Total NP Search Problems with Abundant Solutions

Jiawei Li
The University of Texas at Austin, TX, USA

Abstract

We define a new complexity class TFAP to capture TFNP problems that possess abundant solutions for each input. We identify several problems across diverse fields that belong to TFAP, including WeakPigeon (finding a collision in a mapping from $[2n]$ pigeons to $[n]$ holes), Yamakawa-Zhandry's problem [43], and all problems in TFZPP.

Conversely, we introduce the notion of "semi-gluability" to characterize TFNP problems that could have a unique or a very limited number of solutions for certain inputs. We prove that there is no black-box reduction from any "semi-gluable" problems to any TFAP problems. Furthermore, it can be extended to rule out *randomized* black-box reduction in most cases. We identify that the majority of common TFNP subclasses, including PPA, PPAD, PPADS, PPP, PLS, CLS, SOPL, and UEOPL, are "semi-gluable". This leads to a broad array of oracle separation results within TFNP regime. As a corollary, $\mathsf{UEOPL}^O \not\subseteq \mathsf{PWPP}^O$ relative to an oracle O.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases TFNP, Pigeonhole Principle

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.75

Funding *Jiawei Li*: Supported by Scott Aaronson's Vannevar Bush Fellowship from the US Department of Defense, the Berkeley NSF-QLCI CIQC Center, a Simons Investigator Award, and the Simons "It from Qubit" collaboration. Also funded in part by Igor Carboni Oliveira's the Royal Society University Research Fellowship URF\R1\191059 and by the EPSRC New Horizons Grant EP/V048201/1.

Acknowledgements We thank the the anonymous ITCS reviewers for their helpful comments, especially for pointing out the relevant paper by Müller [34]. We thank Sid Jain, Robert Robere, Hanlin Ren, Yuhao Li, Rahul Santhanam, Shuichi Hirahara, and Scott Aaronson for discussions. We would also like to express our special thanks to Yurong Chen, Zhiyang Xun, and ChatGPT for their kind assistance in the writing of this paper.

1 Introduction

The complexity class TFNP (**T**otal **F**unctions in NP) contains NP search problems which are *guaranteed* to have a solution for any input. A number of notable problems fall into TFNP, including Factoring, Nash, and Gradient-Descent, among others.

TFNP itself is not a well-behaved class; it is a *semantic* class and is conjectured to have no complete problems [32, 35, 37]. Therefore, a main focus over the past three decades is to classify TFNP problems into different *syntactic* subclasses of TFNP. Each syntactic subclass of TFNP is defined by a canonical complete problem, which seeks to find an object from an exponential-size search space, whose existence is guaranteed by the corresponding combinatorial principle. For example, the PPP-complete problem Pigeon is to find a collision when given a mapping (encoded by a circuit of poly(n) size) from $[2^n + 1]$ to $[2^n]$. By the pigeonhole principle, a collision is guaranteed to exist for every mapping. It is common to use a syntactic subclass and its canonical complete problem interchangeably.

In this work, we study the relationship between TFNP subclasses through the lens of *abundance of solutions*. The combinatorial principles underlying these subclasses often dictate two fundamentally different behaviors concerning the number of solutions available.

Table 1 Several common syntactic subclasses of TFNP.

Class	Combinatorial Principle
PPA	Every undirected graph with an odd-degree node must have another odd-degree node.
PPAD	Every directed graph with an unbalanced node (*out-degree* $\neq$ *in-degree*) must have another unbalanced node.
PPADS	Every directed graph with a positively unbalanced node (*out-degree* $>$ *in-degree*) must have a negatively unbalanced node.
PLS	Every DAG has a sink.
PPP	Every mapping from $[N+1]$ to $[N]$ must have a collision.

For all TFNP subclasses enumerated in table 1, constructing inputs with a *unique* solution is straightforward. Furthermore, it can be posited that these unique-solution instances are the *most challenging* inputs. In most cases, it suffices to show oracle separation [4] or optimal query/communication complexity lower bounds with unique-solution instances only (e.g., [8, 26, 44, 5, 24]). Recently, Chen, Li, and Yannakakis [10] showed that Tarski, a TFNP problem situated in PPAD $\cap$ PLS, can be reduced to Unique-Tarski – a promised version with a unique solution – through a generalized black-box reduction.

Conversely, certain TFNP subclasses inherently possess a multitude of solutions – often exponentially many – for every input. A significant example is the class PWPP [28], which is similar to PPP, but requires to find a collision pair in a mapping from $[2^{n+1}]$ to $[2^n]$. Regardless of the chosen mapping, there will be a minimum of 2^n pairs of distinct collisions.

Further motivation stems from *cryptography* and *quantum computing*. A seminal result by Simon [40] showed that collision-resistant hash function cannot be constructed from one-way permutation in a black-box manner, thereby indicating an oracle separation between PPP (associated with one-way permutation) and PWPP (associated with collision-resistant hash function). This proof leverages, to a degree, the contrast in the abundance of solutions between these two problems. Adding to this, Rosen, Segev, and Shahaf [38] presented a series of findings concerning the average-case complexity of TFNP and cryptographic primitives where the number of solutions played a crucial role. For instance, they demonstrated that, in the *black-box* setting, the existence of injective trapdoor functions does not imply the average-case hardness of PPAD instances (specifically, End-of-Line instances on $\{0,1\}^n$) with $O(2^{o(n)})$ solutions.

Shifting to a quantum perspective. It is proven that the quantum and classical (either randomized or deterministic) query complexity for *total boolean functions* are polynomially related (see Aaronson et al. [2] for a comprehensive exposition). A long-standing open problem was whether this relationship extends to total NP *search* problems. Until recently, Yamakawa and Zhandry [43] refuted this possibility by constructing the first TFNP problem (we call it *Yamakawa-Zhandry's problem*)[1] that exhibits polynomial quantum query complexity alongside exponential randomized query complexity. One can check that Yamakawa-Zhandry's problem yields an exponential number of solutions for any input.

[1] The original version of the Yamakawa-Zhandry's problem was defined relative to a random oracle. It was later adapted to constitute a total problem ([43], Section 6)

In fact, by adapting an argument by Goldreich, Goldwasser, and Ron ([20], Theorem 4.1) to the quantum setting, it can be shown that any TFNP problem exhibiting a super-polynomial quantum speed-up cannot have an efficient quantum algorithm that is *pseudo-deterministic*[2]. In essence, a large quantity of non-trivially distinct solutions is necessary for achieving a super-polynomial quantum speed-up in black-box TFNP.

Given these pieces of evidence, we systematically investigate the following question:

How does the abundance of solutions influence the complexity of TFNP *problems?*

Here we use the term "abundance" rather than "number" because the mere *number* of solutions is not a reliable metric. It is simple to modify any TFNP problems to have exponentially many solutions by padding unnecessary bits to the end of each solution.

1.1 Our Contribution

In this section, We present our results in the black-box (query) model for ease of discussion. Equivalently, these results can be interpreted as relative to specific oracles in the white-box setting. Class and problems in the black-box setting are usually written with the superscript "dt", which stands for decision tree.

We define a new complexity class TFAP^{dt} ("A" stands for "abundant") to capture TFNP^{dt} problems that non-trivially have many solutions for every input. The definition of TFAP^{dt} utilizes a novel formalization of the *abundance of solutions*.

▶ **Definition 1** (Informal). *A* TFNP^{dt} *problem belongs to* TFAP^{dt} *if any* nearly-complete *partial assignment is* witnessing*, or it can be reduced to a* TFAP^{dt} *problem.*

In this context, a partial assignment x is nearly complete if most of the input bits are assigned in x; x is witnessing if a solution o exists such that o remains a valid solution for all (complete) inputs consistent with x. Essentially, a problem within TFAP^{dt} retains its totality even when a minor portion of input bits is removed, and the term "abundant" can be equated to a form of *robust totality*. Buss et al. [7] point out that weak pigeonhole principle are "over-determined", which is basically the concept "robust" defined in Definition 20.

We identify several members of TFAP^{dt}, including PWPP^{dt} and Yamakawa-Zhandry's problem as mentioned in the introduction, and all problems in TFZPP^{dt}.

▶ **Theorem 2.** PWPP^{dt}*, Yamakawa-Zhandry's problem, and* TFZPP^{dt} *are contained in* TFAP^{dt}.

The proof for the inclusion of PWPP^{dt} is straightforward: removing a small number of pigeons from the input still leaves sufficient pigeons to create a collision. The argument for Yamakawa-Zhandry's problem follows a similar logic, using the (weak) averaging principle. The inclusion of TFZPP^{dt} is also quite natural, since any problem in TFZPP^{dt} must have *abundant* random seeds that lead to valid solutions.

Note that both TFZPP^{dt} and Yamakawa-Zhandry's problem are in the intersection of TFNP^{dt} and FBQP^{dt}, and we call this class TFBQP^{dt}. We conjectured that TFBQP^{dt} is also contained in TFAP^{dt}; more intuition behind this conjecture is discussed in Section 6.

[2] An algorithm is pseudo-deterministic if on every input, there exists a canonical solution that is output with high probability.

