Electronic Colloquium on Computational Complexity, Report No. 157 (2025)

Recovery Reductions, Conjectures, and Barriers

Tejas Nareddy* Abhishek Mishra†

2025/09/06

Abstract

We introduce and initiate the study of a new model of reductions called the *random noise model*. In this model, the truth table $\mathtt{T}_f$ of the function f is corrupted on a randomly chosen δ-fraction of instances. A randomized algorithm $\mathcal{A}$ is a $(t, \delta, 1-\varepsilon)$-recovery reduction for f if:

1. With probability $1-\varepsilon$ over the choice of δ-fraction corruptions, given access to the corrupted truth table, the algorithm $\mathcal{A}$ computes $f(\phi)$ correctly with probability at least $2/3$ on every input ϕ.
2. The algorithm $\mathcal{A}$ runs in time $O(t)$.

This model, a natural relaxation of average-case complexity, has practical motivations and is mathematically interesting.

Pointing towards this, we show the existence of robust deterministic polynomial-time recovery reductions with optimal parameters up to polynomial factors (that is, deterministic $\left(poly(n), 0.5 - 1/poly(n), 1 - e^{-\Omega(poly(n))}\right)$-recovery reductions) for a large function class SLNP^S containing many of the canonical NP-complete problems - SAT, kSAT, kCSP, CLIQUE and more. As a corollary, we obtain that the barrier of Bogdanov and Trevisan (2006) for non-adaptive worst-case to average-case reductions does not apply to our mild non-adaptive relaxation.

Furthermore, we establish recovery reductions with optimal parameters for *Orthogonal Vectors* and *Parity k-Clique* problems. These problems exhibit structural similarities to NP-complete problems, with *Orthogonal Vectors* admitting a $2^{0.5n}$-time reduction from kSAT on n variables; and *Parity k-Clique* a subexponential-time reduction from 3SAT.

Keywords: Constraint Satisfaction Problems; Fine-Grained Complexity; Graph Problems; Group-Theoretic Algorithms; NP-Complete Problems; Random Noise Model; Recovery Reductions; Satisfiability.

*Khoury College of Computer Science, Northeastern University. Email: nareddy.s@northeastern.edu.

†Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, Pilani-333031, Rajasthan, India. Email: abhishek.mishra@pilani.bits-pilani.ac.in.

ISSN 1433-8092

Contents

1 Introduction

Average-case complexity deals with the complexity of solving computational problems on, say, at least an 80% fraction of instances of every size. Much work has been done in the area, showing strong results for #P and polynomials believed not to be in P (Levin, 1986; Gemmell and Sudan, 1992; Feige and Lund, 1996; Sudan, 1996; Cai et al., 1999; Sudan et al., 2001). Typically, the paradigm to prove the average-case hardness of a problem is to show a reduction from solving the problem in the worst case to solving the problem on average. Recently, owing to the explosion of interest in fine-grained complexity theory (Abboud and Williams, 2014; Williams and Williams, 2018; Williams, 2018), work has also been focused on proving the fine-grained average-case hardness of problems in P (Ball et al., 2017; Goldreich and Rothblum, 2018; Boix-Adserà et al., 2019; Kane and Williams, 2019; Goldreich, 2020; Dalirrooyfard et al., 2020; Asadi et al., 2022, 2024).

While much progress has been made on the average-case hardness of non-Boolean functions, proving the average-case hardness of computing Boolean functions poses a significant technical challenge. In the absence of large finite fields to compute our function over, we may no longer straightforwardly use tools such as the Schwartz-Zippel-DeMillo-Lipton lemma (Schwartz, 1980; Zippel, 1979; Demillo and Lipton, 1978). Hence, it is a natural question to ask what natural relaxations of average-case hardness might simultaneously be practically interesting, mathematically rich, applicable, and easier to prove for larger classes of functions (including Boolean functions). We propose a model that satisfies these criteria and provides a new approach to tackling the gap left by worst-case to average-average case reductions: the random noise model.

1.1 The Setup: Using Corrupted Truth Tables

We propose our model of "almost" worst-case to average-average case reductions. First, we define what corruption means in the random noise model.

