered using solutions $o_1, \ldots, o_{i-1}$, respectively. C_{last} should be thought of as T_{Eval}.

So why are we free to assume that the underlying WeakPigeon^2 instance we unroll has n oracle gates, where each oracle gate itself is queried on a circuit which has an n bit input and an $n-1$ bit output? This is without loss of generality since one can always add dummy gates to force each circuit to have n oracle gates and one can also always increase the input/output length of a WeakPigeon$^{\text{i}}$ instance by the following standard trick. Let $n > m$. If we have a WeakPigeon instance $J : \{0,1\}^m \to \{0,1\}^{m-1}$, solving this instance can be reduced to solving WeakPigeon on $K : \{0,1\}^n \to \{0,1\}^{n-1}$ where $K(x) = J(x_{[1,m]}) \circ x_{[m+1,n]}$ (we note without proof that this padding trick relativizes and therefore also works for WeakPigeon$^{\text{B}}$ for any TFNP problem B). Such tricks let us assume that each WeakPigeon instance has the same input length and same number of oracle gates, which will drastically simplify our proofs and avoid the cumbersome tracking of input lengths at each nested oracle call.

Having justified the assumptions in Definition 3.6, we now examine what a solution looks like. One should think of x and y as forming a collision in the overall circuit C_{last}, $t_1, t_1', \ldots, t_n, t_n'$ as the subwitnesses required to evaluate the overall circuit on x, and $u_1, u_1', \ldots, u_n, u_n'$ as the subwitnesses required to evaluate the overall circuit on y. Item 1 guarantees that the subwitnesses $t_1, t_1', \ldots, t_n, t_n'$ are valid. Item 2 guarantees that the subwitnesses $u_1, u_1', \ldots, u_n, u_n'$ are valid. Item 3 guarantees that x and y form a solution to the whole collision-finding problem.

One advantage of the unrolled formulation of WeakPigeon^2 in Definition 3.6 as opposed to the oracle circuit definition is that Definition 3.6 has a natural black-box formulation: a string $I \in \{0,1\}^N$ encoding the truth tables of $C_1, \ldots, C_n, C_{\text{last}}$. In Section 6, we define the black-box WeakPigeon hierarchy rather than just the black-box analogue of WeakPigeon^2. Our definition is based on the same intuition that the oracle circuit definition of TFNP problem iteration is equivalent to the unrolled definition of Definition 3.6, and the latter has a natural black-box formulation.

3.2 Hierarchies and consistent subwitnesses

It is important to note at this point that our definitions of hierarchies are subtly different than those in [GL25]. In particular, Ghentiyala and Li define A$^{\text{B}}$ as follows.

Definition 3.7. *Let* A *and* B *be* TFNP *problems and* A *be a circuit problem with a black-box verifier* $V : \{0,1\}^n \times \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}$*, one which only queries the circuit input to* A *in a black-box manner. The input to* $\mathrm{A}^{\mathrm{B}}_{\checkmark}$ *is defined the same as* A *except the circuit input* C^{B} *of* A *contains* t B *oracle gates. The verifier for* $\mathrm{A}^{\mathrm{B}}_{\checkmark}$*,* $V' : \{0,1\}^n \times \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}$*, is defined as follows.* V' *takes as input* $(C^{\mathrm{B}}, a), (y, w_{1,1}, \ldots, w_{\mathrm{poly}(n),t})$*.* V' *simulates the computation of* $V((C^{\mathrm{B}}, a), y)$ *and on the* i^{th} *evaluation of* C^{B}*, say on input* x*, the black-box verifier* V' *receives* $C_{\mathrm{Eval}}(x, w_{i,1}, w_{i,2}, \ldots, w_{i,t})$*. If, at any point,* $C_{\mathrm{Eval}}(x, w_{i,1}, w_{i,2}, \ldots, w_{i,t}) = \perp$ *or* V' *finds the sub-witnesses* $w_{i,j}$ *to not be* internally consistent *(which we define below),* V' *outputs* 0*. Otherwise,* V' *outputs the result of* V *after the simulation of* V *terminates.*

We say that the $w_{1,1}, \ldots, w_{\mathrm{poly}(n),t}$ *are not* internally consistent *(with respect to the simulation) if, during the execution of* V'*, there exist two oracle queries made on the same input* u *and the simulation uses answer* $w_{a,b}$ *to the first oracle gate query and* $w_{a',b'}$ *as the answer to the second oracle gate query and* $w_{a,b} \neq w_{a',b'}$*.*

Definition 3.8. *Let* A *be* TFNP *class whose canonical complete problem is the circuit problem* A*.* $\mathsf{A}^{\mathrm{A}}_{\checkmark}$ *is defined to be the* TFNP *subclass with complete problem* $\mathrm{A}^{\mathrm{A}}_{\checkmark}$*. Furthermore, for any positive integer* i*, define* $\mathrm{A}^i_{\checkmark}$ *so that* $\mathrm{A}^1_{\checkmark} = \mathrm{A}$ *and* $\mathrm{A}^{i+1}_{\checkmark} = \mathrm{A}^{(\mathrm{A}^i_{\checkmark})}_{\checkmark}$ *(for convenience, we let* $\mathrm{A}^0_{\checkmark}$ *be some* FP*-complete problem).* A^i *is the class of* TFNP *problems with a many-one reduction to* $\mathrm{A}^i_{\checkmark}$*.*

Note the crucial difference of enforcing the condition they refer to as internal consistency. To differentiate our Definition 3.3 from the definition of [GL25] (which uses the same notation), we use a ✓ subscript (denoting the consistency check) whenever we are referring to the definition given in [GL25].

Note also that a solution to A^{B} is not always a solution to $\mathrm{A}^{\mathrm{B}}{}_{\checkmark}$. For example, the solution $x = 1111, x' = 1011$ and $p = 5, p' = 7$ in Section 3.1.2 is not internally consistent. In some sense, for any problem A^{B}, internal consistency is asserting the existence of a function oracle $O : \{0,1\}^* \to \{0,1\}^*$ where $O(x)$ is a solution to B on input instance x, such that the answer y provided in Definition 3.7 is valid relative to O. The key advantage of Definition 3.7 is that it enforces that if A has a black-box many-one reduction to A', then for any B, $\mathrm{A}^{\mathrm{B}}_{\checkmark}$ has a black-box many-one reduction to $\mathrm{A}'^{\mathrm{B}}_{\checkmark}$ under FP^{B} reductions [GL25]. We do not know whether the nice closure properties that hold for Definition 3.7 hold for Definition 3.3.

In hindsight, Definition 3.3 is perhaps more natural than Definition 3.7. And while ensuring consistency of solutions ensures such closure properties and asserts the existence of a fixed oracle O relative to which the provided answer is valid, there is no intrinsic justification for it. TFNP problem iteration remains perfectly well defined and often easier to reason about under our relaxed definition. In particular, our proof in Section 5.3 (which uses the Turing closure of LOSSY to assume that each oracle circuit has exactly one Lossy gate) seems to depend on our more relaxed definition. We also find it interesting that Theorem 5.1 does not require the more restrictive Definition 3.7. Ultimately, we believe that our more relaxed definition is natural and merits study in its own right. However, we leave it as an open question in Section 1.2 whether the desired closure properties of Definition 3.7 hold or almost hold under our relaxed definition.

4 Conditional collapses

4.1 Uniform, non-uniform, and randomized collapses

Much like how a collapse at any level of the polynomial hierarchy (e.g. $\Sigma^{\mathsf{P}}_2 = \mathsf{NP}$) implies collapses at all higher levels (if $\Sigma^{\mathsf{P}}_2 = \mathsf{NP}$, then $\mathsf{PH} = \mathsf{NP}$), a collapse at some level i of some TFNP hierarchy implies a collapse at all levels above i.

Observation 4.1. *Let* A *be a* TFNP *circuit problem and* i *be any nonnegative integer. If* A^{i+1} *reduces to* A^i *under (non-uniform) many-one reductions, then for any* $j \geq i$*,* A^j *reduces to* A^i *under (non-uniform) many-one reductions.*

Proof. We will show the result for uniform many-one reductions by induction on j (the case for non-uniform many-one reductions follows using the exact same argument). The base case of $i = j$ trivially holds. We now show the inductive case. The input to A^{j+1} consists of a circuit $C^{A^j} : \{0,1\}^n \to \{0,1\}^{\mathrm{poly}(n)}$ and some input $a \in \{0,1\}^{\mathrm{poly}(n)}$. We now show a reduction from A^{j+1} to A^i. Note that $A^{j+1} = A^{A^j}$ has a

many-one reduction to $A^{A^i} = A^{i+1}$ by the inductive hypothesis and Lemma 3.5. By assumption, A^{i+1} has a many-one reduction to A^i. Chaining these two reductions, we obtain a many-one reduction from A^{j+1} to A^i, as desired. □

Interestingly, such conditional collapses hold for randomized reductions as well. To show this, we first extend Lemma 3.5 to randomized algorithms. The result will then follow easily as a corollary.

Lemma 4.2. *Let* A *be a* TFNP *circuit problem,* B, B' *be* TFNP *problems, and* $c \in \mathbb{N}$. *If* B *has a randomized many-one reduction to* B'*, then* A^{B} *has a randomized many-one reduction to* $\mathrm{A}^{\mathrm{B}'}$ *which succeeds with probability* $1 - 2^{-n^c}$ *where* n *is the size of the input instance to* A^{B}.

Proof. Let $\mathrm{T}_{\mathrm{B}'}$ be the TFNP problem denoting the Turing-closure of B'. In particular, $\mathrm{T}_{\mathrm{B}'}$ takes as input $C^{\mathrm{B}'} : \{0,1\}^n \to \{0,1\}^{\mathrm{poly}(n)}, x \in \{0,1\}^n$. A valid solution consists of $w_1, \ldots, w_{\mathrm{poly}(n)} \in \{0,1\}^{\mathrm{poly}(n)}$ such that $C^{\mathrm{B}'}_{\mathrm{Eval}}(x, w_1, \ldots, w_{\mathrm{poly}(n)}) \neq \bot$.