On the other side, we call a TFNP^{dt} problem (or a class) *lean* if it is not contained in TFAP^{dt}. We introduce the notion of "semi-gluability", extending the "gluability" defined by Göös et al. [22], to characterize lean problems. At a high level, semi-gluability only requires being *gluable* for a *distribution* of inputs, rather than for worst-case inputs as required by gluability. We establish a general theorem that separates *semi-gluable* problems from TFAP.

▶ **Theorem 3.** *No many-one reduction exists from any semi-gluable* TFNP^{dt} *problem to any* TFAP^{dt} *problem. Consequently, all semi-gluable problems are lean.*

we show that the majority of common TFNP^{dt} subclasses are semi-gluable, by which we identify these classes as lean class.

▶ **Theorem 4.** PPA^{dt}, PPAD^{dt}, PPADS^{dt}, PLS^{dt}, PPP^{dt}, CLS^{dt}, SOPL^{dt}, *and* UEOPL^{dt} *are* semi-gluable.

▶ **Corollary 5.** $\mathsf{UEOPL}^{dt} \not\subseteq \mathsf{PWPP}^{dt}$.

Furthermore, we also rule out any *randomized* many-one reduction from UEOPL^{dt} to any TFAP^{dt} problem. Consequently, such a stronger separation also works for any classes that contain UEOPL^{dt}. As most of the black-box separations results previously known in TFNP only work for *deterministic* reductions [4, 33, 6, 22], our results may be of interest in average-case complexity and cryptography.

▶ **Theorem 6.** *There is no randomized many-one reduction from* PPA^{dt}, PPAD^{dt}, PPADS^{dt}, PLS^{dt}, PPP^{dt}, CLS^{dt}, SOPL^{dt} *or* UEOPL^{dt} *to any problems in* TFAP^{dt}.

Our framework of distinguishing "lean" from "abundant" leads to a wide range of oracle separation results. See figure 1 for a summary.

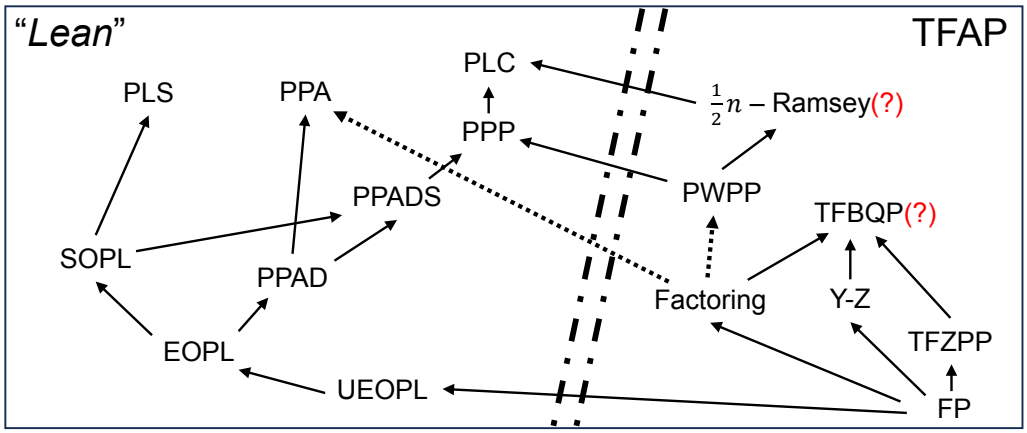

Figure 1 A depiction of the TFNP landscape divided into the domains of TFAP and "lean" classes. Bold arrows represent many-one reductions; dotted arrows are randomized many-one reductions. Informally, our main result asserts that no arrow will point from the left side to the right side in the black-box setting. "Y-Z" is an abbreviation for Yamakawa-Zhandry's problem [43]. The class PLC [36] and $\frac{1}{2}n$-$\textsc{Ramsey}$ are introduced and discussed in Section 6. Note that TFBQP and $\frac{1}{2}n$-$\textsc{Ramsey}$ are only conjectured to be in TFAP, marked with a red question mark.

Relationship with the work of Müller [34]

The authors were not aware of the work of Müller [34] at the time of submission, and it was kindly pointed out by anonymous reviewers. Müller studies the same phenomenon from a slightly different angle using bounded arithmetic. Müller formalizes and identifies *weak* and

strong ("abundant" and "lean" in this work) combinatorial principles by forcing with partial structures. He then proves a general separation theorem ruling out reductions from strong principles to weak, similar to Theorem 3. Besides significant similarities with the work of Müller [34], there are also several differences worth noting:

- Our framework shows black-box separations for PLS and its subclass SOPL, CLS, UEOPL over any TFAP classes; however, Müller's framework is built upon theory $\mathsf{T}_2^1(\mathsf{PV}(\alpha))$, where PLS can be reduced to the empty relation.
- Our work rules out *randomized non-uniform* (decision-tree) many-one reductions; while Müller's work rules out *deterministic uniform* (Turing machine) Turing reductions.

We refer the reader to the full version for a more comprehensive discussion.

Paper organization

Section 2 provides the necessary background on TFNP and other definitions used throughout this paper. We formally define class TFAP^{dt} in Section 3, and show several members of this class. Section 4 introduces the notion of "semi-gluability" and uses it to show that a large portion of the common TFNP subclasses are lean classes. Section 5 discusses the implications of TFAP in a white-box context. We conclude with a discussion on potential future directions in Section 6.

1.2 Background, Related Works

TFNP was initially introduced by Megiddo and Papadimitriou [32]. TFNP problems cannot be NP-hard unless NP = coNP [32]. Furthermore, in contrast to the FNP-complete problem Search-SAT, which is equivalent to the decision problem SAT via the classic search-to-decision reductions, the majority of TFNP problems are inherently *search* problems. These are conjectured not to have corresponding *decision* problems ([4], Section 5).

Class PLS is proposed by Johnson et al. [29]; PPA, PPAD, PPADS, PPP are defined in the seminal work by Papadimitriou [35], which also identified many important problems in these classes. Since then, TFNP theory has effectively captured problems in various fields, including game theory [12, 9], optimization [16, 14], cryptography [41], and combinatorics [18], to name a few. In more recent developments, several new syntactical subclasses of TFNP are defined to better characterize problems in the lower tiers of the TFNP hierarchy, including PWPP [28], CLS [13], EOPL [17, 26], SOPL [23] and UEOPL [17].

There has been notable advancement in understanding the relationship between TFNP subclasses. In a breakthrough result by Fearnley et al. [16], CLS is shown to be equal to $\mathsf{PLS} \cap \mathsf{PPAD}$; following this, Göös et al. [21] showed that $\mathsf{EOPL} = \mathsf{PLS} \cap \mathsf{PPAD} = \mathsf{CLS}$ and $\mathsf{SOPL} = \mathsf{PLS} \cap \mathsf{PPADS}$. On the other side, proving an unconditional separation between TFNP subclasses is not feasible as it would imply $\mathsf{P} \neq \mathsf{NP}$. However, it is still possible to prove unconditional separations relative to certain oracles. Oracle separation in TFNP is of particular interest, given that all the known reductions between TFNP subclasses are relativized.

Beame et al. [4] initiated the study of the oracle separation within TFNP and showed all possible separations between PPP, PPA, PPAD, and PPADS. As of now, all possible separation between PPP, PPA, PPAD, PPADS, PLS, CLS, SOPL, and UEOPL are known due to a series of studies by Morioka [33], Buresh-Oppenheim and Morioka [6], and Göös et al. [22]. A key instrument for achieving several oracle separation results in these works is the inherent connection between proof complexity and black-box TFNP (cf. [15]).

Papadimitriou [35] noted that one-way permutation implies the average-case hardness of class PPP. Following this, there has been a large volume of works studying the relationship between TFNP classes and cryptographic primitives (e.g., [28, 38, 41, 26, 25, 31]). Notably, several works took the number of solutions into consideration [38, 25, 31]. The oracle separation between PPP and PWPP is implied by the black-box separation between one-way permutation and collision-resistant hash function, as proven by Simon [40][3].

The potential of TFNP problems as suitable candidates for efficient quantum algorithms was first brought up by Papadimitriou during a talk in 2003. For a long time, only negative result was known for the majority of TFNP subclasses in the black-box setting [3, 1, 11]. The recent breakthrough by Yamakawa and Zhandry [43] introduced the first problem that not only falls within the intersection of TFNP^{dt} and FBQP^{dt}, but also exhibits high randomized query complexity.

2 Basics of TFNP

We cover the basics of TFNP in white-box (Turing machine) and black-box (decision tree) settings in Section 2.1 and 2.2 respectively. We highlight the interrelation between these two models in Section 2.3. Additional definitions of TFNP subclasses and problems are presented in Section 2.4.

Notations

The notation $\mathrm{poly}(n)$ represents any function that is polynomial in n, while $\mathrm{negl}(n)$ stands for any function that diminishes faster than $\frac{1}{p(n)}$ for any polynomial function $p(n)$. We use uppercase "N" for the input length of black-box problems, while lowercase "n" is mostly used as a parameter for white-box problems, with the general assumption being $n = O(\log N)$ throughout this paper. The i-th bit of a bit string x is represented as x_i.