Definition 1. **Random Noise Corruption**
Suppose $f_n : \Sigma^{p(n)} \to \mathcal{D}$[1] is the part of the function f of input length parameter n and let $\mathtt{T}_{f_n}$ denote its truth table on instances of size parameter n (length $p(n)$)[2].

We define $\mathcal{N}_\delta : \mathcal{D}^{|\Sigma|^{p(n)}} \to \mathcal{D}^{|\Sigma|^{p(n)}}$ as a random noise operator that acts as follows, acting on $\mathtt{T}_{f_n}$ as $\mathcal{N}_\delta \mathtt{T}_{f_n}$ - a subset $S \subset \Sigma^{p(n)}$ of size $\delta|\Sigma|^{p(n)}$ is chosen uniformly at random. A corrupted truth table $\mathtt{T}'_{f_n}$ is produced by modifying all entries in S in any possible way, with no restrictions on how they are changed[3], leaving all other entries unchanged.

Now, we are ready to define what it means for a function f to have a reduction in the random noise model.

[1]Here, we allow Σ to be any constant length alphabet, and $\mathcal{D}$ is any set. The variable n is the growing instance size parameter and $p : \mathbb{N} \to \mathbb{N}$ is a polynomial of constant degree.

[2]We use the phrase "truth table" even if $\mathcal{D}$ is not $\{0, 1\}$, simply out of convention. When we say the truth table, we always refer to the table of all evaluations.

[3]These changes may be adversarial, may be made to minimize the time complexity of computing the function represented by $\mathtt{T}'_f$, or to fit any other criterion.

Definition 2. **Recovery Reductions in the Random Noise Model**
For a given function $f : \Sigma^* \to \mathcal{D}$, a $(t(n), \delta, \varepsilon)$-recovery reduction in the random noise model is a randomized algorithm $\mathcal{A}$, defined as follows:

- The random noise operator $\mathcal{N}_\delta$ is applied on $\mathtt{T}_{f_n}$ to produce a corrupted truth table $\mathtt{T}'_{f_n}$. It is unknown to the algorithm which answers are corrupted.
- With probability $1 - \varepsilon$ over the randomness of choice of corruptions, $\mathcal{A}$, given oracle access to the δ-fraction corrupted truth table $\mathtt{T}'_{f_n}$, computes $f_n(x)$ correctly in $O\left(t(n)\right)$ time with probability at least $2/3$.

We emphasize that the algorithm must be correct on every inputs with probability at least $2/3$, conditioned on the $1 - \varepsilon$ measure favorable corruption due the noise operator.

We use the phrase "recovery reduction" to highlight that such an algorithm both resembles the practical recovery algorithms that operate on corrupted data and a self-reduction in some sense. Going forward, we may also call this a "recovery algorithm" when thinking of it this way seems intuitively helpful.

This is a natural, practical, and well-explored model in the context of learning from noisy examples (Angluin and Laird, 1988; Cesa-Bianchi et al., 1999; Kalai and Servedio, 2003; Akavia, 2008) and error recovery in databases (Zhang et al., 2020). These recovery and learning algorithms are helpful even if they work with a probability arbitrarily close to 1 instead of with a probability of exactly 1. Moreover, since if the class PH does not collapse, NP-complete problems cannot have polynomial-time non-adaptive worst-case to average-case reductions (Feigenbaum and Fortnow, 1993; Bogdanov and Trevisan, 2006), this is a slight relaxation that we can think of as allowing "almost" worst-case to average-case reduction that may exist non-adaptively for NP-complete problems. As we will see in later sections, this is a natural and unifying model for the significant NP-complete problems and for some problems in P that are structurally similar to NP-complete problems or admit fine-grained reductions from NP-complete problems.