We will show that A^{B} has a randomized many-one reduction to $\mathrm{A}^{\mathrm{B}'}$. The description length of any B oracle gate in C^{B} is bounded by some $m = \mathrm{poly}(n)$ where n is the size of the circuit input to A^{B}. We assume without loss of generality that all oracle gates in C^{B} have the same description length m. Say that the reduction from B to B' uses $p = \mathrm{poly}(m)$ bits of randomness. Let c' be some sufficiently large constant (depending on c). Consider sampling $p \cdot m^{c'}$ random bits r, simulating the randomized reduction from B to B' $m^{c'}$ times (using the ith block of p bits of r for simulation i), and outputting any answer which solves the original B instance if one is found among the $m^{c'}$ repetitions (and otherwise outputs $\bot$). Observe that this specifies a randomized many-one reduction from B to $\mathrm{T}_{\mathrm{B}'}$ with randomness r. We say r is valid advice if such a reduction succeeds for all B instances of description length at most m. A simple union bound shows that r is not valid advice with probability at most 2^{-n^c} for sufficiently large n (since c' was chosen to be sufficiently large).

The randomized reduction from A^{B} to $\mathrm{A}^{\mathrm{T}_{\mathrm{B}'}}$ simulates the non-uniform reduction from A^{B} to $\mathrm{A}^{\mathrm{T}_{\mathrm{B}'}}$ from Lemma 3.5 using a random string $r \in \{0,1\}^{p \cdot m^{c'}}$ as advice. This is clearly a polynomial time reduction. Note that as long as r is valid advice, the reduction will succeed. Since r is valid advice with probability at least $1 - 2^{-n^c}$, our proposed reduction succeeds with probability at least $1 - 2^{-n^c}$. Finally, $\mathrm{A}^{\mathrm{T}_{\mathrm{B}'}}$ trivially reduces to $\mathrm{A}^{\mathrm{B}'}$. Therefore, A^{B} has a randomized reduction to $\mathrm{A}^{\mathrm{B}'}$.

□

Corollary 4.3. *Let* A *be a* TFNP *circuit problem and* i *be any nonnegative integer. If* A^{i+1} *reduces to* A^i *under randomized many-one reductions, then for any* $j \geq i$, A^j *reduces to* A^i *under randomized many-one reductions.*

Proof. We will show this by induction on j. The base case of $i = j$ holds trivially. We now show the inductive case. Let c be a constant such that the witness length of an answer to A^{j+1} is bounded by m^c when the input length is m. We will assume without loss of generality that all witnesses are of length m^c. The input to A^{j+1} consists of a circuit $C^{A^j} : \{0,1\}^n \to \{0,1\}^{\mathrm{poly}(n)}$ and some input $a \in \{0,1\}^{\mathrm{poly}(n)}$. Let m denote the size of the input (C^{A^j}, a), thereby bounding a solution to this instance by m^c. $A^{j+1} = A^{A^j}$ has a randomized many-one reduction to $A^{A^i} = A^{i+1}$ which succeeds with probability at least $1 - 2^{-m^c}$ by the inductive hypothesis and Lemma 4.2. A^{i+1} has a randomized many-one reduction to A^i which succeeds with probability at least $1/2$ by assumption. Our reduction will chain these two reductions to obtain a randomized many-one reduction from A^{j+1} to A^i which succeeds with some probability $p \geq \frac{1}{2}(1 - 2^{-m^c})$. If this reduction fails to output a valid answer to A^{j+1} on input (C^{A^j}, a), the reduction tries a random witness w of length m^c as a solution to A^{j+1} on input (C^{A^j}, a) and outputs it if it is valid (and outputs $\bot$ otherwise).

This reduction clearly works in polynomial time. The probability of success of the reduction is p plus $(1-p)$ times the probability that a uniformly random solution solves (C^{A^j}, a) (which is at least 2^{-m^c} since at least one witness must solve the problem). It is not hard to see that $p + (1-p)2^{-m^c} \geq 1/2$. We have used the fact that a random witness works with sufficient probability to boost the probability of success up to $1/2$, an idea already present in [Jeř16]. Therefore, the reduction succeeds with probability at least $1/2$, as desired. □

4.2 Conditional collapse of the $\mathrm{Either}(\mathrm{A},\mathrm{B})$ hierarchy

TFNP oracle problems A^{B} can be seen as an extension of the $\mathrm{Either}(\mathrm{A},\mathrm{B})$ problem. Both should be seen as operators which allow us to construct a new TFNP problem from existing TFNP problems. However, these operators are, in some sense, going in opposite directions. A^{B} is at least as powerful as the most powerful of A and B, while $\mathrm{Either}(\mathrm{A},\mathrm{B})$ is at most as powerful as the least powerful of A and B. It is therefore natural to consider how these two notions—one which gives us more power and another which gives us less—interact. From a more practical standpoint, it is worth exploring the structure of Either hierarchies since many important TFNP problems belong to several TFNP subclasses (most famously, $\mathsf{CLS} = \mathsf{PPAD} \cap \mathsf{PLS}$ [FGHS22]).

Before we show some conditional collapse, let us begin by recalling the structure of the problem $\mathrm{Either}(\mathrm{A},\mathrm{B})$. If A and B are circuit problems, then $\mathrm{Either}(\mathrm{A},\mathrm{B})$ is a circuit problem which takes in one circuit C as input where the first n input bits and first $\mathrm{poly}(n)$ output bits represent the input/output behavior of the A part of the input and the next n input bits and next $\mathrm{poly}(n)$ output bits represent the input/output behavior of the B part of the input (see Definition 3.1). In what follows, we will assume without loss of generality that $\mathrm{Either}(\mathrm{A},\mathrm{B})$ requires that the two component circuits represented in the input to $\mathrm{Either}(\mathrm{A},\mathrm{B})$ have the same number of input bits.

We now show the following rather intuitive lemma that $\mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{C}}$ can be reduced to $\mathrm{Either}(\mathrm{A}^{\mathrm{C}},\mathrm{B}^{\mathrm{C}})$.

Lemma 4.4. *Let* $\mathrm{A},\mathrm{B},\mathrm{C}$ *be* TFNP *circuit problems.* $\mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{C}}$ *has a many-one reduction to* $\mathrm{Either}(\mathrm{A}^{\mathrm{C}},\mathrm{B}^{\mathrm{C}})$.

Proof. The input to the reduction is a circuit D^{C} where the first n input bits and first $\mathrm{poly}(n)$ output bits represent the A part of the input and the remaining input/output bits represent the B part of the input. Call the circuits restricted to these parts of the input/output bits U^{C} and V^{C}, respectively. We call our $\mathrm{Either}(\mathrm{A}^{\mathrm{C}},\mathrm{B}^{\mathrm{C}})$ oracle on $U^{\mathrm{C}}, V^{\mathrm{C}}$ and get back an answer to one, which we then output. The reduction clearly runs in polynomial time and correctness follows by the definition of $\mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{C}}$. □

Lemma 4.5. *Let* A,B *be* TFNP *circuit problems. For all positive integers* i, $\mathrm{Either}(\mathrm{A},\mathrm{B})^i$ *has a many-one reduction to* $\mathrm{Either}(A^i, B^i)$.

Proof. We will prove the lemma by induction on i. It trivially holds for $i = 1$. We now show the inductive case.

By the inductive hypothesis and Lemma 3.5, $\mathrm{Either}(\mathrm{A},\mathrm{B})^i = \mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{Either}(\mathrm{A},\mathrm{B})^{i-1}}$ reduces to $\mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{Either}(\mathrm{A}^{i-1},\mathrm{B}^{i-1})}$. By Lemma 3.5 and Lemma 4.4, this reduces to $\mathrm{Either}(\mathrm{A}^{\mathrm{Either}(\mathrm{A}^{i-1},\mathrm{B}^{i-1})}, \mathrm{B}^{\mathrm{Either}(\mathrm{A}^{i-1},\mathrm{B}^{i-1})})$. This problem clearly reduces to $\mathrm{Either}(\mathrm{A}^{\mathrm{A}^{i-1}}, \mathrm{B}^{\mathrm{B}^{i-1}})$, which is equivalent to $\mathrm{Either}(\mathrm{A}^i,\mathrm{B}^i)$, as desired. □

We now show that in the case when B is self-low, one can reduce $\mathrm{Either}(\mathrm{A}^i,\mathrm{B})$ to $\mathrm{Either}(\mathrm{A},\mathrm{B})^i$. This is of interest since we know that many important TFNP problems are in fact self-low. In particular, [GL25] showed that all problems which are complete for PLS, PPA, or LOSSY under black-box reductions are self-low.

Theorem 4.6. *Let* A,B *be* TFNP *circuit problems. For all positive integers* i, $\mathrm{Either}(\mathrm{A}^i, B)$ *has a many-one reduction to* $\mathrm{Either}(\mathrm{A},\mathrm{B})^i$.

Proof. We show this by induction on i. It clearly holds for $i = 1$. We now show the inductive case. First, note that we can reduce an $\mathrm{Either}(\mathrm{A}^{i+1},\mathrm{B})$ instance C, D to $\mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{Either}(\mathrm{A}^i,\mathrm{B})}$ as follows. We replace each A^i oracle gate in C queried on an A^i instance I with an oracle gate for $\mathrm{Either}(\mathrm{A}^i,\mathrm{B})$ which asks us to solve A^i on I or B on D. If all these $\mathrm{Either}(\mathrm{A}^i,\mathrm{B})$ oracle queries are answered with solutions to A^i, then we will be able to solve C, and if any is answered with a solution to B, then we will be able to solve D. Therefore, $\mathrm{Either}(\mathrm{A}^{i+1},\mathrm{B})$ reduces to $\mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{Either}(\mathrm{A}^i,\mathrm{B})}$. Furthermore, by the inductive hypothesis, $\mathrm{Either}(\mathrm{A}^i,\mathrm{B})$ reduces to $\mathrm{Either}(\mathrm{A},\mathrm{B})^i$. Therefore, by Lemma 3.5, $\mathrm{Either}(\mathrm{A},\mathrm{B})^{\mathrm{Either}(\mathrm{A}^i,\mathrm{B})}$ reduces to $\mathrm{Either}(\mathrm{A},\mathrm{B})^{(\mathrm{Either}(\mathrm{A},\mathrm{B})^i)} = \mathrm{Either}(\mathrm{A},\mathrm{B})^{i+1}$. Chaining these reductions yields the desired result. □

Finally, we use Theorem 4.6 to connect problems which we know are self-low (e.g., SoD) to those whose self-low status remains unknown (e.g., EoL).