2.1 White-Box TFNP

▶ **Definition 7** (White-Box TFNP). *A* total NP *search (*TFNP*) problem in the Turing machine model is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ where*

- *$R(x, o)$ is checkable in time* $\mathrm{poly}(|x|, |o|)$*;*
- *for all input x, there is a solution $o, |o| = \mathrm{poly}(|x|)$ such that $(x, o) \in R$.*

▶ **Definition 8.** *A* many-one *reduction from R to Q is a pair of polynomial-time computable functions f, g such that*

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

We present the definition of Pigeon in the white-box model.

▶ **Definition 9.** Pigeon*: The input consists of an n-bit string 1^n and a* $\mathrm{poly}(n)$*-size circuit f, which encodes a mapping f from $2^n + 1$ pigeons to 2^n holes. The goal is to find a collision, i.e., a pair of pigeons $i, j \in [2^n + 1], i \neq j$ and $f(i) = f(j)$.*

[3] Sotiraki et al. ([41], Section 1.1) mentioned that the separation between CLS is PWPP is known, albeit no specific reference is given for this assertion.

Class PPP includes any TFNP problems that can be many-one reduced to Pigeon.

Informally, a white-box TFNP problem is called *natural* if there is no circuit explicitly given in the problem input[4]. For example,

- Factoring: Given a number N (in binary encoding), output "PRIME" if N is prime; otherwise, find a non-trivial factor of N.
- Nash: Given two matrices $A, B \in \mathbb{R}^{n \times n}$ as the payoff matrices of a bimatrix game, find a Nash equilibrium.

All the canonical complete problems used to define syntactical TFNP subclasses are *unnatural* problems, that is, there are explicit circuits given in their inputs. Moreover, all the reductions we currently know between these canonical complete problems are black-box (relativized), i.e., the reduction does not need to look into the implementation of the circuit.

2.2 Black-Box TFNP

▶ **Definition 10** (Black-Box TFNP). *A* total *search problem in the decision tree model is a sequence of relations* $\mathrm{R}^{dt} = \{R_N \subseteq \{0,1\}^N \times O_N\}$*, 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* TFNP^{dt} *if, for each solution* $o \in O_N$*, there is a decision tree* V_o *of depth* $\mathrm{poly}(\log N)$ *such that for every input* $x \in \{0,1\}^N$*,* $V_o(x) = 1$ *iff* $(x, o) \in R_N$*.*

Any unnatural white-box TFNP problem corresponds to a black-box TFNP problem by replacing the circuit given in the input with an oracle with query access. For instance, consider defining Pigeon in the black-box model.

▶ **Definition 11.** Pigeon^{dt}*: Given oracle access to a mapping* f *from* $N+1$ *pigeons to* N *holes. The goal is to find a collision, i.e., a pair of pigeons* $i, j \in [N+1], i \neq j$ *and* $f(i) = f(j)$*.*

WeakPigeon^{dt}*: Same as* Pigeon*, but the input is a mapping from* $[2N]$ *to* $[N]$*.*

To simplify, we adhere to several conventions:

- For problems with the non-binary input alphabet, we simulate it with the usual binary encoding. All problems discussed in this paper have $\mathrm{poly}(N)$ alphabet size and can be simulated with $O(\log N)$ overheads.
- The problem R_N is permitted to have input bits on the order of $\mathrm{poly}(N)$.
- We sometimes abuse notation by calling an individual relation R_N a search problem, rather than a whole sequence $\mathrm{R} = (R_N)$.
- A decision tree is called *low-depth* if it has depth $\mathrm{poly}(\log N)$.
- The superscript "dt" is omitted when referring to TFNP^{dt} problems if the context makes it clear.

▶ **Definition 12** (Decision Tree Reduction). *A low-depth decision tree reduction from relation* $R \subseteq \{0,1\}^N \times O$ *to* $Q \subseteq \{0,1\}^M \times O'$ *is a set of low-depth decision trees* $f_i : \{0,1\}^N \to \{0,1\}$ *for each* $i \in [M]$ *and* $g_o : \{0,1\}^N \to O$ *for each* $o \in O'$ *such that for any* $x \in \{0,1\}^N$*,*

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

where $f(x) \in \{0,1\}^M$ *has* $f_i(x)$ *as the* i*-th bit.*

[4] The terminology of "natural" is adopted by most of the TFNP literature, though one may argue problems like Pigeon are quite *natural* in common sense even with a circuit in it.

A TFNP^{dt} *problem* $\mathrm{R} = (R_N)$ *can be* many-one *reduced to* TFNP^{dt} *problem* $\mathrm{Q} = (Q_N)$ *(written as* $\mathrm{R} \leq_m \mathrm{Q}$*) if for each* N*, there is a low-depth decision tree reduction* (f_i, g_o) *from* R_N *to* Q_M *with* $\log(M) = \mathrm{poly}(\log N)$.

Class PPP^{dt} and PWPP^{dt} include any TFNP^{dt} problems that can be many-one reduced to $\textsc{Pigeon}^{dt}$ and $\textsc{WeakPigeon}^{dt}$ respectively.

▶ **Definition 13** (Randomized Reduction)**.** *A* TFNP^{dt} *problem* $\mathrm{R} = (R_N)$ *can be randomized many-one reduced to* TFNP^{dt} *problem* $\mathrm{Q} = (Q_N)$ *if for each* N*, there is a distribution* D *of low-depth decision tree reductions from* R_N *to* Q_M*, such that* $\log(M) = \mathrm{poly}(\log N)$*, and for any* $x \in \{0,1\}^N$*,*

$$\Pr_{((f_i),(g_o))\sim D}[(x, g_o(x)) \in R \Leftarrow (f(x), o) \in Q] \geq \frac{1}{\mathrm{poly}(\log N)}.$$

2.3 Connection Between the Two Realms

We call a TFNP^{dt} problem R *syntactical*[5] if all the (non-uniform) decision trees $(V_o), o \in O_N$ in Definition 10 can be implemented by a single polynomial-time oracle Turing machine U; that is, for each N,

$$U(o)^x = V_o(x), \forall x \in \{0,1\}^N, \forall o \in O_N.$$

For example, $\textsc{Pigeon}^{dt}$ is syntactical because U can be efficiently implemented by making two queries to the oracle to check whether its input encodes a collision. The oracle Turing machine U can be interpreted as a *syntax* object that encodes a certain existing principle.

If the input oracle x is implemented by a $\mathrm{poly}(\log N)$-size circuit and this circuit is explicitly given to the input, we get an unnatural problem in the white-box TFNP, which further corresponds to a syntactical TFNP subclass. Conversely, given any syntactical TFNP subclass A, one can replace the circuit in its canonical complete problem by an oracle, and thus get a syntactical class in TFNP^{dt}, written as A^{dt}.

We have been using the following theorem implicitly throughout this work.

▶ **Theorem 14** (Beame et al. [4], Informal)**.** *For two syntactical* TFNP *subclasses* A, B*,* $\mathsf{A}^{dt} \nsubseteq \mathsf{B}^{dt}$ implies *the existence of a (generic) oracle* O *separating* A *from* B*.*

2.4 More Definitions

▶ **Definition 15.** *A* TFNP^{dt} *problem* $\mathrm{R} = (R_N)$ *is in* TFZPP^{dt} *if for each* N*, there exists a set of low-depth decision trees* $(T_r), r \in D$ *such that for any* $x \in \{0,1\}^N$*,* $\Pr_{r\sim D}[(x, T_r(x)) \in R_N] \geq \frac{1}{2}$.

The constant $1/2$ here is arbitrary; one can achieve zero error via repetition with $\mathrm{poly}(\log N)$ queries in expectation.

▶ **Definition 16** ($\textsc{End-of-Potential-Line}$, [17])**.** EOPL^{dt}*: Given mappings* $S, P : [N] \to [N]$ *that* $P(1) = 1 \neq S(1)$ *and a function* $V : [N] \to [N]$ *such that* $V(1) = 1$*, find either*

1. *a vertex* $u \in [N]$ *such that* $S(P(u)) \neq u \neq 1$ *or* $P(S(u)) \neq u$*, or*
2. *a vertex* $u \in [N]$ *such that* $u \neq S(u)$*,* $P(S(u)) = u$*, and* $V(S(u)) - V(u) \leq 0$*.*

[5] A *syntactical* TFNP^{dt} problem is essentially a type-2 TFNP (TFNP^2) problem, see [4].

▶ **Definition 17** (UNIQUE-EOPL, [17]). UEOPL^{dt}*: Same as* EOPL^{dt}*, but we allow one more type of solution: two vertex* $u, v \in [N]$*, such that* $u \neq v$*,* $u \neq S(u), v \neq S(v)$*, and either* $V(u) = V(v)$ *or* $V(u) < V(v) < V(S(u))$*.*

Class EOPL^{dt} and UEOPL^{dt} includes all problems that can be many-one reduced to EOPL^{dt} and UEOPL^{dt} respectively. Göös et al. [22] showed that EOPL^{dt} = CLS^{dt}, so both terms will be used interchangeably. Note that UEOPL^{dt} is equivalent to EOPL^{dt} when the input is promised to have a unique valid line.

The definition of other TFNP subclasses are not explicitly used in this paper, and we refer readers to, e.g., Göös et al. ([22], Section 3.2) for a comprehensive summary.