If $\varepsilon = 0$, the existence of a recovery reduction in this model is equivalent to f having an $O\left(t(n)\right)$-time worst-to-average case reduction with error tolerance δ. Constructing a reduction in the random noise model with $\varepsilon > 0$ does not strictly imply a hardness result for computing f on a $(1 - \delta)$-fraction of instances. Indeed, we prove that there is a $(poly(n), 0.5 - 1/poly(n), \exp\left(-\Omega(n^2)\right))$-reduction for the decision problem of detecting cliques of size $n/2$, while it is known due to (Erdős and Rényi, 1963; Pólya, 1937) that $\mathcal{G}(n, 1/2)$[4] does not have a clique of size $n/2$ with probability $1 - o(1)$ - printing 0 without reading the input is a good average-case algorithm. However, this does prove that $(1 - \varepsilon)$-fraction of functions exactly $(1 - \delta)$-close to f are at least as hard to compute as the worst-case complexity of f.

Remark 1. Allowing ε to be larger than 0 is a natural parameter relaxation. For NP-complete problems, since we still want to have polynomial time reductions we may either relax the worst-to-average-case reductions by allowing the error tolerance δ to be small or ε to be non-zero. The result of Bogdanov and Trevisan (2006) prohibits a relaxation of only

[4]The distribution of graphs on n vertices where each edge is selected with probability $1/2$

δ to $1/poly(n)$ unless PH collapses. This suggests that the most natural parameter to relax is ε.

We do note that one potential workaround to the barrier posed by Bogdanov and Trevisan (2006) is to have an adaptive reduction. There exist many works showing the power of adaptive worst-to-average case reductions compared to non-adaptive ones (Feigenbaum et al., 1992; Naik et al., 1993; Babai and Laplante, 1999; Akavia et al., 2006), and also landmark adaptive reductions (Micciancio, 2004; Ajtai, 1996).

1.2 Towards a Generalized NP Function with Symmetry

Before we define the function class we construct recovery reductions for, we must define some preliminary algebraic structures, the first of which is a semilattice.

Definition 3. **Semilattice**
A semilattice $(\mathcal{D}, \odot)$ is a set $\mathcal{D}$ and a binary operation $\odot : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ such that:

1. **(Associativity)** For every $x, y, z \in \mathcal{D}$, $(x \odot y) \odot z = x \odot (y \odot z)$.
2. **(Commutativity)** For every $x, y \in \mathcal{D}$, $x \odot y = y \odot x$.
3. **(Idempotence)** For every $d \in \mathcal{D}$, $d \odot d = d$.

Of course, we require that the semilattice operations be efficiently computable, so we define the following subclass of semilattices.

Definition 4. **Polynomial-Time Computable Semilattice**
A semilattice $(\mathcal{D}, \odot)$ is polynomial time computable if for any elements x and y of $\mathcal{D}$, the product $x \odot y$ is computable in $poly\left(|x| + |y|\right)$ time.

As an exercise, one may verify that $(\{\, 0, 1 \,\}, \vee)$ is a polynomial-time computable semilattice.

We now define a generalization of the class NP that, instead of simply the OR operation, allows any poly-time computable semilattice operation to be performed between the polynomial-time computable predicate values.

Definition 5. **Semilattice NP**
Semilattice NP (SLNP) is the class of functions defined as follows. A function f is in the class SLNP if:

1. There is a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time computable semilattice $(\mathcal{D}, \odot)$ such that for every natural number n, the function $f_n : \Sigma^{p(n)} \to \mathcal{D}$ is the function f restricted to inputs of size parameter n on a constant sized alphabet Σ.
2. There exists a function $h_n : \Sigma^{p(n)} \times \mathcal{C}_n \to \mathcal{D}$ such that

$$f_n(\phi) = \bigodot_{x \in \mathcal{C}_n} h_n(\phi, x),$$

where $|x| = poly(n)$, $\mathcal{C}_n = \Pi^{poly(n)}$ for some constant- sized alphabet Π and $h_n(\phi, x)$ can be computed in time polynomial in n.

As we noticed before that $(\{0,1\}, \vee)$ is a semilattice, it is simple to see that $\mathsf{NP} \subset \mathsf{SLNP}$ (proof in Appendix A).

Lemma 1. *The complexity class* NP *is contained in the function class* SLNP*.*

Now, we add the restriction that our function has some symmetries involved, plus some other convenient properties.