Corollary 4.7. *If for any nonnegative integer i,* EoL^{i+1} *has a many-one reduction to* EoL^{i}*, then for all integers $j \geq i$,* $\mathrm{Either}(\mathrm{EoL}, \mathrm{SoD})^{j}$ *has a many-one reduction to* $\mathrm{Either}(\mathrm{EoL}, \mathrm{SoD})^{i}$*. Similarly, if for any nonnegative integer i,* SoL^{i+1} *has a many-one reduction to* SoL^{i}*, then for all integers $j > i$,* $\mathrm{Either}(\mathrm{SoL}, \mathrm{SoD})^{j}$ *has a many-one reduction to* $\mathrm{Either}(\mathrm{SoL}, \mathrm{SoD})^{i}$*.*

Proof. We prove this for the case of EoL. $\mathrm{Either}(\mathrm{EoL}, \mathrm{SoD})^{j}$ reduces to $\mathrm{Either}(\mathrm{EoL}^{j}, \mathrm{SoD}^{j})$ by Lemma 4.5. Since EoL^{j} reduces to EoL^{i} (Observation 4.1) and SoD is self-low, $\mathrm{Either}(\mathrm{EoL}^{j}, \mathrm{SoD}^{j})$ reduces to $\mathrm{Either}(\mathrm{EoL}^{i}, \mathrm{SoD})$. By Theorem 4.6, $\mathrm{Either}(\mathrm{EoL}^{i}, \mathrm{SoD})$ reduces to $\mathrm{Either}(\mathrm{EoL}, \mathrm{SoD})^{i}$. Chaining these three reductions proves the desired corollary.

The same reasoning applies to SoL. □

5 The WeakPigeon hierarchy and the consequences of its collapse

5.1 Multi-Collisions in the WeakPigeon hierarchy

We will now show that, for any fixed integer $\ell > 1$, though the problem of finding a 2^{ℓ}-wise collision does not reduce to WeakPigeon under black-box reductions [JLRX24], it does reduce to $\mathrm{WeakPigeon}^{\ell}$ under black-box reductions. This was first shown implicitly in the work of Buzek and Tessaro [BT24], who construct collision-resistant hash functions from multicollision-resistant hash functions. What we do here can be seen as interpreting their work in our language of hierarchies. We now give some intuition underlying the rather technical proof.

For simplicity, let us fix $\ell = 2$ and consider the problem of finding a 4-wise collision in $C : \{0,1\}^{2n} \to \{0,1\}^{n-1}$. Say we could find distinct $x, x' \in \{0,1\}^{n}$, distinct $t_1, t_2 \in \{0,1\}^{n}$, and distinct $t'_1, t'_2 \in \{0,1\}^{n}$ (note that $x \circ t_1, x \circ t_2, x' \circ t'_1, x' \circ t'_2$ are all distinct) such that all the following conditions hold.

$$C(x \circ t_1) = C(x \circ t_2)$$
$$C(x' \circ t'_1) = C(x' \circ t'_2)$$
$$C(x \circ t_1) = C(x' \circ t'_1)$$

Then we would have found a 4-wise collision in C. Indeed, these three conditions would ensure $C(x \circ t_1) = C(x \circ t_2) = C(x' \circ t'_1) = C(x' \circ t'_2)$. We will now construct a $\mathrm{WeakPigeon}^{2}$ instance $C^{\mathrm{WeakPigeon}}$ where valid subwitnesses to the internal WeakPigeon gate will ensure $C(x \circ t_1) = C(x \circ t_2), C(x' \circ t'_1) = C(x' \circ t'_2)$ and a collision in $C^{\mathrm{WeakPigeon}}$ will ensure that the third condition $C(x \circ t_1) = C(x' \circ t'_1)$ holds.

Theorem 5.1. *For every positive integer ℓ,* $2^{\ell}\text{-Pigeon}(2^{\ell n}, 2^{n-1})$ *reduces to* $\mathrm{WeakPigeon}^{\ell}$*.*

Proof. We show this via induction on ℓ. It holds trivially for the case $\ell = 1$. We now show it for the inductive case. Assume the theorem holds for the case of $\ell - 1$. For notational convenience, let $\mathrm{P} := 2^{\ell-1}\text{-Pigeon}(2^{(\ell-1)n}, 2^{n-1})$. Say we are given a circuit $C : \{0,1\}^{\ell n} \to \{0,1\}^{n-1}$ in which we wish to find a 2^{ℓ}-wise collision. We reduce the problem of finding a 2^{ℓ}-wise collision in C to $\mathrm{WeakPigeon}^{\mathrm{P}}$, which by the inductive hypothesis and Lemma 3.5 reduces to $\mathrm{WeakPigeon}^{(\mathrm{WeakPigeon}^{\ell-1})} = \mathrm{WeakPigeon}^{\ell}$.

The reduction to $\mathrm{WeakPigeon}^{\mathrm{P}}$ constructs an instance of $\mathrm{WeakPigeon}^{\mathrm{P}}$ as follows. For convenience, we first define $C_x : \{0,1\}^{(\ell-1)\cdot n} \to \{0,1\}^{n-1}$ as $C_x(a) = C(x \circ a)$ for any $x \in \{0,1\}^{n}$.

We then define $C'^{\mathrm{P}} : \{0,1\}^{n} \to \{0,1\}^{n-1}$ so that on input $x \in \{0,1\}^{n}$, it feeds C_x to its P oracle to get back $t_1, \ldots, t_{2^{\ell-1}}$ and then outputs $C(x \circ t_1)$. We now feed C'^{P} to our $\mathrm{WeakPigeon}^{\mathrm{P}}$ oracle to get back $x, t_1, \ldots, t_{2^{\ell-1}}, x', t'_1, \ldots, t'_{2^{\ell-1}}$ where $C'_{\mathrm{Eval}}(x, t_1, \ldots, t_{2^{\ell-1}}) = C'_{\mathrm{Eval}}(x', t'_1, \ldots, t'_{2^{\ell-1}}) \neq \perp$ and $x \neq x'$. The reduction then outputs $x \circ t_1, \ldots, x \circ t_{2^{\ell-1}}, x' \circ t'_1, \ldots, x' \circ t'_{2^{\ell-1}}$.

The reduction clearly runs in polynomial time. We now show correctness. The definition of $\mathrm{WeakPigeon}^{\mathrm{P}}$ ensures that $x \neq x'$, $t_1, \ldots, t_{2^{\ell-1}}$ are all distinct, and $t'_1, \ldots, t'_{2^{\ell-1}}$ are all distinct. Therefore, $x \circ t_1, \ldots, x \circ t_{2^{\ell-1}}, x' \circ t'_1, \ldots, x' \circ t'_{2^{\ell-1}}$ are all distinct.

All that is left to do is show that $C(x \circ t_1) = \cdots = C(x \circ t_{2^{\ell-1}}) = C(x' \circ t'_1) = \cdots = C(x' \circ t'_{2^{\ell-1}})$. Notice that the following three equalities hold.

$$C_x(t_1) = C_x(t_2) = \cdots = C_x(t_{2^{\ell-1}}) \tag{1}$$

$$C_{x'}(t'_1) = C_{x'}(t'_2) = \cdots = C_{x'}(t'_{2^{\ell-1}}) \tag{2}$$

$$C'_{\text{Eval}}(x, t_1, \ldots, t_{2^{\ell-1}}) = C'_{\text{Eval}}(x', t'_1, \ldots, t'_{2^{\ell-1}}) \tag{3}$$

Equation (1) holds since $t_1, \ldots, t_{2^{\ell-1}}$ is a solution to P on input C_x. Equation (2) holds since $t'_1, \ldots, t'_{2^{\ell-1}}$ is a solution to P on input $C_{x'}$. Equation (3) holds since $x, t_1, \ldots, t_{2^{\ell-1}}, x', t'_1, \ldots, t'_{2^{\ell-1}}$ is a solution to WeakPigeon$^{\text{P}}$ on input C'^{P}. By our definition of C'^{P}, $C'^{\text{P}}(x)$ evaluated using valid subwitnesses $u_1, \ldots, u_{2^{\ell-1}}$ always outputs $C(x \circ u_1)$. Therefore, the left-hand side of Equation (3) evaluates to $C(x \circ t_1)$ and the right-hand side evaluates to $C(x' \circ t'_1)$. Therefore, $C(x \circ t_1) = C(x' \circ t'_1)$. Combining this with Equation (1) and Equation (2) yields the desired result.

□

Corollary 5.2. *For every positive integer ℓ, 2^ℓ-Pigeon$(2^{\ell n}, 2^{n-1})$ reduces to* Pigeon$^\ell$.

Proof. Observe that by Lemma A.1, WeakPigeon$^\ell$ reduces to Pigeon$^\ell$, and then apply Theorem 5.1. □

Applying Observation 4.1 immediately yields the following corollary.

Corollary 5.3. *Let $\mathcal{C} \in \{\mathsf{FP}, \mathsf{TFZPP}, \mathsf{FP/poly}\}$ and $\ell \in \mathbb{N}$.* WeakPigeon $\in \mathcal{C}$ *if and only if* 2^ℓ-Pigeon$(2^{\ell n}, 2^{n-1}) \in \mathcal{C}$.

Proof. The if direction follows from the fact that 2-Pigeon$(2^{\ell n}, 2^{n-1})$ is complete for PWPP [Jeř16]. The only if direction follows from combining Theorem 5.1 with Observation 4.1 and Corollary 4.3. □

We can also use Theorem 5.1 along with the following theorem of [JLRX24] to show that WeakPigeon^2 is not contained in PPP under black-box reductions. This stands in contrast to the first level of the WeakPigeon hierarchy, which is contained in PPP.

Theorem 5.4 ([JLRX24])**.** *If t is a constant and M, N, M', N' are parameters chosen so that $M' = \text{poly}(M)$, $M \geq tN + 1$, and $M' \geq (t-1)N' + 1$, then $t+1$-Pigeon(M, N) does not have an efficient black-box reduction to t-Pigeon(M', N').*

Corollary 5.5. *There is no black-box reduction from* WeakPigeon^2 *to* Pigeon.