▶ **Definition 18** (Yamakawa-Zhandry's Problem [43], Simplified). *Fix an error correcting code* $C \subseteq \Sigma^n$ *on alphabet* Σ*. Let* (h_k) *be a family of* λ*-wise independent functions from* C *to* $\{0,1\}^n$*.*

The input encodes a mapping $f : \Sigma \to \{0,1\}$*. The goal is to find a key* k *and* t *codewords* $c^{(1)} \ldots, c^{(t)} \in C$ *such that*

$$f(c^{(i)}) \oplus h_k(c^{(i)}) = 0^n, \forall i \in [t],$$

where $\oplus$ *is bitwise-XOR, and* $f(c^{(i)}) \coloneqq (f(c_1^{(i)}), \ldots, f(c_n^{(i)}))$*.*

The parameters satisfy $n < \lambda \ll t = \mathrm{poly}(n)$*,* $|C| \geq 2^{2n}$*, and* $|\Sigma| = 2^{\mathrm{poly}(n)}$*.*

According to the convention, when the problem is considered in the black-box setting, the mapping f is given by the query access to an oracle; in the white-box setting, f is encoded by a $\mathrm{poly}(n)$-size circuit.

We assume the λ-wise independent functions family (h_k) is implemented by the well-known low-degree polynomial construction (see, e.g., Vadhan [42], Section 3.5.5). A *folded Reed-Solomon* code with a certain parameter setting is used for C in [43] to make the problem easy for quantum algorithms but hard for classical algorithms. We will only use the following basic property of code C to show this problem is in TFAP.

▶ **Fact 19.** *Let* S *be any subset of the alphabet* Σ *such that* $|S|/|\Sigma| = \mathrm{negl}(n)$*. Denote the set of codewords that contain at least one of the characters in* S *by* C_S*. We have* $|C_S|/|C| = \mathrm{negl}(n)$*.*

3 Class TFAP and its Member

We formally define TFAP in the black-box setting (i.e., TFAP^{dt}) and discuss some basic properties of TFAP^{dt} in Section 3.1. Then, in Section 3.2, we prove that WEAKPIGEON^{dt}, Yamakawa-Zhandry's problem, and all problems in TFZPP^{dt} belong to TFAP^{dt}.

3.1 Definition of TFAP^{dt}

It is more natural to define the class TFAP in the black-box setting. Let $\mathrm{R} = (R_N), R_N \subseteq \{0,1\}^N \times O_N$ be a TFNP^{dt} problem. A *partial assignment* of R_N is a string $x \in \{0,1,*\}^N$, where $x_i = *$ means the i-th bit is not assigned yet. The *size* of a partial assignment is the number of non-$*$ bits in it. A partial assignment is *witnessing* if there exists some solution $o \in O_N$ that for any (whole) input $y \in \{0,1\}^N$ consistent with x, we have $(y, o) \in R_N$.

▶ **Definition 20.** *A* TFNP^{dt} *problem* $\mathrm{R} = (R_N)$ *is* robust *if for any function* $h = \mathrm{negl}(\log N)$*, there exists a constant* N_0 *such that for each* $N \geq N_0$*, any partial assignment* x *of size at least* $(1 - h(N)) \cdot N$ *is witnessing regarding to* R_N*.*

A TFNP^{dt} *problem* R *is in* TFAP^{dt} *if it is* robust*, or it can be many-one reduced to a* TFAP^{dt} *problem.*

A TFNP^{dt} *problem* R *is* lean *if it is not in* TFAP^{dt}*.*

We define a partial assignment x as *nearly complete* if it has size $(1 - \mathrm{negl}(N)) \cdot N$.

While there exist simple TFAP^{dt} problems that scarcely meet the criteria for being *robust* – for instance, letting the only solution be the first bit of the input – the subsequent lemma demonstrates that, without sacrificing generality, we can presume a TFAP^{dt} problem to be *robust*.

▶ **Lemma 21.** *For any* TFAP^{dt} *problem* R*, there is a robust* TFNP^{dt} *problem* Q *such that* $\mathrm{R} =_m \mathrm{Q}$.

Proof. By definition, $\mathrm{R} = (R_N)$ can be reduced to an *robust* problem S. Let $M = M(N)$ be the input length of S when reducing from an instance of R_N. Without loss of generality, we assume $M(N) \geq N$ and $M(N)$ is monotone increasing; otherwise, one can replace S with solving multiple instances of S, which is an *robust* problem with longer input.

We construct problem $\mathrm{Q} = (Q_K)$ as follow. Assume $K = N + M(N)$ for some N, we interpret the first N bits of inputs as an instance of R_N and the last $M(N)$ bits as an instance of S_M. The goal is to find either a solution for the R_N instance or a solution for the S_M instance.

Since $M(N) \geq N$, S being *robust* implies Q is also *robust*. To reduce from R to Q, one can first reduce R to S and then concatenate both instances as an instance of Q. The reverse direction is trivial. ◀

Next, we scrutinize the definition of TFAP^{dt} through a sanity check. First, we note that merely appending unnecessary bits to the end of each solution does not enhance its *abundance*; similarly, copying the entire input several times is useless.

Moreover, there exist problems that always have many solutions that, nonetheless, do not qualify as *robust* according to our definition. For instance, the problem $\textsc{Pigeon}(N + N^{0.9})$, which mirrors PIGEON but features a mapping from $[N + N^{0.9}]$ to $[N]$. Despite the fact that it possesses at least $N^{0.9}$ distinct collisions, a nearly complete partial assignment that only allocates the holes for N pigeons might fail to witness any solution, especially if they form a permutation of $[N]$. Consequently, as per Definition 20, $\textsc{Pigeon}(N + N^{0.9})$ does not satisfy the criteria for being *robust*.

This outcome aligns with our expectations, given that $\textsc{Pigeon}(N+N^{0.9})$ is indeed PPP^{dt}-complete through a copy argument: by copying any given PIGEON instance – comprising $M + 1$ pigeons and M holes – M^9 times, we get an instance of $\textsc{Pigeon}(N + N^{0.9})$ where $N = M^{10}$. This implies that all collisions in the $\textsc{Pigeon}(N + N^{0.9})$ instance potentially map to a single collision in the original PIGEON instance. One can check that this copy argument does not work for WEAKPIGEON.

For similar reasons, END-OF-LINE *with multiple sources* [19] – a problem shown to be PPAD-complete – stands as another example of a problem with plenty of solutions yet not technically *robust*.

3.2 Members of TFAP^{dt}

▶ **Theorem 22.** $\mathsf{PWPP}^{dt} \subseteq \mathsf{TFAP}^{dt}$.

Proof. It suffices to show WeakPigeon^{dt} is *robust*. In a WeakPigeon^{dt} instance, each pigeon takes $\log N$ bits to encode the hole it gets mapped to. Now fix any partial assignment x, a pigeon is *eradicated* in x if one of the bits belonging to that pigeon is not assigned in x. Therefore, if x is nearly complete, only a negligible fraction of pigeons are eradicated. Use the weak pigeonhole principle again, there must be a collision by two remaining pigeons. ◀

▶ **Theorem 23.** *The Yamakawa-Zhandry's problem is in* TFAP^{dt}.

Proof. We follow the same proof framework as Theorem 22. Fixing any nearly complete partial assignment x, a codeword c is eradicated in x if there is a character w that appears in c that the value $f(w)$ is not assigned in x. Using Fact 19, only a negligible fraction of codewords are eradicated in a nearly complete assignment x.

Now assume a λ-wise independent function h_k is randomly drawn. Let f be any mapping that is consistent with x. Since $\lambda > n$, we have $\Pr[f(c) \oplus h_k(c) = 0^n] \geq 2^{-n}$ for any codeword c not eradicated in x. Define set of codewords

$$C_{k,x} \coloneqq \{c \in C : f(c) \oplus h_k(c) = 0^n \wedge c \text{ is not eradicated in } x\}.$$

By the setting of parameters, we have

$$\mathbf{E}[|C_{k,x}|] \geq (|C| \cdot (1 - \mathrm{negl}(n))) \cdot 2^{-n} \gg t.$$

Finally, by the *averaging principle*, there exists a key k with $|C_{k,x}| \geq t$, i.e., there will be a solution witnessed by x. This concludes that the Yamakawa-Zhandry's problem is robust. ◀

▶ **Theorem 24.** TFZPP$^{dt} \subseteq$ TFAP^{dt}.

Proof. Consider a TFZPP^{dt} problem R_N solved by a distribution D of decision trees $(T_r), r \in D$. Observe that R_N can be many-one reduced to a problem Q_N of finding a correct random seed r, i.e., $(x, r) \in Q_N \Leftarrow (x, T_r(x)) \in R_N$. It suffices to show that Q_N is in TFAP^{dt} regarding Definition 20.

To this end, we reduce Q_N to a robust problem S_N. For simplicity, we assume all decision trees V_r which check the validity of the seed r in Q_N have the same depth d. The input of S_N consists of d-bits blocks (B_r) for each random seed $r \in D$. Let $B_r(i)$ be the i-th bit of B_r. A block B_r is interpreted as a partial assignment of Q_N that corresponds to a path in V_r. More specifically, the path is generated from up to down, at the i-th level of V_r, it queries a bit (in the input of Q_N) and receives answer $B_r(i)$.