Definition 6. G**-Invariant Semilattice** NP
Suppose $G = (G_1, G_2, \ldots)$ is an infinite sequence of groups. A function f is in G-Invariant Semilattice NP, SLNP^G if:

1. The function f is contained in the function class SLNP.
2. For each group G_n, there exist group actions $\alpha : G_n \times \Sigma^{p(n)} \to \Sigma^{p(n)}$ (p is the size function associated with the function f) and $\beta : G_n \times \mathcal{C}_n \to \mathcal{C}_n$ such that:
   (a) For every $\phi \in \Sigma^{p(n)}$, $x \in \mathcal{C}_n$, and $g \in G_n$, we have that
   $$h_n\left(\alpha_g(\phi), \beta_g(x)\right) = h_n\left(\phi, x\right).$$
   (b) The group action β partitions the set (the certificate space) $\mathcal{C}_n$ into $j(n) = \mathit{poly}(n)$ distinct orbits $\mathcal{C}^{\beta}_{n,1}, \mathcal{C}^{\beta}_{n,2}, \ldots, \mathcal{C}^{\beta}_{n,j(n)}$.
   (c) In $\mathit{poly}(n)$ time, it is possible to compute a list $\left(x_1, x_2, \ldots, x_{j(n)}\right) \in \mathcal{C}_n^{j(n)}$ such that for each $i \in [j(n)]$, $x_i \in \mathcal{C}^{\beta}_{n,i}$.
   (d) The group actions α and β are computable in $\mathit{poly}\left(n\right)$ time.

Lemma 2. ***Isomorphism Invariance Property of*** SLNP^G
Given an indicator function $f \in \mathsf{SLNP}^G$, with group sequence $G_1, G_2, \ldots$ and group actions α and β, we have that for every element $g \in G_n$ and input $\phi \in \Sigma^{p(n)}$,

$$f_n\left(\alpha_g(\phi)\right) = f_n(\phi).$$

Our final definition forces that our sequence of groups is a sequence of symmetric groups.

Definition 7. S**-Invariant Semilattice** NP
A function f is the class S-Invariant Semilattice NP, or SLNP^S if:

1. The function f is in a function class G-Invariant Semilattice NP, SLNP^G.
2. The group sequence $G = (G_1, G_2, \ldots) = \left(S_{m(1)}, S_{m(2)}, \ldots\right)$ where $m : \mathbb{N} \to \mathbb{N}$ is a function whose value and computation time grow at most polynomial in its input.

1.3 Our Results

We state below, the main result of our paper, giving recovery reductions in the random noise model for many of the canonical NP-hard problems.

Theorem 1. *For every $\epsilon > 0$, any function f in the class SLNP^S has a fully deterministic $(poly(n, 1/\epsilon), 0.5 - \epsilon, \exp(-\Omega(poly(n))))$-recovery reduction in the random noise model.*

Theorem 2. *For every $\epsilon > 0$, the following problems have fully deterministic $(poly(n, 1/\epsilon), 0.5 - \epsilon, \exp(-\Omega(poly(n))))$-recovery reductions in the random noise model -* SAT*,* kSAT*,* kCSP*, MAX-*kCSP*,* CLIQUE*,* INDSET*,* VERTEXCOVER*,* kCOLOR*,* HAMCYCLE *and* HAMPATH*.*

We state this result more informally, in an algorithmic sense, for the example of Boolean Satisfiability.

Informal Theorem 1. *Suppose we are given a truth table $\mathsf{T}'_{\mathrm{SAT}}$ for* SAT *instances on n variables such that a randomly chosen $(0.5 - 1/poly(n))$-fraction of the answers are flipped. There is a polynomial time deterministic procedure with access to $\mathsf{T}'_{\mathrm{SAT}}$ that recovers $\mathrm{SAT}(\phi)$ for every formula ϕ, with probability $1 - 2^{-\Omega(poly(n))}$ over the choice of corruptions.*

We emphasize that this probabilistic guarantee is not for each ϕ independently, but that with this probability guarantee, our algorithm works correctly for *every* input ϕ.

Our reduction can be seen as an efficient deterministic recovery algorithm for entries of truth tables in NP-complete problems that fails with very low probability over the choice of corruptions. Our result can be seen as saying, in a database recovery view, "NP-complete truth tables have redundancy built in."