Proof. If there were a black-box reduction from WeakPigeon^2 to Pigeon, there would be a black-box reduction from 4-Pigeon$(2^{2n}, 2^{n-1})$ to Pigeon, contradicting Theorem 5.4. □

5.2 A collapse of the Pigeon hierarchy if PPP = promiseUEOPL

Here we show that if PPP = promiseUEOPL, then the Pigeon hierarchy collapses to promiseUEOPL. To do so, we will state the more general theorem that Pigeon$^{\text{SoL}}$ is many-one equivalent to Pigeon, assuming all SoL instances in the Pigeon$^{\text{SoL}}$ instance have a unique solution. This means that PPP is somewhat capable of simulating access to a PPADS machine. Therefore, when reducing to PPP, one can generally assume that problems which reduce to PPADS (like simple stochastic games) have polynomial time algorithms. Again, there is one major caveat: *we assume that the* SoL *oracle gates are only ever queried on inputs which have a unique solution.* Therefore, before presenting our proof, we develop some of the theory around TFNP oracle problems where we know that the oracle gates will only be queried on instances that have unique solutions.

5.2.1 Unique solution oracles

We want to define a version of Pigeon$^{\text{SoL}}$ where the SoL instances on which the oracle is queried have a unique solution. Sadly, checking if a SoL instance has a unique solution is likely intractable. Therefore, it seems unclear how we can ensure that the SoL instances submitted to the SoL oracle gate in the input to Pigeon$^{\text{SoL}}$ have a unique solution. Our solution to this is very much in the spirit of how Pigeon is defined. Recall that in the definition of Pigeon, we are really interested in finding a preimage of 0 in the case where the input to Pigeon is a permutation $C : [N] \to [N]$. However, to make Pigeon a TFNP problem, we add another type of solution which always occurs in the case that C is not a permutation, namely, a collision in C. We will take a similar approach to defining oracle TFNP problems where the oracle should only be queried on instances with a unique solution.

Definition 5.6. *Let* A *be a* TFNP *circuit problem and let* B *be a* TFNP *problem. The problem* $\text{A}^{\exists!\text{B}}$ *is defined as follows. We are given as input a* $\text{poly}(n)$ *size oracle circuit* $C^{\text{B}} : \{0,1\}^n \to \{0,1\}^{\text{poly}(n)}$ *and some auxiliary input* $a \in \{0,1\}^{\text{poly}(n)}$*. The goal is to output one of the following.*

1. *A solution to* A^{B} *on input* C^{B}, a*.*
2. $x \in \{0,1\}^n$*,* $i \geq 0$*, and* $w_1, \ldots, w_i$*,* (w_{i+1}, w'_{i+1}) *such that*
   (a) $w_{i+1} \neq w'_{i+1}$*,*
   (b) *for all* $j \in [1, i+1]$*,* w_j *is a valid solution to oracle gate query* j *of* C^{B} *when the first* $j-1$ *queries are answered using* $w_1, \ldots, w_{j-1}$ *respectively,*
   (c) w'_{i+1} *is a valid solution to oracle gate query* $i+1$ *of* C^{B} *when the first* i *queries are answered using* $w_1, \ldots, w_i$ *respectively.*

One should think of type 1 solutions to $\text{A}^{\exists!\text{B}}$ as the desired type of solutions which will always occur when the oracle gates in C^{B} are always queried on instances with unique solutions (the good case). One should also think of type 2 solutions as degenerate solutions, those which will always occur when the oracle gates in C^{B} are queried on instances which sometimes do not have unique solutions (the bad case). In particular, w_{i+1}, w'_{i+1} witness the fact that there exists an x such that some B oracle query that can be made during the evaluation of $C^{\text{B}}(x)$ does not admit unique solutions. In particular, this holds since w_{i+1} and w'_{i+1} are both solutions to the $(i+1)$th B oracle query made while evaluating $C^{\text{B}}(x)$ when the first i subwitnesses are $w_1, \ldots, w_i$. However, we should be careful to note it is possible to be in the bad case and still have a type 1 solution exist. This should be reminiscent of the definition of Pigeon where we are in the case where the input C is not a permutation but 0 still has a preimage. We now formalize this idea in the following claim.

Claim 5.7. *Let* A *be a* TFNP *circuit problem and let* B *be a* TFNP *problem. The problem* $\text{A}^{\exists!\text{B}}$ *is a* TFNP *problem.*

Proof. $\text{A}^{\exists!\text{B}}$ is total: a type 1 solution always exists since A^{B} is total. It is not hard to see that $\text{A}^{\exists!\text{B}}$ is an NP relation. Type 1 solutions are efficiently verifiable since A^{B} is a TFNP problem. Type 2 solutions are efficiently verifiable since verifying them simply requires checking whether $\text{poly}(n)$ subwitnesses are valid subwitnesses for the oracle gates to the TFNP problem B on certain inputs, along with some polynomial time computation, both of which can be done in polynomial time. □

5.2.2 Pigeon with a unique SoL oracle

Theorem 5.8. Pigeon$^{\exists!\text{SoL}}$ *has a many-one reduction to* Pigeon.

Proof. We reduce Pigeon$^{\exists!\text{SoL}}$ to Pigeon. We assume without loss of generality that the input circuit $C^{\text{SoL}} : [N] \to [N]$ has exactly one SoL oracle gate. We can do this by Turing closure of PPADS (Lemma 2.16)[5]. If, at some point during the reduction, we find that uniqueness of SoL solutions in C^{SoL} is not satisfied, we output a type 2 solution to Pigeon$^{\exists!\text{SoL}}$.

[5]Though the fact that collapsing $\text{poly}(n)$ SoL queries down to one is not stated as a theorem in [BJ12], a quick look at the proof confirms this.

For any $x \in [N]$, we will let $P_x, S_x : [M] \to [M]$ denote the SoL instance queried by $C^{\text{SoL}}(x)$. We assume without loss of generality that $P_x(0) = 0$ and $0 \notin \text{Im}(S_x)$ (we can always reduce solving SoL on P_x, S_x to solving SoL on P'_x, S'_x such that these conditions are satisfied).

We now reduce to a Pigeon instance $C' : [N] \times [M] \to [N] \times [M]$. We define $C'(x, y)$ as follows.

1. If $P_x(S_x(y)) = y$, output $(x, S_x(y))$.
2. If $P_x(S_x(y)) \neq y$, output $(C_{\text{Eval}}(x, y), 0)$.

We now feed C' to our Pigeon oracle to get either i) (a, b) such that $C'(a, b) = (0, 0)$, or ii) $(a, b), (a', b')$ such that $(a, b) \neq (a', b')$ but $C'(a, b) = C'(a', b')$. We now split the behavior of the reduction into three cases.

1. If $C'(a, b) = (0, 0)$, we output (a, b) as a type 1 solution to Pigeon$^{\exists!\text{SoL}}$.
2. If $C'(a, b) \neq (0, 0)$ and $a = a'$, then we output $a, (b, b')$ as a type 2 solution to Pigeon$^{\exists!\text{SoL}}$.
3. If $C'(a, b) \neq (0, 0)$ and $a \neq a'$, we output $(a, b), (a', b')$ as a type 1 solution to Pigeon$^{\exists!\text{SoL}}$.

The reduction clearly runs in polynomial time. We now show correctness. First, suppose $C'(a, b) = (0, 0)$. Then, $C'(a, b)$ is evaluated using case 2 since $S_a(b) \neq 0$ by assumption. Therefore, $C_{\text{Eval}}(a, b) = 0$, making (a, b) a solution to the original instance, implying that our reduction works in the case that $C'(a, b) = (0, 0)$. We therefore restrict our attention to the case that $C'(a, b) \neq (0, 0)$ going forward.

To reduce the number of cases we need to analyze, we first show that it can never be the case that one of $C'(a, b), C'(a', b')$ is evaluated using case 1 and the other using case 2. To see why, assume towards a contradiction one of $C'(a, b), C'(a', b')$ is evaluated using case 1 and the other using case 2. Assume without loss of generality that $C'(a, b)$ is evaluated using case 1. Since $C'(a, b) = C'(a', b')$, unwrapping the definition of C', we see that $(a, S_a(b)) = (C_{\text{Eval}}(a', b'), 0)$. But by assumption, $0 \notin \text{Im}(S_a)$. We have therefore arrived at a contradiction.

We now split the analysis into two cases.

1. Say $a = a'$. Since $(a, b) \neq (a', b')$, this implies that $b \neq b'$. Therefore $C'(a, b) = C'(a, b')$. There are two subcases.
   (a) $C'(a, b)$ and $C'(a, b')$ are both evaluated using case 1. That means $P_a(S_a(b)) = b$ and $P_a(S_a(b')) = b'$. Since $C'(a, b) = C'(a, b')$, $S_a(b) = S_a(b')$. This implies $P_a(S_a(b)) = P_a(S_a(b')) = b'$. This contradicts the fact that $P_a(S_a(b)) = b$. Therefore, this case can never occur.
   (b) $C'(a, b)$ and $C'(a, b')$ are both evaluated using case 2. This implies that $P_a(S_a(b)) \neq b$ and $P_a(S_a(b')) \neq b'$. Notice that this means that b and b' are two different solutions to the SoL instance (P_a, S_a), namely b and b'. Therefore, $a, (b, b')$ is a type 2 solution to our Pigeon$^{\exists!\text{SoL}}$ instance.
2. Say $a \neq a'$. Again, there are two subcases.
   (a) $C'(a, b)$ and $C'(a', b')$ are evaluated using case 1. This cannot happen since then $C'(a, b) = (a, S_a(b)) \neq (a', S_{a'}(b')) = C'(a', b')$, violating the assumption that $C'(a, b) = C'(a', b')$.
   (b) $C'(a, b)$ and $C'(a', b')$ are evaluated using case 2. This implies $C_{\text{Eval}}(a, b) = C_{\text{Eval}}(a', b') \neq \perp$. Therefore, $(a, b), (a', b')$ form a type 1 solution to our Pigeon$^{\exists!\text{SoL}}$ instance.

Therefore, the reduction always outputs a valid answer. □

We can then use Theorem 5.8 to observe that if PPP reduces to SoL where the SoL instances generated have unique solutions, then the whole Pigeon hierarchy collapses to SoL. Although this result is certainly interesting, SoL instances that have unique solutions do not form a standard class in TFNP (since we do not know how to check efficiently whether a given SoL instance has a unique solution). Fortunately, promiseUEOPL has been defined [FGMS20] and immediately reduces to such SoL instances.

Corollary 5.9. *If* PPP $\subseteq$ promiseUEOPL*, then* Pigeon* $\subseteq$ promiseUEOPL.