There are two types of solutions to S_N:

1. a seed r that B_r encodes an accepting path of V_r, i.e., B_r encodes a witnessing partial assignment of Q_N;
2. two seeds r_1, r_2 that the partial assignment corresponding to B_{r_1}, B_{r_2} are not consistent.

The reduction from Q_N to S_N is straightforward. For each r, B_r is generated by running the decision tree V_r on the input of Q_N. There is no type-2 solution of S_N in this case, and any type-1 solution of S_N is also a solution to Q_N.

It remains to show that S_N is robust. For any nearly complete partial assignment x, $(1 - \mathrm{negl}(\log N))$ fraction of blocks (B_r) are entirely assigned in x since d is $\mathrm{poly}(\log N)$. Suppose there is no type-2 solution witnessed by x, then there exists an input y of Q_N such that y is consistent with all blocks (B_r) contained in x. Note that at least half of the blocks (B_r) in an instance of S_N are witnessing when it is reduced from Q_N. Therefore, at least one of the witnessing B_r is entirely contained in x. This concludes that S_N is robust. ◀

4 Semi-Gluability and Lean TFNP Classes

In this section, we prove our main theorem that strongly separates the majority of the TFNP^{dt} subclasses from TFAP^{dt}.

▶ **Theorem 6.** *There is no randomized many-one reduction from* PPA^{dt}, PPAD^{dt}, PPADS^{dt}, PLS^{dt}, PPP^{dt}, CLS^{dt}, SOPL^{dt} *or* UEOPL^{dt} *to any problems in* TFAP^{dt}.

Our technique can be viewed as a generalization of the proof by Jain et al. [27], which separates PPP from n-PWPP. First, we generalized the concept of *gluability* defined in [22] (implicitly used in [4]) to distributional problems, and we call it *semi-gluability*. We formally define semi-gluability and establish our major tool theorem in Section 4.1.

▶ **Theorem 3.** *No many-one reduction exists from any semi-gluable* TFNP^{dt} *problem to any* TFAP^{dt} *problem. Consequently, all semi-gluable problems are lean.*

Then, we prove that all classes mentioned in Theorem 6 are semi-gluable.

▶ **Theorem 4.** PPA^{dt}, PPAD^{dt}, PPADS^{dt}, PLS^{dt}, PPP^{dt}, CLS^{dt}, SOPL^{dt}, *and* UEOPL^{dt} *are* semi-gluable.

We present a straightforward proof for the semi-gluability for PLS^{dt}, CLS^{dt}, SOPL^{dt}, and UEOPL^{dt} in Section 4.2. The proof for other classes is even simpler, hence we have relegated them Appendix A. Theorems 3 and 4 already imply a weaker form of Theorem 6, which rules out deterministic many-one reduction.

It is worth noting that UEOPL^{dt} is a subclass of all other classes mentioned in Theorem 6 [17]. Therefore, only considering UEOPL^{dt} suffices to prove Theorem 6. Nevertheless, Theorem 4 underscores the independent interest in the concept of semi-gluability, as it captures a significant portion of lean problems.

Finally, we complete the proof of Theorem 6 by ruling out randomized many-one reductions from UEOPL^{dt} to any TFAP^{dt} problems in Section 4.3. In this step, we apply a "double-dice" trick to bound the success probability of any randomized reduction, which also appears in the concurrent work by Jain et al. [27] (Section 3.2).

4.1 Definition of Semi-Glueability

Our formalization of *semi-gluability* is built upon the work of Göös et al. ([22], Section 7.1). The key modification here is that we are defining *gluability* in relation to a specific distribution of inputs.

Let $\mathrm{R} = (R_N), R_N \subseteq \{0,1\}^N \times O_N$ be a TFNP^{dt} problem. For a given distribution μ over the inputs of R_N, we use $D_\mu(R_N|x)$ to denote the distributional query complexity[6] of R_N conditional on the partial assignment x. For a witnessing partial assignment x, $D_\mu(R_N|x) = 0$. We say a partial assignment x being *good* with respect to μ if $D_\mu(R_N|x)$ is bounded by $\mathrm{poly}(\log N)$, and *bad* otherwise. Furthermore, two partial assignments $x, y \in \{0,1,*\}^N$ are *consistent* with respect to μ if there exists an input z in the support of μ such that both x and y agree with z on all non-$*$ bits.

[6] Without loss of generality, we assume the distributional query complexity is defined in a way that the algorithms have to succeed with probability at least $1/\mathrm{poly}(\log N)$, following Definition 13. The success rate can be boosted up via repetition for TFNP^{dt} problems.

At a high level, an NP search problem is *μ-gluable* if, given any small collection of consistent, bad (and "well-behaved") partial assignments of $\text{poly}(\log N)$-size, their union is still *bad*.

We now defined a class of "well-behaved" partial assignments called *trimmed assignments*[7].

▶ **Definition 25** (Trimmed Assignments). *Let $(T_x), x \in \{0,1,*\}^N$ be a family of low-depth decision trees over $\{0,1\}^N$, where each tree corresponds to a partial assignment x in $\text{poly}(\log N)$-size.*

Given a (complete) input $z \in \{0,1\}^N$, the "trimmed assignment of x by T_x" is the union of x and the leaf of T_x that is reached by following a path consistent with the bits specified in z.

In other words, the trimmed assignment of x contains additional information derived from the input z that complements the partial information in x, thereby extending x to a more "complete" assignment while maintaining consistency with z. It is helpful to think the whole input z is unknown and it is generated on-the-fly during the trimming process. Therefore, we usually use a trimmed assignment without specifying the whole input z.

As an example of trimming, consider problems defined on non-binary alphabets $[N]$, like PIGEON^{dt}. Here, each element of $[N]$ is represented using $\log N$ bits. We define a partial assignment x to be *trimmed* with respect to the alphabet $[N]$ under the following condition: if any bit representing a character w in $[N]$ is assigned in x, then all bits representing w must be assigned in x. To facilitate this, we introduce a low-depth decision tree T_x, which can process a partial assignment x of size $\text{poly}(\log N)$ over the alphabet $[N]$. This tree operates by querying all bits associated with any partially specified character in x to obtain a trimmed assignment.

It is important to note that the set of decision trees (T_x) is utilized exclusively for query purposes to attain a trimmed assignment; the actual outputs of individual trees in this set are not of concern in this context.

▶ **Definition 26** (μ-Gluable and Semi-Gluable). *Let $\mathrm{R} = (R_N)$ be a TFNP^{dt} problem and $\mu = (\mu_N)$ be a sequence of distributions over $\{0,1\}^N$.*

We define R *to be* μ-gluable *under the following conditions:*

- *For an infinite series of N, the distributional query complexity $D_{\mu_N}(R_N)$ is greater than any $\text{poly}(\log N)$ functions.*
- *There exists a family of decision trees (T_x), such that for any collection $\mathcal{P}$ of partial assignments that adhere to the following:*
  - *The set $\mathcal{P}$ contains $\text{poly}(\log N)$ assignments.*
  - *All partial assignments in $\mathcal{P}$ are consistent with each other in terms of μ_N.*
  - *Each partial assignment in $\mathcal{P}$ is trimmed by (T_x), has a size of $\text{poly}(\log N)$, and is bad with respect to μ_N.*

  the union of all partial assignments in $\mathcal{P}$ remains bad with respect to μ_N.

A TFNP^{dt} problem R *is* semi-gluable *if there exists a sequence of distributions $\mu = (\mu_N)$ that* R *is μ-gluable.*

We say a syntactic TFNP^{dt} subclass is semi-gluable if any one of its complete problems is semi-gluable.

[7] Trimmed assignment is a simplified notion of "completions" defined in Göös et al. ([22], Definition 9).

Now we are ready to prove the main Theorem 3. Broadly speaking, in the process of reducing an average-case hard problem R to a *robust* problem Q, a majority of the bits in Q yield a bad partial assignment of R. By removing all "good" bits in the input of Q, which is a small fraction, the remaining input bits still witness a solution o of problem Q since Q is *robust*. Now, all the bits used to verify solution o are "bad", and so the union of them is still bad by the μ-gluability of R. Consequently, solution o provides little help to solving the original instance of R.

Proof of Theorem 3. Let $\mathrm{R} = (R_N)$ be any semi-gluable TFNP^{dt} problem, i.e., there is a sequence of distributions $\mu = (\mu_N)$ for which R is μ-gluable. Assume a family of decision trees (T_x) is used to trim each partial assignment in the condition of the μ-gluability.

Let $\mathrm{Q} = (Q_M)$ be any *robust* TFNP^{dt} problem. It suffices to show that when R_N instances are drawn from distribution μ_N, any (deterministic) decision tree reduction $(f_i, g_o), i \in M, o \in O'$ from $R_N \subseteq \{0,1\}^N \times O$ to $Q_M \subseteq \{0,1\}^M \times O'$ may make mistakes with non-zero probability.

For each bit $i \in [M]$, we construct a new decision tree f_i^T from f_i: for each leaf node p of f_i, which corresponds to a partial assignment $x \in \{0,1,*\}^N$, we attach the decision tree T_x to the bottom of node p, while each leaf of T_x still outputs the same value as p did. Note that the functionality of f_i^T is the same as f_i, and its tree depth only increases by $\mathrm{poly}(\log N)$. Now each leaf node of f_i^T corresponds to a trimmed assignment of R_N. We call a leaf node of f_i^T *good* if its corresponding partial assignment in R_N is good, and *bad* otherwise.