In fact, a famed conjecture of Berman and Hartmanis (1977) would imply that every NP-complete problem has some form of a recovery reduction with the same paramaters as the one we have shown for SLNP^S.

Conjecture 1. ***Berman-Hartmanis Conjecture (Berman and Hartmanis, 1977)*** *Between any two* NP*-complete languages, there exists a bijection that is a polynomial-time reduction computable in either direction.*

Theorem 3. *If the Bermann-Hartmanis Conjecture is true, for every* NP*-complete function (the indicator functions of* NP*-complete langauges), there exists a partition of possible inputs, such that all have fully deterministic $(poly(n, 1/\epsilon), 0.5 - \epsilon, \exp(-\Omega(poly(n))))$-recovery reductions in the random noise model for every $\epsilon > 0$.*

Proof. For any NP-complete indicator function f, the polynomial-time bijection $\mathcal{B}$ to SAT due to the conjecture of Berman and Hartmanis (1977) forces a partition of the input space of f on the basis of the input length of the image of f across the function $\mathcal{B}$. On this partition, we apply the recovery reduction for SAT implied by Theorem 1 and Theorem 2. □

Our recovery algorithm is, in one sense, optimal for non-adaptive algorithms. As stated previously, if we were to improve the probability that the procedure works over the choice of random corruptions to 1, then even a randomized polynomial-time algorithm would imply that PH collapses to the third level due to the work of Feigenbaum and Fortnow (1993) and Bogdanov and Trevisan (2006).

Informal Theorem 2. *The barrier of Bogdanov and Trevisan (2006) for non-adaptive worst-case to average-case reductions requires that the noise be adversarial and not random.*

If we were to raise the fraction of corruptions to 0.5, then this would imply the inclusion of NP in BPP since we would be able to simulate the queries to the truth table $\mathsf{T}'_{\mathrm{SAT}}$ using truly random bits. That is, the truth table received is as good as receiving an almost uniformly random string of that length. We are unable to increase the corruption fraction to $0.5 - 2^{-poly(n)}$ since $2^{-poly(n)}$-fraction advantages in correctness, cannot generally be exploited in polynomial time. Moreover, our recovery algorithm is fully deterministic rather than randomized.

Each of our recovery reductions for all the NP-hard problems listed in Theorem 2 follows from the fact that they are all in the class SLNP^S.

This subtly points towards symmetry and invariance being a structural property of NP-hardness. We discuss this further in Section 1.4 on open problems.

Recovery Reductions for Fine-Grained Problems

We also give recovery reductions in the random noise model for the *Orthogonal Vectors* (OV) problem , deciding whether there is a pair of d-dimensional 0/1 vectors in a list of n such vectors whose dot product (over $\mathbb{R}$) is 0, and *Parity k-Clique*, the problem of computing the lowest order bit on the number of cliques of size k in a simple n-vertex graph. The average-case complexity of these problems has been well-studied. Ball et al. (2017) and Dalirrooyfard et al. (2020) show average-case hardness for the low degree extension and a construction called the "factored version" of the OV problem, respectively. For the *Parity k-Clique* problem, Goldreich (2020), improving upon the work of Boix-Adserà et al. (2019), showed that there is an $O(n^2)$-time worst-to-average-case reduction from computing *Parity k-Clique* in the worst case to computing it correctly on a $\left(1 - 2^{-k^2}\right)$-fraction of instances.

In contrast, for both problems, we give recovery reductions in our model with optimal parameters, while not modifying either Boolean function.

Theorem 4. 1. *For every $\epsilon = 1/polylog(n)$ and dimension d at most $O(n^{1-\gamma})$ for some $\gamma > 0$, we have a $\left(\tilde{O}(nd), 0.5 - \epsilon, 1 - 2^{-nd}\right)$-recovery reduction for the* OV *problem.*

2. *For every constant $k > 0$ and $\epsilon = 1/polylog(n)$, we have a $\left(\tilde{O}(n^2), 0.5 - \epsilon, 1 - 2^{-\binom{n}{2}}\right)$-recovery reduction for Parity k-Clique.*

Here, we use randomness in our reduction so we have a reduction time that is linear (up to polylogarithmic factors) in the input size, to avoid the large polynomial overhead our black-box algorithm gives us.