Proof. We will show that Pigeon$^i \in$ promiseUEOPL by induction on i. This holds for the case $i = 1$ by the hypothesis PPP $\subseteq$ promiseUEOPL. We now consider the inductive case.

We first show that if Pigeon$^i \in$ promiseUEOPL, then Pigeon^{i+1} reduces to Pigeon$^{\exists!\text{UEOPL}}$. This closely mirrors the proof of Lemma 3.5 given in [GL25]. Let C^{Pigeon^i} be the input Pigeon^{i+1} instance and let R be the reduction from Pigeon^i to UEOPL which outputs UEOPL instances with a unique end-of-potential line (e.g. promiseUEOPL instances). We view R as an oracle gate circuit M^{UEOPL} with one UEOPL oracle gate. In particular, M^{UEOPL} takes in a Pigeon^i instance x, performs some computation f to produce a UEOPL instance $f(x)$, calls its UEOPL oracle gate on $f(x)$ to get back o, and then performs some computation $g(o)$ which outputs a solution to Pigeon^i on x. We now construct D^{UEOPL} by replacing each oracle gate in C^{Pigeon^i} by M^{UEOPL} (of the appropriate input length). We now call our Pigeon$^{\exists!\text{UEOPL}}$ oracle on D^{UEOPL} to get back either a type 1 solution or a type 2 solution (see Definition 5.6). Note that since R always outputs a UEOPL instance with a unique end-of-potential-line, there cannot be a type 2 solution. Let $z, w_1, \ldots, w_\ell$ be the type 1 solution output by our Pigeon$^{\exists!\text{UEOPL}}$ oracle where w_i answers instance u_i. We now output $z, g(u_1, w_1), \ldots, g(u_\ell, w_\ell)$.

The reduction is clearly efficient since R is efficient. Correctness of our reduction follows from the correctness of R. In particular, since $w_1, \ldots, w_\ell$ are answers to the UEOPL queries made while verifying the solution z on D^{UEOPL}, $g(u_1, w_1), \ldots, g(u_\ell, w_\ell)$ are the answers to the Pigeon^i queries made while verifying the solution z on C^{Pigeon^i}.

Therefore, using the above, along with the inductive hypothesis, Pigeon^{i+1} reduces to Pigeon$^{\exists!\text{UEOPL}}$. This clearly reduces to Pigeon$^{\exists!\text{SoL}}$, which by Theorem 5.8 reduces to Pigeon, which reduces to promiseUEOPL. Chaining these reductions lets us conclude that Pigeon^{i+1} has a many-one reduction to promiseUEOPL, as desired. □

Corollary 5.10. *If* PPP = promiseUEOPL*, then for any positive integer* ℓ*,* 2^ℓ-Pigeon$(2^{\ell n}, 2^{n-1}) \in$ promiseUEOPL*.*

Proof. This follows from Corollary 5.9 and Corollary 5.2. □

5.3 A collapse of the WeakPigeon hierarchy if PWPP = LOSSY

Here, we show that if PWPP = LOSSY, then the WeakPigeon hierarchy collapses to LOSSY. To do so, we start by showing that Lossy is low for WeakPigeon. In other words, a Lossy oracle gives WeakPigeon no additional power.

Theorem 5.11. WeakPigeon$^{\text{Lossy}}$ *has a many-one reduction to* WeakPigeon*.*

Proof. We will show a reduction from WeakPigeon$^{\text{Lossy}}$ to WeakPigeon. By Lemma A.2, it suffices to show a reduction from (2-Pigeon$(2^{n+1}, 2^{n-1})$)$^{\text{Lossy}}$ to WeakPigeon. Let $C^{\text{Lossy}} : \{0,1\}^{n+1} \to \{0,1\}^{n-1}$ be the (2-Pigeon$(2^{n+1}, 2^{n-1})$)$^{\text{Lossy}}$ instance the reduction is given. Because LOSSY is Turing-closed (Lemma 2.17), we will assume that there is exactly one Lossy oracle gate in the circuit C (note that we can do this since we are not asking for a consistent solution in WeakPigeon$^{\text{Lossy}}$). By Lemma 2.18 and Lemma 3.5, we will also assume without loss of generality that this Lossy gate takes as input $C_x : \{0,1\}^{m+1} \to \{0,1\}^{m-1}, D_x : \{0,1\}^{m-1} \to \{0,1\}^{m+1}$ (parameterized by x).

We now construct the WeakPigeon instance to which we reduce: $C' : \{0,1\}^{n+1} \times \{0,1\}^{m+1} \to \{0,1\}^{n+m+1}$. $C'(x, y)$ behaves as follows.

1. If y is a valid solution to Lossy on input C_x, D_x (i.e., $D_x(C_x(y)) \neq y$), then $C'(x, y)$ outputs $C_{\text{Eval}}(x, y) \circ y \circ 0$.
2. Otherwise, $C'(x, y)$ outputs $x \circ C_x(y) \circ 1$.

Note that in case 1, C' outputs a bitstring of length $(n-1) + (m+1) + 1 = n + m + 1$, and in case 2, the circuit outputs a string of length $(n+1) + (m-1) + 1 = n + m + 1$. Therefore, C' indeed does compress by one bit. The reduction then calls its WeakPigeon oracle on C' to get back an answer x_1, y_1, x_2, y_2 such that $C'(x_1, y_1) = C'(x_2, y_2)$ and $(x_1, y_1) \neq (x_2, y_2)$. The reduction then outputs $(x_1, y_1), (x_2, y_2)$.

The reduction clearly runs in polynomial time. We now show correctness. There are three cases to consider.

1. One of $C'(x_1, y_1), C'(x_2, y_2)$ is evaluated using case 1 and the other using case 2. Notice that this cannot happen since that means one of $C'(x_1, y_1), C'(x_2, y_2)$ ends with a zero and the other ends with a one. But this would imply $C'(x_1, y_1) \neq C'(x_2, y_2)$, violating our assumption that $C'(x_1, y_1) = C'(x_2, y_2)$.

2. Both $C'(x_1, y_1), C'(x_2, y_2)$ are evaluated using case 1. If this happens, $C_{\text{Eval}}(x_1, y_1) \neq \perp$, $C_{\text{Eval}}(x_2, y_2) \neq \perp$, and $C_{\text{Eval}}(x_1, y_1) \circ y_1 = C_{\text{Eval}}(x_2, y_2) \circ y_2$. This implies $y_1 = y_2$, and since we know $(x_1, y_1) \neq (x_2, y_2)$, it must be the case that $x_1 \neq x_2$. Therefore, $x_1 \neq x_2$ and $C_{\text{Eval}}(x_1, y_1) = C_{\text{Eval}}(x_2, y_2) \neq \perp$. Therefore, the reduction outputs a valid solution to $\text{WeakPigeon}^{\text{Lossy}}$ on input C^{Lossy}.

3. Both $C'(x_1, y_1), C'(x_2, y_2)$ are evaluated using case 2. Notice that since $C'(x_1, y_1) = x_1 \circ C_{x_1}(y_1) \circ 1 = x_2 \circ C_{x_2}(y_2) \circ 1 = C'(x_2, y_2)$, it must be the case that $x_1 = x_2$. Since we know $(x_1, y_1) \neq (x_2, y_2)$, we must have $y_1 \neq y_2$. By the definition of C' and the fact that $C'(x_1, y_1) = C'(x_2, y_2)$, $C_{x_1}(y_1) = C_{x_2}(y_2)$, and therefore $D_{x_1}(C_{x_1}(y_1)) = D_{x_1}(C_{x_1}(y_2))$. But recall that $x_1 = x_2$, and therefore $D_{x_1}(C_{x_1}(y_1)) = D_{x_2}(C_{x_2}(y_2))$. However, since both $C'(x_1, y_1), C'(x_2, y_2)$ are evaluated using case 2, $D_{x_1}(C_{x_1}(y_1)) = y_1$ and $D_{x_2}(C_{x_2}(y_2)) = y_2$. Therefore, $y_1 = y_2$, violating our assumption that $y_1 \neq y_2$. Therefore, this case never occurs.

□

Corollary 5.12. $\mathsf{PWPP} = \mathsf{LOSSY}$ *if and only if for all positive integers* i, WeakPigeon^i *is equivalent to* Lossy *under many-one reductions. Furthermore,* $\mathsf{PWPP} = \mathsf{LOSSY}$ *if and only if for every positive integer* ℓ, 2^ℓ-$\text{Pigeon}(2^{\ell n}, 2^{n-1})$ *reduces to* Lossy.

Proof. We first show the if direction. If for all positive integers i, WeakPigeon^i is equivalent to Lossy under many-one reductions, then WeakPigeon reduces to Lossy. It is also well-known that Lossy reduces to WeakPigeon unconditionally. These two results together imply $\mathsf{PWPP} = \mathsf{LOSSY}$.

We now show the only if direction by induction on i. It holds for $i = 1$ by the hypothesis that $\mathsf{PWPP} = \mathsf{LOSSY}$. For the inductive case, assume that the theorem holds for i. Then $\text{WeakPigeon}^{i+1} = \text{WeakPigeon}^{\text{WeakPigeon}^i}$ reduces to $\text{WeakPigeon}^{\text{Lossy}}$, which itself reduces to WeakPigeon, which reduces to Lossy. Here, the first reduction follows from the inductive hypothesis (and Lemma 3.5), the second reduction follows from Theorem 5.11, and the final reduction follows from the hypothesis $\mathsf{PWPP} = \mathsf{LOSSY}$. Therefore, WeakPigeon^{i+1} reduces to Lossy, which trivially reduces to WeakPigeon^{i+1}. This implies WeakPigeon^{i+1} is equivalent to Lossy under many-one reductions, completing the inductive case.

The second statement follows by observing that for all positive integers i, WeakPigeon^i is equivalent to Lossy under many-one reductions if and only if for all positive integers ℓ, 2^ℓ-$\text{Pigeon}(2^{\ell n}, 2^{n-1})$ reduces to Lossy. □

6 Separations from the WeakPigeon Hierarchy via Proof Complexity

In this section we show an upper bound on the black-box complexity of the WeakPigeon hierarchy via proof complexity. This upper bound will allow us to show that PPAD does not have a black-box polynomial reduction to the WeakPigeon hierarchy (the same therefore holds for the larger classes PPP, PPA). To establish this, we show that any black-box TFNP search problem which is polynomial time reducible to the WeakPigeon hierarchy must be provably total in bounded depth Frege, by proofs of size quasipolynomial in the oracle input size. Since it is known that totality statements of complete problems for PPAD are not provable in this way, the separation follows.