Let z be an instance of R_N randomly drawn from μ_N. The crucial observation is that the probability of $f_i^T(z)$ reaching an good leaf node is $\mathrm{negl}(\log N)$. Otherwise, f_i^T implies a (randomized) $\mathrm{poly}(\log N)$ query algorithm for R_N which succeeds with non-negligible probability. This contradicts to the fact that $D_{\mu_N}(R_N)$ is super-logarithmic in N.

Let γ_z be the fraction of index i over $[M]$ such that $f_i^T(z)$ reaches a good leaf node. By the linearity of expectation, we have

$$\mathbf{E}_{z\sim\mu_N}[\gamma_z] = \mathrm{negl}(\log N).$$

Then by the Markov inequality,

$$\Pr_{z\sim\mu_N}[\gamma_z = \mathrm{negl}(\log N)] = 1 - \mathrm{negl}(\log N).$$

In other words, with $(1 - \mathrm{negl}(\log N))$ probability, $(1 - \mathrm{negl}(\log N))$ fractions of bits in $f^T(z)$ are determined by a bad leaf. We call those bits from bad leaves by *bad* bits.

Now fixing an instance z of R_N with $\gamma_z = \mathrm{negl}(\log N)$, all the bad bits in $f^T(z)$ consist of a nearly complete partial assignment x. Since Q_M is *robust*, there exists a solution $o \in O'$ of the Q_M instance $f^T(z)$ that is witnessed by x. The reduction then adversarially returns this *bad* solution o.

Finally, we show that this bad solution o is indeed useless with μ-gluability. By running the decision tree V_o on the Q_M instance $f^T(z)$ to verify the solution o, we get a $\mathrm{poly}(\log N)$-size partial assignment y of input $f^T(z)$ with bad bits only. Each bit assigned in y corresponds to a bad partial assignment of z; we denote this set of partial assignments by $\mathcal{P}_o$.

Without loss of generality, we assume the algorithm (i.e., the reduction) is given $\mathcal{P}_o$ besides the solution o. The set $\mathcal{P}_o$ satisfies the conditions in Definition 26, so the union of assignments in $\mathcal{P}_o$ is bad by the μ-gluability of R. That is, the distributional complexity $D_\mu(R_N|\mathcal{P}_o)$ is still very large. Furthermore, the algorithm could only make $\mathrm{poly}(\log N)$ more queries in the decision tree g_o. Thus, this reduction will make mistakes on some inputs consistent with $\mathcal{P}_o$. ◀

4.2 Semi-Gluability of PLS^{dt}, CLS^{dt}, SOPL^{dt}, and UEOPL^{dt}

The crux of proving semi-gluablity is choosing a hard distribution of instances. As mentioned in the introduction, it makes sense to focus on instances with a unique solution. Here we prove the semi-gluability of CLS^{dt}, and the proofs for PLS^{dt}, SOPL^{dt}, and UEOPL^{dt} follow from the same argument.

We work on problem EoPL^{dt} (Definition 16), which is complete for CLS^{dt} [21]. Recall that in EoPL^{dt}, a graph with N vertices is specified by a predecessor function P, a successor function S, and a value function V. We construct a distribution $\mu = (\mu_N)$ for EoPL^{dt} instances as follow: For each N, let $M = \lfloor\sqrt{N}\rfloor$. We uniformly pick a subset L' of $M-1$ vertices from $[N]\backslash\{1\}$ and let $L = L' \cup \{1\}$. Vertices in L form a random line starting from 1 with the value monotone increasing by 1 in each step; all other vertices are isolated points with value 1.

Formally, a random order $\sigma : [M] \to L$ is uniformly chosen conditional on $\sigma(1) = 1$. We have,

1. $S(\sigma(i)) = \sigma(i+1), \forall i \in [M-1]$; $S(\sigma(M)) = \sigma(M)$;
2. $P(\sigma(i)) = \sigma(i-1), \forall i \in \{2, \ldots, M\}$; $P(1) = 1$;
3. $S(u) = P(u) = u, \forall u \notin L$;
4. $V(\sigma(i)) = i, \forall i \in [M]; V(u) = 1, \forall u \notin L$.

Göös et al. ([22], Lemma 15) provided a family of decision trees (T_x) for trimming EoPL^{dt} instances. While instances in the support of μ allow us to use a simpler trimming method: Assume the inputs are encoded in alphabet $[N]^3$, where each character in the input encodes a triplet $(S(u), P(u), V(u))$ for some vertex u. We let each partial assignment trimmed over the alphabet $[N]^3$. Now for a trimmed partial assignment x, we say a vertex u is *queried* by x (written as $u \in Q_x$) if the triplet $(S(u), P(u), V(u))$ is assigned in x; a vertex u is *revealed* by x if u is queried, or u appears as the predecessor or successor in the queries made in x.

For every EoPL^{dt} instance in μ, the unique solution is hidden at the end of the line L. Given a trimmed partial assignment x, a score function $s(x)$ is defined as the distance to the end of the line L from the furthest points queried in x, i.e., $s(x) = M - \max_{u \in Q_x} V(u)$. We claim that $s(x)$ decides the distributional query complexity of EoPL^{dt} conditional on x.

▶ **Lemma 27.** *For any trimmed partial assignment x of size* poly$(\log N)$*, we have*

$$D_\mu(\text{EoPL}_N | x) = \Theta(s(x)).$$

The semi-gluability of CLS^{dt} follows from Lemma 27. Observe that a partial assignment is bad if and only if its score function is large by Lemma 27. As the score function of the union of multiple partial assignments only depends on the one with the smallest score function, the union is still bad if each individual partial assignment is bad.

Finally, we present of proof of Lemma 27, which is rather straightforward by exploiting the symmetrical nature of the distribution μ.

Proof of Lemma 27. $D_\mu(\text{EoPL}_N | x) \le s(x)$ is implied by the simple line-tracing algorithm, which keeps querying the successor of the current furthest point on the line, until reaching the end.

It remains to show that $D_\mu(\text{EoPL}_N | x) = \Omega(s(x))$. Consider any deterministic algorithm A with $(1 - \text{negl}(\log N))$ success rate. W.l.o.g., we assume A outputs a solution only when the current score function is 0.

A has two possible choices in each step: query a vertex that has been revealed, or has yet been revealed. In the first case, the current score function will decrease by 1 or stay unchanged.

In the latter case, by the symmetry of μ, A will hit a vertex in L with probability at most

$$M/(N - \text{poly}(\log N)) = O(\frac{1}{\sqrt{N}}).$$

In which case, the current score function will decrease by at most $s(x)$; if A hits an isolated point, $s(x)$ remains unchanged.

Combining the analysis for both cases, A requires at least $O(s(x))$ steps to achieve a high success rate. ◀

▶ Remark 28. Hubácek and Yogev [26] provided a hard distribution of CLS^{dt} instances[8] indirectly by reducing from a distribution of *local search* instances with *staircase* construction [44]. In comparison, our proof enables a more fine-grained analysis of the distributional query complexity condition on partial assignments.

4.3 On Randomized Reductions

Built upon Section 4.1, 4.2, we present a more fine-grained analysis in this section to further rule out any randomized many-one reduction from UEOPL^{dt} to any TFAP^{dt} problem. Due to its similarity with [27] (Section 3.2), we only provide a sketch for several proofs in this section.

Recall that the distribution μ of EOPL^{dt} instances defined in Section 4.2 contain a unique solution hidden at the end of the only one path. Therefore, μ is also a distribution of UEOPL^{dt} instances with the same set of solution by Definition 16, 17, and UEOPL^{dt} is μ-gluable.

Again, we fix $\text{Q} = (Q_M)$ to be any *robust* TFNP^{dt} problem. By Yao's Minimax principle, it suffices to show that when UEOPL instances are drawn from the distribution $\mu = (\mu_N)$, any (deterministic) decision tree reduction $(f_i, g_o), i \in M, o \in O'$ from UEOPL_N to $Q_M \subseteq \{0,1\}^M \times O'$ is correct with at most $\text{negl}(\log N)$ probability. Let $d = d(N)$ be the depth of this reduction.

Without loss of generality, we assume each query made in decision trees (f_i, g_o) is querying a triplet $(S(u), P(u), V(u))$ for a vertex u. This only incurs an $O(\log N)$ factor of overheads to the depth of the reduction, and by this, we could assume all the partial assignments are trimmed.

When (g_o) are depth-0

We say a reduction is depth-k ($k \le d$) if all the decision trees (g_o) are depth-k, while (f_i) are still depth-d. We start with a special case with depth-0 reductions, i.e., $g_o \in [N]$ is the vertex that is claimed to be the unique end of the line in the UEOPL^{dt} instance.

Denote the unique solution of the UEOPL^{dt} instance z by u^*. The following lemma bound the success probability of any depth-0 reductions.

▶ **Lemma 29.** *For any depth-0 reduction (f_i, g_o),*

$$\Pr_{z \sim \mu_N}[g_o = u^* \Leftarrow (f(z), o) \in Q_M] < \text{negl}(\log N).$$

[8] Precisely, the problem considered in [26] is called EOML^{dt} (End-of-Metered-Line), which is equivalent to EOPL^{dt} as shown in Fearnley et al. [17].