In some way, both these problems have structure similar to NP-complete problems. OV admits a $2^{0.5n}$-time reduction from kSAT with n variables Williams (2005) and *Parity k-Clique* is ETH-hard (Goldreich and Rothblum, 2018; Chen et al., 2006). This suggests a relationship between reduction from NP even if the reduction is subexponential, and the existence of a recovery reduction. We note that our techniques do not imply random reductions for arbitrary polynomial-time computable functions. We further discuss this in Remark 3.

Remark 2. We prove our main theorem (Theorem 2) for the NP-hard problems listed in the theorem statement. However, we emphasize that this is not an exhaustive list of NP-hard problems whose recovery reductions follow from Theorem 1. We believe that many others follow, even with short proof sketches, but we only list the popular NP-hard problems for brevity.

1.4 Open Problems

1.4.1 NP-Completeness and Recovery Reductions

Since we showed recovery reductions for many of the natural NP-complete problems, it is natural to wonder if the existence of a recovery reduction is an inherent property of NP-completeness. We conjecture the following about NP-complete problems.

Conjecture 2. ***Recovery Reductions for Every*** **NP-*Complete Problem***
For every $\epsilon > 0$, every NP*-complete problem has a*
$(poly(n, 1/\epsilon), 0.5 - \epsilon, \exp(-poly(n)))$*-recovery reduction in the random noise model.*

We also strengthen this and make this conjecture for deterministic recovery reductions.

Conjecture 3. ***Deterministic Recovery Reductions for Every*** **NP-*Complete Problem***
For every $\epsilon > 0$, every NP*-complete problem has a deterministic*
$(poly(n, 1/\epsilon), 0.5 - \epsilon, \exp(-poly(n)))$*-recovery reduction in the random noise model.*

Open Problem 1. *Is* NP*, considering the indicator function of each language, contained in the class* SLNP^S*?*

Due to Theorem 1, a positive answer to 1 implies Conjecture 3, which in turn implies Conjecture 2.

The authors believe Conjecture 1 to be true since heuristically, NP-complete problems must be expressive and since they admit reductions from problems in NP with rich symmetries, it seems as though NP-complete problems must retain some essence of the original symmetries.

As discussed before, Berman and Hartmanis (1977) originally conjectured that between any-two NP-complete problems, there is a bijection between the two languages that is polynomial-time computable on both sides. They call this a p-isomorphism. This conjecture would already imply deterministic recovery reductions for NP-complete languages not on truth tables of fixed input length for that language, but a truth table on the input set that is the image of the p-isomorphism from (say) SAT instances on n variables. Really, this is a

relabelled SAT truth table, and we could compute the labels via the two-way p-isomorphism and use the recovery reduction for SAT as the main procedure.

Hence, we believe that our conjectures are important in the study of the structure of NP-completeness.

Open Problem 2. ***Recovery Reductions for Expressive Problems in*** P
Suppose f is a problem in P *of (randomized) time complexity $T(N)$ (on instances of f of size N) with a $2^{o(n)}$-time reduction from n variable instances of* 3SAT*. Does every such function f have a $(t(N), 0.5-\epsilon, 1-\exp(-\Omega(poly(N))))$-recovery reduction in the random noise model such that $t(N) = o(T(N))$ for every constant $\epsilon > 0$.*

While OV and *Parity k-Clique* do not necessarily have these properties, we raise the above question. Implicitly assuming the Exponential Time Hypothesis (ETH)[5] (Impagliazzo et al., 2001; Impagliazzo and Paturi, 2001), if a problem f admits a subexponential time reduction from an NP-complete problem L, must it contain the relevant symmetry conditioned on the fact that the language L contains symmetry? If so, it is highly possible that f contains enough symmetry to have a non-trivial[6] recovery reduction.

We ask what consequences the above proposed conjectures might have.

Open Problem 3. ***Consequences of Recovery Reductions for Every*** NP***-Complete Problem***
What conditional results in complexity theory can be shown assuming Conjecture 2, Conjecture 3, or Conjecture 1?