Key to our proof will be the result of Paris, Wilkie, and Woods [PWW88] that the weak pigeonhole principle has quasipolynomial size, bounded depth Frege proofs. Crucially for us, this result holds even when each pigeon is allowed to fly to multiple holes ([MPW02]) since our WeakPigeon^i instances $C^{\text{WeakPigeon}^{i-1}} : \{0,1\}^n \to \{0,1\}^{n-1}$ define relations rather than functions.

Theorem 6.1 ([PWW88]). *If $(p(x,y))_{x\in\{0,1\}^n, y\in\{0,1\}^{n-1}}$ are propositional variables, then there is a depth $O(1)$, size $2^{\mathrm{poly}(n)}$ Frege proof of the following statement:*

$$\left(\bigwedge_{x\in\{0,1\}^n}\bigvee_{y\in\{0,1\}^{n-1}} p(x,y)\right) \Rightarrow \left(\bigvee_{x\neq x'\in\{0,1\}^n, y\in\{0,1\}^{n-1}} p(x,y)\wedge p(x',y)\right).$$

Our main result here is, in fact, more general than we have stated so far, essentially stating that for any black-box TFNP problem $\mathcal{S}$ which has quasipolynomial size, depth d Frege proofs, WeakPigeon$^{\mathcal{S}}$ has quasipolynomial size, depth $d+C$ Frege proofs, where C is a universal constant. We will assume familiarity with the rules of Frege systems for the remainder of this section; for an in-depth reference the reader may consult [Kra19].

We require the following definition:

Definition 6.2. *Let $\mathcal{S}\subseteq(\{0,1\}^m)^k\times\{0,1\}^k$ define some black-box TFNP search problem in the following sense: given an oracle $G:\{0,1\}^m\to\{0,1\}$, the search problem is to find $x=(x^1,\dots,x^k)\in(\{0,1\}^m)^k$ and $b\in\{0,1\}^k$ so that $(x,b)\in\mathcal{S}$ and $G(x^i)=b_i$ for all $i\leq k$. We interpret $\mathcal{S}$ equivalently as a width-k DNF tautology on 2^m propositional variables $(G(v))_{v\in\{0,1\}^m}$ expressing the totality of $\mathcal{S}$ as a search problem.*

We construct the total search problem/tautology WeakPigeon$^{\mathcal{S}}$ *as follows. The propositional variables are the individual bits of the truth tables of the following functions:*

$$G_1:\{0,1\}^n\times\{0,1\}^m\to\{0,1\}$$
$$\dots$$
$$G_i:\{0,1\}^n\times(\{0,1\}^{mk})^{i-1}\times\{0,1\}^m\to\{0,1\}$$
$$\dots$$
$$G_n:\{0,1\}^n\times(\{0,1\}^{mk})^{n-1}\times\{0,1\}^m\to\{0,1\}$$
$$F:\{0,1\}^n\times(\{0,1\}^{mk})^n\to\{0,1\}^{n-1}$$

The solutions to the search problem WeakPigeon$^{\mathcal{S}}$ *(resp. terms of the DNF) consist of the following: $x^1,x^2\in\{0,1\}^n, \overline{z}^1,\overline{z}^2\in(\{0,1\}^{mk})^n$ so that:*

1. $x^1\neq x^2$
2. $F(x^1,\overline{z}^1)=F(x^2,\overline{z}^2)$
3. *For each $j\in\{1,2\}$ and every $i\in[n]$, we have*

$$(\overline{z}^j[i]_1,\dots,\overline{z}^j[i]_k, G_i(x^j,\overline{z}^j[<i],\overline{z}^j[i]_1),\dots,G_i(x^j,\overline{z}^j[<i],\overline{z}^j[i]_k))\in\mathcal{S},$$

where the third condition uses the following shorthand: for elements $\overline{z}^j\in(\{0,1\}^{mk})^i$, we decompose $\overline{z}^j$ as $\overline{z}^j=\overline{z}^j[1],\dots,\overline{z}^j[i]\in\{0,1\}^{mk}$, set $\overline{z}^j[<r]:=(\overline{z}^j[1],\dots,\overline{z}^j[r-1])$ and $\overline{z}^j[\leq r]:=(\overline{z}^j[1],\dots,\overline{z}^j[r])$, and decompose $\overline{z}^j[i]\in\{0,1\}^{mk}$ as $\overline{z}^j[i]_1,\dots,\overline{z}^j[i]_k\in\{0,1\}^m$.

Definition 6.2 should be seen as a black-box translation of the circuits produced by the procedure described in Section 3.1.3. Note that we may assume without loss of generality that each G_i has a single-bit output as for any $G':\{0,1\}^m\to\{0,1\}^w$ which has a multi-bit output, we may define $G:\{0,1\}^m\times[w]\to\{0,1\}$ as $G(x,i)=G'(x)_i$ without any semantic change to Definition 6.2. We could do the same transformation for F to produce an $F:\{0,1\}^n\times(\{0,1\}^{mk})^n\times[n-1]\to\{0,1\}$ with a single-bit output, but this would require us to add additional constraints (e.g., the subwitnesses $\overline{z}^1_1,\dots,\overline{z}^1_{n-1}$ used to read the $n-1$ bit output of $F(x^1)$ via $F(x^1,\overline{z}^1_1,1),F(x^1,\overline{z}^1_2,2),\dots,F(x^1,\overline{z}^1_{n-1},n-1)$ should all be the same). This would only introduce more complexity into an already rather dense definition. We therefore sacrifice generality to work with the more structured definition of F which has an $n-1$ bit output.

The purpose of this definition is the following:

Lemma 6.3. *Let $\mathcal{R}$ be any search problem which is many-one black-box reducible to a problem in the ith level of the* WeakPigeon *hierarchy. Then, for some $m = \mathrm{poly}(n)$, if we let $\mathcal{S}_1$ be the* WeakPigeon *search problem on oracles $F : \{0,1\}^m \to \{0,1\}^{m-1}$ and let $\mathcal{S}_j = \mathrm{WeakPigeon}^{\mathcal{S}_{j-1}}$ according to Definition 6.2, then $\mathcal{R}$ is many-one black-box reducible to $\mathcal{S}_i$.*

This follows from the relevant definitions and the fact that WeakPigeon is *upwards self-reducible*: we may always efficiently reduce an instance on n-bit input oracles to an instance on m-bit input oracles for any $m > n$, $m \leq \mathrm{poly}(n)$.

We now proceed to the main technical theorem of this section:

Theorem 6.4. *There is an absolute constant C so that the following holds. If $\mathcal{S}$ is as above, there is a Frege proof of size $2^{\mathrm{poly}(n,m,k)}$ and depth C of the totality of* $\mathrm{WeakPigeon}^{\mathcal{S}}$ *using the totality of $\mathcal{S}$ as a substitution axiom. In particular, if $\mathcal{S}$ has a depth d Frege proof of size s, then* $\mathrm{WeakPigeon}^{\mathcal{S}}$ *has a depth $d + C$ Frege proof of size $s2^{\mathrm{poly}(n,m,k)}$.*

Proof. We first define some propositional formulas over the underlying propositional variables of $\mathrm{WeakPigeon}^{\mathcal{S}}$.

1. For $x \in \{0,1\}^n, z \in (\{0,1\}^{mk})^n, b \in \{0,1\}^{n-1}$, define
$$[F(x,z) = b] := \mathbb{1}\left\{\bigwedge_{s=1}^{n-1} F(x,z)_s = b_s\right\}.$$
2. For each $i \leq n$, $\overline{z} \in (\{0,1\}^{mk})^i$, $x \in \{0,1\}^n$, define
$$\alpha_i(x,\overline{z}) := \mathbb{1}\Big\{(\overline{z}[i]_1, \ldots, \overline{z}[i]_k, G_i(x, \overline{z}[< i], \overline{z}[i]_1), \ldots, G_i(x, \overline{z}[< i], \overline{z}[i]_k)) \in \mathcal{S}\Big\}.$$
Note that $\alpha_i(x,\overline{z})$ is a width-k DNF. $\alpha_i(x,\overline{z})$ should be read as saying that $\overline{z}[i]$ is a valid solution for the $\mathcal{S}$ instance $G_i(x, \overline{z}[< i], \cdot)$.
3. For each $i \in [n]$, $x \in \{0,1\}^n$, and $\overline{z} \in (\{0,1\}^{mk})^i$, define
$$V_i(x,\overline{z}) := \bigwedge_{h \leq i} \alpha_h(x, \overline{z}[\leq h]).$$
Again, note that $V_i(x,\overline{z})$ can be represented as an nk-DNF by distribution. $V_i(x,\overline{z})$ should be read as saying that for every $h \leq i$, $\overline{z}[h]$ is a valid solution for the instance $G_h(x, \overline{z}[< h], \cdot)$.

We will first show that for every $i \leq n$ we can derive efficiently the following formula:

$$\phi_i := \bigwedge_{x \in \{0,1\}^n} \bigvee_{\overline{z} \in (\{0,1\}^{mk})^i} V_i(x,\overline{z})$$

To prove ϕ_i, we first need to prove the following formulae $\psi_1, \ldots, \psi_n$, where

$$\psi_i := \bigwedge_{x \in \{0,1\}^n} \bigwedge_{\overline{z} \in (\{0,1\}^{mk})^{i-1}} \bigvee_{\overline{z}_1, \ldots, \overline{z}_k \in \{0,1\}^m} (\overline{z}_1, \ldots, \overline{z}_k, G_i(x, \overline{z}[< i], \overline{z}_1), \ldots, G_i(x, \overline{z}[< i], \overline{z}_k)) \in \mathcal{S}$$

To prove $\psi_1, \ldots, \psi_n$, we simply observe that, for every value of $i, x, \overline{z}$, the formula

$$\bigvee_{\overline{z}_1, \ldots, \overline{z}_k \in \{0,1\}^m} (\overline{z}_1, \ldots, \overline{z}_k, G_i(x, \overline{z}[< i], \overline{z}_1), \ldots, G_i(x, \overline{z}[< i], \overline{z}_k)) \in \mathcal{S}$$

is a substitution instance of $\mathcal{S}$. Hence, we may derive $\psi_1, \ldots, \psi_n$ in depth $d + O(1)$ and size $s2^{\mathrm{poly}(n,m,k)}$ using the $\bigwedge$-introduction rule. Note that ϕ_1 is equivalent to ψ_1, so we have also derived ϕ_1 at this point. At the very end we will show how to derive ϕ_{i+1} from ϕ_i and ψ_{i+1}; by induction this will allow us to derive all $\phi_1, \ldots, \phi_n$.