We apply a *double-dice* trick to prove Lemma 29: Let $z^{(1)} \sim \mu_N$ to be a random instance of UEoPL^{dt} with solution u^*. We generate a second instance $z^{(2)}$ from $z^{(1)}$ by uniformly chosen a vertex $v^* \in [N] \backslash \{1\}$, and swap u^* and v^*. Note that the marginal distribution of $z^{(2)}$ is also μ_N. The advantage of using two dice comes from the following fact.

▶ **Fact 30.** *For any instance $z^{(1)}$ and any bad solution o of $f(z^{(1)})$, o is also a bad solution of $f(z^{(2)})$ with probability $1 - \mathrm{negl}(\log N)$.*

Proof. Since o is bad, vertex u^* is not revealed in the set of partial assignments $\mathcal{P}_o$. Also, $z^{(2)}$ is different to $z^{(1)}$ locally on u^* and v^*. Therefore, with $1 - \mathrm{negl}(\log N)$ probability on v^*, the union of $\mathcal{P}_o$ is also a partial assignment of $z^{(2)}$, and o is a bad solution of $z^{(2)}$. ◀

Now we are ready to prove Lemma 29.

Proof Sketch of Lemma 29. We bound the success probability of the reduction on $z^{(2)}$ indirectly with the help of $z^{(1)}$.

For a random $z^{(1)}$ drawn from μ_N, there are two possibilities: either $f(z^{(1)})$ has a bad solution, or $f(z^{(1)})$ does not have any bad solution. The second case will happen with $\mathrm{negl}(\log N)$ probability by the proof of Theorem 3, so we simply assume the reduction is correct in this case.

Now assume $f(z^{(1)})$ has a bad solution o^*. From Fact 30, we know that o^* is also a bad solution of $f(z^{(2)})$ with high probability; however, the reduction will output a fixed solution g_{o^*} on those $z^{(2)}$. In other words, the reduction could only be correct in the case that $v^* = g_{o^*}$. Therefore, the reduction will be wrong on $z^{(2)}$ with $1 - \mathrm{negl}(\log N)$ probability condition on such kind of $z^{(1)}$.

Combining both cases, we prove that the success probability of any depth-0 reduction is $\mathrm{negl}(\log N)$. ◀

When (g_o) are depth-k

We first illustrate the idea for $k = 1$, and then generalize the proof to any $k \le d$. Now consider a "triple-dice" method by generating three random instances $z^{(1)}, z^{(2)}, z^{(3)}$ in order:

1. $z^{(1)}$ is randomly drawn from μ_N with solution u^*;
2. $z^{(2)}$ is generated from $z^{(1)}$ by randomly permute the last two vertices $u^*, P(u^*)$ on the line in $z^{(1)}$; let v^* be the solution of $z^{(2)}$.
3. $z^{(3)}$ is generated from $z^{(2)}$ by randomly permute the last vertex v^* on the line in $z^{(2)}$; let w^* be the solution of $z^{(3)}$.

For each solution o of Q, denote the only query made by g_o by $q_o \in [N]$. W.l.o.g., we assume $q_o = u^*$ when o is a good solution of $f(z)$. The following lemma shows that q_o will not hit the critical part of inputs with high probability.

▶ **Lemma 31.** *For any depth-1 reduction (f_i, g_o),*

$$\Pr_{z \sim \mu_N}[q_o \in \{u^*, P(u^*)\} \Leftarrow (f(z), o) \in Q_M] < \mathrm{negl}(\log N).$$

Proof Sketch. Observe that the statement in Lemma 31 is almost the same as Lemma 29. Therefore, we could deem q_o as an output from a depth-0 reduction, and the reduction succeeds if its output q_o either hits u^* or $P(u^*)$.

We can replace the second dice in the double-dice argument by $z^{(2)}$, where both $u^*, P(u^*)$ are permuted. It is easy to check that Fact 30 still holds, and the rest of the argument is the same as Lemma 29. ◀

Note that the marginal distribution of $z^{(3)}$ is also μ_N, therefore, we could consider the success probability of reduction (f_i, g_o) on $z^{(3)}$.

▶ **Lemma 32.** *For any depth-1 reduction* (f_i, g_o)*,*

$$\Pr_{z^{(1)},z^{(2)},z^{(3)}}[g_o(z^{(3)}) = w^* \Leftarrow (f(z^{(3)}), o) \in Q_M] < \mathrm{negl}(\log N).$$

Proof Sketch. The proof strategy is similar to Lemma 29.

For a random $z^{(1)}$ drawn from μ_N, we still consider two possibilities: either $f(z^{(1)})$ has a bad solution o^* that $q_{o^*} \notin \{u^*, P(u^*)\}$, or $f(z^{(1)})$ does not have any solution of this kind. The second case will happen with $\mathrm{negl}(\log N)$ probability by Lemma 3, so we could assume the reduction is correct in the second case.

Now assume $f(z^{(1)})$ has a bad solution o^* that $q_{o^*} \notin \{u^*, P(u^*)\}$. Generalizing Fact 30, we can show that with high probability, o^* is also a bad solution of $f(z^{(2)}), f(z^{(3)})$, and $q_{o^*} \notin \{v^*, P^{(2)}(v^*), w^*, P^{(3)}(w^*)\}$, where $P^{(2)}, P^{(3)}$ are the predecessor functions for $z^{(2)}, z^{(3)}$ respectively. In this case, we always have $g_{o^*}(z^{(2)}) = g_{o^*}(z^{(3)})$, and this reduction could only be correct when $w^* = g_{o^*}(z^{(2)})$, which happens with very low probability. Note that there may be $g_{o^*}(z^{(1)}) \neq g_{o^*}(z^{(2)})$, since the vertex $P(P(u^*))$ may be queried by q_{o^*}, which reveals the value of $P(u^*)$.

Combining both cases, we conclude that the success probability of any depth-1 reduction is $\mathrm{negl}(\log N)$. ◀

Finally, we complete the proof by generalizing to depth-k reductions.

Proof Sketch of Theorem 4. When (g_o) are depth-k, we will use $(k+2)$ dice $z^{(1)}, \ldots, z^{(k+2)}$. For each $i > 1$, $z^{(i)}$ is generated from $z^{(i-1)}$ by randomly permute the the last $k+3-i$ vertices on its path.

Intuitively, after making i $(0 \leq i \leq k)$ queries in g_o, the last $k+1-i$ vertices in $z^{(i+2)}$ are not revealed with high probability. Therefore, the reduction has to make a blind guess for the solution of $z^{(k+2)}$ in the end, which will be wrong with high probability. ◀

5 TFAP in the White-Box Setting

In this section, we turn our attention to the definition of TFAP and its implications in the white-box setting. While Definition 20 (of "abundant") is fundamentally crafted for the query model, it necessitates an indirect approach to define TFAP within the white-box setting. Leveraging the insights shared in Section 2.3, we establish that every syntactic subclass of TFNP in the white-box setting inherently mirrors a syntactic subclass in the TFNP^{dt} domain.

▶ **Definition 33.** *A syntactic subclass of* TFNP*, denoted as* C*, is classified under* TFAP *if its corresponding syntactic* TFNP^{dt} *subclass* C^{dt} *is contained in* TFAP^{dt}*.*

An unnatural TFNP *problem, denoted as* R*, is classified under* TFAP *if its corresponding syntactic* TFNP^{dt} *problem* R^{dt} *is contained in* TFAP^{dt}*.*

A natural TFNP *problem is contained in the white-box closure of* TFAP *if it can be many-one reduced to an unnatural problem in* TFAP*.*

This definition serves as a bridge, facilitating the exploration of TFAP in the white-box setting by drawing upon its established characteristics in the TFNP^{dt} framework. In particular, the inclusion of PWPP, Yamakawa-Zhandry's problem, and any unnatural TFZPP problem (Theorem 22,23,24) can be cast into the white-box world. Like TFNP, TFAP is also a *semantic* class, and thus we do not expect it to have a complete problem.

Note that by its definition, TFAP is closed under relativized (black-box) many-one reductions. However, it is not necessarily closed under non-relativized (white-box) many-one reductions. Despite this, the known reductions between white-box syntactic TFNP subclasses are relativized. This means that while we should use TFAP (and its white-box closure) cautiously as a barrier separating white-box TFNP problems, finding a non-relativized result that bypasses the "TFAP barrier" would be a significant development.

Looking at the FACTORING problem offers an interesting perspective. Jeřábek [28] demonstrated that FACTORING falls under PWPP through a randomized reduction, and according to Definition 33, PWPP is a subset of TFAP. Although at first sight FACTORING does not appear to be "abundant" due to the limited number of solutions in many inputs, the randomized reduction to WEAKPIGEON enables us to employ a trick similar to that in Lemma 21 to create an equivalent version of FACTORING with plenty of solutions. Moreover, due to the famous Shor's algorithm [39], FACTORING is a part of FBQP. Recall that our earlier speculation regarding the potential subset relationship between TFBQP^{dt} and TFAP^{dt}, the classification of FACTORING within TFAP (through a randomized reduction) might not be just a coincidence.