Say that we have now derived ϕ_n. We next define, for each $x \in \{0,1\}^n$, $b \in \{0,1\}^{n-1}$, the formula

$$R(x,b) = \bigvee_{\overline{z} \in (\{0,1\}^{mk})^n} V_n(x,\overline{z}) \wedge [F(x,\overline{z}) = b].$$

We then may derive, directly from ϕ_n, the formula

$$\bigwedge_{x \in \{0,1\}^n} \bigvee_{b \in \{0,1\}^{n-1}} R(x,b),$$

which says that for every underlying assignment to the variables, R is a total relation. Now, Theorem 6.1 together with modus ponens lets us derive from this

$$\bigvee_{x^1 \neq x^2 \in \{0,1\}^n} \bigvee_{b \in \{0,1\}^{n-1}} R(x^1,b) \wedge R(x^2,b).$$

Expanding the definition of R now gives us

$$\bigvee_{x^1 \neq x^2 \in \{0,1\}^n} \bigvee_{b \in \{0,1\}^{n-1}} \bigvee_{\substack{\overline{z}^1 \in (\{0,1\}^{mk})^n \\ \overline{z}^2 \in (\{0,1\}^{mk})^n}} V_n(x^1,\overline{z}^1) \wedge [F(x^1,\overline{z}^1) = b] \wedge V_n(x^2,\overline{z}^2) \wedge [F(x^2,\overline{z}^2) = b],$$

which is exactly the desired tautology.

It remains only to show that for each $1 \leq i < n$, we can derive ϕ_{i+1} from ϕ_i and ψ_{i+1}. Up to substitution by $O(1)$ depth formulae, it is sufficient to show the following: for finite sets A, B, C with $|A|, |B|, |C| \leq 2^{\mathrm{poly}(n,m,k)}$, propositional variables $(F(a,b))_{a,b \in A \times B}$, $(H(a,b,c))_{a,b,c \in A \times B \times C}$, and formulas

$$\lambda = \bigwedge_{a \in A} \bigvee_{b \in B} F(a,b), \quad \theta = \bigwedge_{(a,b) \in A \times B} \bigvee_{c \in C} H(a,b,c), \quad \gamma = \bigwedge_{a \in A} \bigvee_{(b,c) \in B \times C} F(a,b) \wedge H(a,b,c) \tag{4}$$

show how to derive γ from λ and θ in $O(1)$ depth and size $\mathrm{poly}(|A|, |B|, |C|)$. By $\bigwedge$-weakening, from θ we may derive, for every a, b, the formula $\bigvee_c H(a,b,c)$, and from λ we may derive for every a the formula $\bigvee_b F(a,b)$. Hence, for every a we may derive

$$\begin{aligned}
&\left(\bigwedge_{b \in B} \bigvee_{c \in C} H(a,b,c) \right) \wedge \bigvee_{b \in B} F(a,b) \\
&\to \bigvee_{b \in B} \left(F(a,b) \wedge \bigwedge_{b' \in B} \bigvee_{c \in C} H(a,b',c) \right) \\
&\to \bigvee_{b \in B} \left(F(a,b) \wedge \bigvee_{c \in C} H(a,b,c) \right) \\
&\to \bigvee_{b \in B, c \in C} (F(a,b) \wedge H(a,b,c))
\end{aligned}$$

where the first step uses distributivity, the second uses $\bigwedge$-weakening, and the last uses distributivity again. Using $\bigwedge$-introduction we then derive γ. □

We can then derive the following key corollary which follows directly from Lemma 6.3 and Theorem 6.4:

Corollary 6.5. *For all $i \geq 1$, there is a constant C so that the following holds. If $\mathcal{R}$ is a black-box* TFNP *search problem which is many-one black-box reducible to* WeakPigeon^i*, then $\mathcal{R}$ has depth-C Frege proofs of totality of size quasipolynomial in the size of the oracle input to $\mathcal{R}$.*

Proof. This follows directly by combining Lemma 6.3, Theorem 6.4, and the well-known fact (see [BJ12]) that if $\mathcal{R}$ is black-box many-one reducible to $\mathcal{S}$ and $\mathcal{S}$ has $2^{\mathrm{poly}(n)}$ size bounded-depth Frege proofs of totality, then so does $\mathcal{R}$. □

We finally arrive at our main black-box separation using a powerful lower bound from proof complexity:

Theorem 6.6. *In the black-box model, the complete problem for* PPAD *is not many-one reducible to* WeakPigeon^i *for any constant* i*. Consequently, the same holds for any superclass of* PPAD*, e.g.,* PPP*,* PPA*.*

Proof. The statement of totality of the complete problem for PPP is simply the pigeonhole principle, which was shown to require exponential-size depth-C Frege proofs of totality for any constant C by [KPW95, PBI93]. A complete problem for PPAD called SPERNER was later shown by Dantchev [Dan06] to satisfy the same lower bound via a reduction from the pigeonhole principle lower bounds in [KPW95, PBI93]. A black-box many-one reduction from SPERNER to any fixed level of the WeakPigeon hierarchy would imply quasipolynomial size, bounded depth Frege proofs of SPERNER (by Corollary 6.5), violating this lower bound. □

7 Theorems too good to be true

Given the surprising fact that the problem of finding multicollisions (for an appropriately compressing function) is in Pigeon*, it is natural to ask whether analogous results also hold for other TFNP hierarchies. Here we observe that some containments which are natural to hope for are unlikely. One containment that is natural to hope for is that PPADS is contained in EoL*. This would give us the breakthrough implication that PPAD $\subseteq$ FP $\implies$ PPADS $\subseteq$ FP and would allow us to bypass the black-box barrier showing that there is no black-box reduction from PPADS to PPAD. Unfortunately, such a theorem seems unlikely since it would unconditionally collapse TFNP classes which we believe to be separate.

Theorem 7.1. *If for some nonnegative integer* i*,* PPADS *has a many-one reduction to* EoL^i*, then* PPADS $\subseteq$ PPA*.*

Proof. See Appendix B. □

Other natural theorems to hope for include PPAD $\subseteq$ CLO* and PLS $\subseteq$ CLO* since these would imply CLS $\subseteq$ FP $\implies$ PPAD $\subseteq$ FP and CLS $\subseteq$ FP $\implies$ PLS $\subseteq$ FP, respectively (and bypass known black-box barriers). Here again, we show that such theorems seem unlikely.

Theorem 7.2. *If for some nonnegative integer* i*,* PPAD *has a many-one reduction to* CLO^i*, then* PPAD $\subseteq$ PLS*. Similarly, if for some nonnegative integer* i*,* PLS *has a many-one reduction to* CLO^i*, then* PLS $\subseteq$ PPA*.*

Proof. See Appendix B. □

At a high level, both of the above theorems rely on showing that for some TFNP circuit problems A and B, A reduces to B where B is self-low. This allows us to upper bound A* by B*, which is equivalent to B.

Acknowledgements

The authors would like to thank Zeyong Li, Robert Robere, Noah Stephens-Davidowitz, and Mihalis Yannakakis for helpful discussions and comments on the work.

AI Disclosure

GPT 5.6 Sol and GPT 6 were used to proofread this work for errors. A significant error was found in Section 6, and AI was then used to suggest a fix. The authors verified the fix, adopted an AI-generated draft of the fixed proof which used the same structure as the original erroneous proof (proof of ϕ_i by induction), and then polished the draft. AI was also used to identify and correct other, more minor technical errors as well as for copy editing and literature review. The authors assume all responsibility for correctness and originality.

References

[BGSD25] Huck Bennett, Surendra Ghentiyala, and Noah Stephens-Davidowitz. The More the Merrier! On Total Coding and Lattice Problems and the Complexity of Finding Multicollisions. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:22, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 6, 8

[BJ12] Samuel R Buss and Alan S Johnson. Propositional proofs and reductions between np search problems. *Annals of Pure and Applied Logic*, 163(9):1163–1182, 2012. 6, 8, 18, 25

[BOM04] Josh Buresh-Oppenheim and Tsuyoshi Morioka. Relativized np search problems and propositional proof systems. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 54–67. IEEE, 2004. 4

[BT24] Jan Buzek and Stefano Tessaro. Collision resistance from multi-collision resistance for all constant parameters. In *Annual International Cryptology Conference*, pages 429–458. Springer, 2024. i, 1, 2, 3, 6, 16

[Dam89] Ivan Bjerre Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989. 28

[Dan06] Stefan Dantchev. On the complexity of the sperner lemma. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *Logical Approaches to Computational Barriers*, pages 115–124, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 25

[DP11] Constantinos Daskalakis and Christos Papadimitriou. Continuous local search. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 790–804. SIAM, 2011. 8

[FGHS22] John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: Cls= ppad ∩ pls. *Journal of the ACM*, 70(1):1–74, 2022. 15

[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. 7, 19

[FGPR24] Noah Fleming, Stefan Grosser, Toniann Pitassi, and Robert Robere. Black-box ppp is not turing-closed. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1405–1414, 2024. 6

[GL25] Surendra Ghentiyala and Zeyong Li. Hierarchies within tfnp: building blocks and collapses. *arXiv preprint arXiv:2507.21550*, 2025. i, 1, 2, 3, 5, 9, 10, 12, 13, 15, 20, 28

[Hub26] Pavel Hubáček. Black-box pwpp is not turing-closed. *arXiv preprint arXiv:2602.23809*, 2026. 6

[Jeř16] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016. 2, 8, 14, 17

[JLRX24] Siddhartha Jain, Jiawei Li, Robert Robere, and Zhiyang Xun. On pigeonhole principles and ramsey in tfnp. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 406–428. IEEE, 2024. 3, 5, 8, 16, 17

[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. 7

[Kor22] Oliver Korten. Derandomization from Time-Space Tradeoffs. In Shachar Lovett, editor, *37th Computational Complexity Conference (CCC 2022)*, volume 234 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:26, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 5, 8

[KPW95] Jan Krajíček, Pavel Pudlák, and Alan Woods. An exponential lower bound to the size of bounded depth frege proofs of the pigeonhole principle. *Random Structures & Algorithms*, 7(1):15–39, 1995. 25

[Kra19] Jan Krajíček. *Proof complexity*, volume 170. Cambridge University Press, 2019. 22

[KT+15] Johannes Köbler, Jacobo Torán, et al. Lowness results: the next generation. *Bulletin of EATCS*, 3(117), 2015. 5

[LLR24] Jiawei Li, Yuhao Li, and Hanlin Ren. Metamathematics of resolution lower bounds: A tfnp perspective. *arXiv preprint arXiv:2411.15515*, 2024. 5

[LPT24] Jiatu Li, Edward Pyne, and Roei Tell. Distinguishing, predicting, and certifying: On the long reach of partial notions of pseudorandomness. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–13, 2024. 6, 8

[Mer79] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford university, 1979. 28

[Mor01] Tsuyoshi Morioka. Classification of search problems and their definability in bounded arithmetic. Master's thesis, 2001. 4

[MPW02] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *Journal of Computer and System Sciences*, 64(4):843–872, 2002. 21

[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. 1, 7, 8

[PBI93] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993. 25

[Pip79] Nicholas Pippenger. On simultaneous resource bounds. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 307–311. IEEE, 1979. 1

[PWW88] J. B. Paris, A. J. Wilkie, and A. R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *The Journal of Symbolic Logic*, 53(4):1235–1244, 1988. 4, 21, 22

[RV24] Ron D Rothblum and Prashant Nalini Vasudevan. Collision resistance from multi-collision resistance. *Journal of Cryptology*, 37(2):14, 2024. 3, 6

[Wag86] Klaus W Wagner. The complexity of combinatorial problems with succinct input representation. *Acta informatica*, 23(3):325–356, 1986. 1

A Lifting some reductions

As noted in Section 3.2, under Definition 3.3, we do not know if $A^S \leq B^S$ follows automatically from $A \leq B$ under our definition of TFNP problem iteration. However, we occasionally find this a desirable property and also find that we are able to prove it rather simply. Here, we prove this property for ourselves for the reductions needed in this work.

Lemma A.1. *For every positive integer i,* WeakPigeon^i *has a many-one reduction to* Pigeon^i.

Proof. We show this by induction on i. It clearly holds for $i = 1$. For the inductive case, we can reduce WeakPigeon^{i+1} to $\text{WeakPigeon}^{\text{Pigeon}^i}$ by the inductive hypothesis and Lemma 3.5. Given a $\text{WeakPigeon}^{\text{Pigeon}^i}$ instance $C^{\text{Pigeon}^i} : \{0,1\}^n \to \{0,1\}^{n-1}$, we can reduce it to Pigeon^{i+1} by relativizing the standard reduction which calls its Pigeon^{i+1} oracle on $C'^{\text{Pigeon}^i} : \{0,1\}^n \to \{0,1\}^n$ where $C'^{\text{Pigeon}^i}(x) = C^{\text{Pigeon}^i}(x) \circ 1$. Chaining these reductions shows WeakPigeon^{i+1} reduces to Pigeon^{i+1}. □

Lemma A.2. $\text{WeakPigeon}^{\text{Lossy}}$ *has a many-one reduction to* $(\text{2-Pigeon}(2^{n+1}, 2^{n-1}))^{\text{Lossy}}$.

Proof. We simply recreate the Merkle-Damgård construction [Mer79, Dam89] while being mindful of the Lossy oracles in the input circuit. The input to the reduction will be a circuit $C^{\text{Lossy}} : \{0,1\}^n \to \{0,1\}^{n-1}$ with some number of Lossy oracle gates. We assume without loss of generality that C^{Lossy} has exactly one Lossy oracle gate due to the Turing-closure of LOSSY (Lemma 2.17). Let $C'^{\text{Lossy}} : \{0,1\}^{n+1} \to \{0,1\}^{n-1}$ be defined as $C'^{\text{Lossy}}(x) = C^{\text{Lossy}}(C^{\text{Lossy}}(x_{[1,n]}) \circ x_{n+1})$. Note that C'^{Lossy} has 2 oracle gates. We feed C'^{Lossy} to our $(\text{2-Pigeon}(2^{n+1}, 2^{n-1}))^{\text{Lossy}}$ oracle to obtain distinct $z, z' \in \{0,1\}^{n+1}$ and w_1, w_2, w_1', w_2' such that $C'^{\text{Lossy}}_{\text{Eval}}(z, w_1, w_2) = C'^{\text{Lossy}}_{\text{Eval}}(z', w_1', w_2') \neq \perp$. Let $y = C^{\text{Lossy}}_{\text{Eval}}(z_{[1,n]}, w_1)$ and $y' = C^{\text{Lossy}}_{\text{Eval}}(z'_{[1,n]}, w_1')$. If $z_{[1,n]} \neq z'_{[1,n]}$ and $y = y'$, then the reduction outputs the collision $z_{[1,n]}, z'_{[1,n]}$ with subwitnesses w_1, w_1'. Otherwise, the reduction outputs the collision $y \circ z_{n+1}, y' \circ z'_{n+1}$ with subwitnesses w_2, w_2'.

The reduction clearly runs in polynomial time. We now show correctness. We first note the following facts. By the definition of C', $C'^{\text{Lossy}}_{\text{Eval}}(z, w_1, w_2) = C^{\text{Lossy}}_{\text{Eval}}(y \circ z_{n+1}, w_2)$ and $C'^{\text{Lossy}}_{\text{Eval}}(z', w_1', w_2') = C^{\text{Lossy}}_{\text{Eval}}(y' \circ z'_{n+1}, w_2')$. Combining these two facts gives us the following.

$$C^{\text{Lossy}}_{\text{Eval}}(y \circ z_{n+1}, w_2) = C^{\text{Lossy}}_{\text{Eval}}(y' \circ z'_{n+1}, w_2')$$

Now, the reduction is clearly correct in the case that $z_{[1,n]} \neq z'_{[1,n]}$ and $y = y'$. We therefore consider the cases where either $z_{[1,n]} = z'_{[1,n]}$ or $y \neq y'$. Consider first the case where $z_{[1,n]} = z'_{[1,n]}$. Since $z \neq z'$, it must be the case that $z_{n+1} \neq z'_{n+1}$. Therefore, $y \circ z_{n+1} \neq y' \circ z'_{n+1}$. By the above observations, we know that $C^{\text{Lossy}}_{\text{Eval}}(y \circ z_{n+1}, w_2) = C^{\text{Lossy}}_{\text{Eval}}(y' \circ z'_{n+1}, w_2')$. Therefore, the collision $y \circ z_{n+1}, y' \circ z'_{n+1}$ with subwitnesses w_2, w_2' is a valid solution to our original $\text{WeakPigeon}^{\text{Lossy}}$ instance. Next, consider the case where $y \neq y'$. This implies $y \circ z_{n+1} \neq y' \circ z'_{n+1}$. But, as noted above, $C^{\text{Lossy}}_{\text{Eval}}(y \circ z_{n+1}, w_2) = C^{\text{Lossy}}_{\text{Eval}}(y' \circ z'_{n+1}, w_2')$. Therefore, the collision $y \circ z_{n+1}, y' \circ z'_{n+1}$ with subwitnesses w_2, w_2' is a valid solution to our original $\text{WeakPigeon}^{\text{Lossy}}$ instance. □

B Deferred proofs from Section 7

Here, we prove the barrier results of Section 7. These barriers are fairly straightforward applications of the results of [GL25]. But of course, introducing too many notions from [GL25] (particularly since our definition of A^{B} is subtly different than theirs) could easily confuse the matter. We therefore choose to introduce the relevant notions here, from which the desired result will follow quite easily.

Under the stricter Definition 3.7, [GL25] proved that the classes PLS and PPA are self-low.

Theorem B.1 ([GL25])**.** *For all positive integers i,* $\mathsf{PLS}^i = \mathsf{PLS}$ *and* $\mathsf{PPA}^i = \mathsf{PPA}$.

Furthermore, as an immediate consequence of the results of [GL25], we have the following fact which will prove useful soon.

Theorem B.2 ([GL25])**.** *Let* A, B *be two* TFNP *subclasses defined with respect to two complete circuit problems* A, B*, respectively. Let* C *be any* TFNP *class. If* $\mathsf{A} \subseteq \mathsf{B}$ *under black-box reductions, then* $\mathsf{A}^{\mathsf{C}} \subseteq \mathsf{B}^{\mathsf{C}}$ *under* FP^{C} *reductions.*

We first show the following useful lemma.

Lemma B.3. *For all positive integers i,* $\mathsf{PPAD}^i \subseteq \mathsf{PPA}$ *and* $\mathsf{CLS}^i \subseteq \mathsf{PPA} \cap \mathsf{PLS}$.

Proof. $\mathsf{PPAD}^i \subseteq \mathsf{FP}^{\mathsf{PPA}^i} = \mathsf{FP}^{\mathsf{PPA}} = \mathsf{PPA}$. Here, the first inclusion holds by $\mathsf{PPAD} \subseteq \mathsf{PPA}$ and Theorem B.2, the first equality holds by Theorem B.1, and the final equality holds by Lemma 2.16. Almost exactly the same proof shows $\mathsf{CLS}^i \subseteq \mathsf{PLS}$. The fact that $\mathsf{CLS}^i \subseteq \mathsf{PPA}$ follows from the fact that $\mathsf{CLS}^i \subseteq \mathsf{PPAD}^i \subseteq \mathsf{PPA}$ where the first containment follows from Theorem B.2. □

We are now ready to prove the barriers of Section 7.

Proof of Theorem 7.1. If PPADS reduces to EoL^i, then $\mathsf{PPADS} \subseteq \mathsf{PPAD}^i \subseteq \mathsf{PPA}$ where the last containment follows from Lemma B.3, as desired. □

Proof of Theorem 7.2. If PPAD reduces to CLO^i, then $\mathsf{PPAD} \subseteq \mathsf{CLS}^i \subseteq \mathsf{PLS}$ where the last containment follows from Lemma B.3. If PLS reduces to CLO^i, then $\mathsf{PLS} \subseteq \mathsf{CLS}^i \subseteq \mathsf{PPA}$ where the last containment follows from Lemma B.3. □

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il