6 Future Directions

By examining the landscape of TFNP through the prism of the abundance of solutions, we have delineated a distinct divide that separates the TFNP universe into the domains of TFAP and lean classes. This fresh perspective opens up a rich vein of potential explorations both within and possibly extending beyond the boundaries of TFNP.

Identifying more problems in TFAP

One of the most fascinating questions is the potential inclusion of TFBQP^{dt} in the realms of TFAP^{dt}; it immediately implies oracle separations from TFBQP to the majority of TFNP subclasses via our framework.

▶ **Open Problem 1.** *Is* $\mathsf{TFBQP}^{dt} \subseteq \mathsf{TFAP}^{dt}$*?*

To shed more light on this, the exponential speed-up achieved by a quantum query algorithm is attributed to the querying of certain quantum states in a superposition of exponentially many locations. In that case, a small fraction of input bits should not affect the quantum algorithm's outcome by much, a notion that resonates well with the essence of our TFAP^{dt} definition.

$t(n)$-RAMSEY is another problem that we suspect might belong to TFAP. In this problem, we are given a graph with 2^n vertices and the task is to find either a clique or an independent set of size $t(n)$.

▶ **Open Problem 2.** *Does* $\frac{n}{2}$-RAMSEY *belong to* TFAP*? If not, what about* $\frac{n}{10}$-RAMSEY*?*

The reason we consider this problem is not just because it has many solutions; its totality, based on the *weak iterated pigeonhole principle*, aligns with the principles underlying the inclusion of WEAKPIGEON and Yamakawa-Zhandry's problem in TFAP, which are based on the *weak pigeonhole principle*. A new class, PLC, was introduced by Pasarkar et al. [36] to capture $\frac{n}{2}$-RAMSEY, and they demonstrated that PPP $\subseteq$ PLC, thereby proving that PLC is also a lean class. If $\frac{n}{2}$-RAMSEY is indeed contained in TFAP, it automatically indicates an oracle separation between PLC (or even PPP) and $\frac{n}{2}$-RAMSEY.

Proof complexity, cryptography, and others

We have a few more broad questions concerning TFAP.

1. Can we find a natural counterpart of TFAP^{dt} in proof complexity?
2. Theorem 3 already shows that no randomized many-one reduction exists from a semi-gluable problem to TFAP^{dt}. Could we extend this to rule out even stronger reductions, especially those commonly used in cryptography? Moreover, aside from the known distinctions between collision-resistant hash functions and one-way permutations, could this framework help to explain more separations between cryptographic primitives, or even help to find new ones?
3. Are there other sensible ways to formalize TFNP problems that have a large number of non-trivially distinct solutions?
4. Could we extend our study to problems outside of TFNP, such as *Range-Avoidance* problem [30], which clearly has a plethora of distinct solutions?
5. Can we unify the techniques used in this work and in Müller [34].

References

1 Scott Aaronson. Lower bounds for local search by quantum arguments. *SIAM J. Comput.*, 35(4):804–824, 2006. doi:10.1137/S0097539704447237.

2 Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1330–1342. ACM, 2021. doi:10.1145/3406325.3451047.

3 Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004. doi:10.1145/1008731.1008735.

4 Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. In Frank Thomson Leighton and Allan Borodin, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 303–314. ACM, 1995. doi:10.1145/225058.225147.

5 Simina Brânzei and Jiawei Li. The query complexity of local search and brouwer in rounds. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory, 2-5 July 2022, London, UK*, volume 178 of *Proceedings of Machine Learning Research*, pages 5128–5145. PMLR, 2022. URL: https://proceedings.mlr.press/v178/branzei22a.html.

6 Josh Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*, pages 54–67. IEEE Computer Society, 2004. doi:10.1109/CCC.2004.1313795.

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

8 Xi Chen and Xiaotie Deng. Matching algorithmic bounds for finding a brouwer fixed point. *J. ACM*, 55(3):13:1–13:26, 2008. doi:10.1145/1379759.1379761.

9 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):14, 2009.

10 Xi Chen, Yuhao Li, and Mihalis Yannakakis. Reducing tarski to unique tarski (in the black-box model). In Amnon Ta-Shma, editor, *38th Computational Complexity Conference, CCC 2023, July 17-20, 2023, Warwick, UK*, volume 264 of *LIPIcs*, pages 21:1–21:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CCC.2023.21.

11 Xi Chen, Xiaoming Sun, and Shang-Hua Teng. Quantum separation of local search and fixed point computation. *Algorithmica*, 56(3):364–382, 2010. doi:10.1007/s00453-009-9289-0.

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

13 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.

14 Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis. The complexity of constrained min-max optimization. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1466–1478. ACM, 2021. doi:10.1145/3406325.3451125.

15 Susanna F. de Rezende, Mika Göös, and Robert Robere. Guest column: Proofs, circuits, and communication. *SIGACT News*, 53(1):59–82, 2022. doi:10.1145/3532737.3532746.

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

17 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *J. Comput. Syst. Sci.*, 114:1–35, 2020. doi:10.1016/j.jcss.2020.05.007.

18 Aris Filos-Ratsikas and Paul W Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 638–649, 2019.

19 Paul W Goldberg and Alexandros Hollender. The hairy ball problem is ppad-complete. *arXiv preprint*, 2019. arXiv:1902.07657.

20 Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 127–138. ACM, 2013. doi:10.1145/2422436.2422453.

21 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.33.

22 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1150–1161. IEEE, 2022. doi:10.1109/FOCS54457.2022.00111.

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

24 Mika Göös and Aviad Rubinstein. Near-optimal communication lower bounds for approximate nash equilibria. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 397–403. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00045.

25 Pavel Hubácek, Chethan Kamath, Karel Král, and Veronika Slívová. On average-case hardness in TFNP from one-way functions. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 614–638. Springer, 2020. doi:10.1007/978-3-030-64381-2_22.

26 Pavel Hubácek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. *SIAM J. Comput.*, 49(6):1128–1172, 2020. doi:10.1137/17M1118014.

27 Siddhartha Jain, Jiawei Li, Robert Robere, and Zhiyang Xun. On Pigeonhole Principles and Ramsey in TFNP. In submission, 2023.

28 Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016.

29 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.

30 Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.44.

31 Veronika Královská. On the complexity of search problems with a unique solution. *Charles University Digital Repository*, 2021.

32 Nimrod Megiddo and Christos H Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

33 Tsuyoshi Morioka. Classification of search problems and their definability in bounded arithmetic. *Electron. Colloquium Comput. Complex.*, TR01-082, 2001. arXiv:TR01-082.

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

35 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.

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

37 Pavel Pudlák. On the complexity of finding falsifying assignments for herbrand disjunctions. *Arch. Math. Log.*, 54(7-8):769–783, 2015. doi:10.1007/s00153-015-0439-6.

38 Alon Rosen, Gil Segev, and Ido Shahaf. Can ppad hardness be based on standard cryptographic assumptions? In *Theory of Cryptography Conference*, pages 747–776. Springer, 2017.

39 Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332, 1999. doi:10.1137/S0036144598347011.

40 Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998. doi:10.1007/BFb0054137.

41 Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. Ppp-completeness with connections to cryptography. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 148–158. IEEE, 2018.

42 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

43 Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 69–74. IEEE, 2022. doi:10.1109/FOCS54457.2022.00014.

44 Shengyu Zhang. Tight bounds for randomized and quantum local search. *SIAM J. Comput.*, 39(3):948–977, 2009. doi:10.1137/06066775X.

A Semi-Gluability of PPP^{dt}, PPA^{dt}, PPAD^{dt}, and PPADS^{dt}

We show the semi-gluability of PPP^{dt}, PPA^{dt}, PPAD^{dt}, and PPADS^{dt} in this section; together with Section 4.2, we conclude of the proof of Theorem 4.

Let us start from the semi-gluability of PPP^{dt}. Consider the following distributions $\mu = (\mu_N)$ for PIGEON instances: For each N, we have a function f that randomly assigns the first N pigeons to N holes, according to a uniformly random permutation; the $(N+1)$-th pigeon is always placed to the first hole. In this case, the only solution is given by the other pigeon $i^* \in [N]$ which also gets mapped to hole 1 besides pigeon $(N+1)$.

Assume all the partial assignments are trimmed with respect to the alphabet $[N]$. Using the symmetric nature of the distribution μ, it is not hard to show that a partial assignment x of $\mathrm{poly}(\log N)$ size is informative iff x is witnessing, i.e., x assigns a pigeon $i^* \in [N]$ that $f(i^*) = 1$. Therefore, the union of any $\mathrm{poly}(\log N)$ uninformative partial assignment is still uninformative. According to Definition 26, PPP^{dt} is semi-gluable.

The semi-gluablity of PPA^{dt}, PPAD^{dt}, and PPADS^{dt} follows from a similar approach. We give a sketch here.

Inspired by the reduction from PPADS^{dt} to PPP^{dt}, each PPP^{dt} instance drawn from $\mu = (\mu_N)$ also corresponds to an instance of PPA^{dt}, PPAD^{dt}, or PPADS^{dt}, where the only solution corresponds to the collision. The extra structure information in PPA^{dt}, PPAD^{dt}, or PPADS^{dt} instances provides no advantages, as one can still use symmetry to show that a partial assignment is informative only if x is witnessing. Therefore, PPA^{dt}, PPAD^{dt}, and PPADS^{dt} are also semi-gluable.