1.4.2 Studying Correction in Various Models of Errors

We can see errors as follows - a source of errors produces a string and XORs the string to the truth table. In our case, the source picks a random subset of inputs, with fixed Hamming weight and adversarially chooses the entries there and we try to decode with high probability over the behavior of the source. In the traditional setting in coding theory, the source introduces an adversarial string of fixed Hamming weight and the task is to decode any input with high probability.

Informal Question 1. Can we construct efficient recovery reductions for functions when the error is produced by an arbitrary AC^0 or an NC^0 source?

We believe the above question to be an intermediate between the two settings - where the former is easy to construct reductions for and the latter is difficult and in some cases prohibitive to construct reductions in the latter model.

[5] The Exponential Time Hypothesis, ETH states that 3SAT on n variables requires 2^{cn} deterministic time to decide for some constant $c > 0$.

[6] A trivial recovery reduction would simply use the algorithm that computes f normally in the worst-case.

2 Intuition and Technical Overview

Suppose we have our function f in the class SLNP^S. The concrete function to keep in mind here is CLIQUE, where $f_n : [n] \times \{0,1\}^{\binom{n}{2}} \to \{0,1\}$[7], written as $f_n(k, H)$, is 1 if and only if the simple n vertex graph H contains a clique of size k.

This function is in the class SLNP^S since we can write

$$f_n(k, H) = \bigvee_{T \subset [n]} \mathbb{1}_{\text{The vertex set encoded by } T \text{ forms a clique in } H \text{ and } |T|=k},$$

where $\mathbb{1}_A$ is the indicator function of A[8]. A permutation π of S_n acts on the graph H by permuting it according to the permutation π and acts on the set T by replacing every element $i \in T$ with $\pi(i)$.

Example - Sketching a Randomized Recovery Reduction for CLIQUE

Our recovery reductions rely on the following intuitive observations.

1. Symmetry - having a large automorphism group - makes computation easy.
2. Suppose our input ϕ has a lot of distinct instances isomorphic to it - it has a large orbit. The law of large numbers guarantees that if the truth table $\mathtt{T}_f$ is corrupted at random on 49% of instances, then with high probability, the corrupted truth table has at least 50.5% of isomorphic instances have correct answers in the corrupted truth table $\mathtt{T}'_f$.

The Orbit-Stabilizer Theorem (Lemma 4), saying that $|Aut_G(\phi)|\,|Orb_G(\phi)| = |G|$ guarantees that at least one of the two conditions is true!

When the Graph is Highly Symmetric

The first observation means that if our instance ϕ is highly symmetric - it's automorphism group is large - then, we can compress the proof verification process. For our function $f(k, H)$, suppose H has a very large automorphism group. Intuitively, this means that the presence of one clique of size k implies the existence of many others - we do not need to check whether there is a clique of size k for each of these vertex sets S - it suffices to check only one of them in this family. If there is a clique of size k in this position encoded by S, we can return the answer 1 for $f(k, H)$. Otherwise, we conclude immediately that many other possible vertex families of size k - at least those in the orbit of S when acted on by the automorphism group $Aut(H)$ - do not contain cliques of size k in the graph H.

Using the Orbit-Stabilizer Theorem, our job, if we know that the index of $|Aut(H)|$,

$$n!/|Aut(H)|,$$

[7] This can be modified to be of the form $f_n : \{0,1\}^{p(n)} \to \mathcal{D}$ if we treat $[n]$ as $\{0,1\}^{\lceil \log n \rceil}$ and let f_n be uniformly zero if the first $\lceil \log n \rceil$ bits encode a number larger than n.

[8] That is, $\mathbb{1}_A$ is 1 if the assertion A is true and 0 if A is false.

is small - say $O(n^3)$ - is to list a set of the $O(n^3)$ right coset representatives (a right transversal) U_H^R of the automorphism group $Aut(H)$ and evaluate whether the vertex set $\beta_\pi(\{1, 2, \ldots, k\})$[9] contains a clique of size k in the graph H for each π in the right transversal U_H^R. We list each coset representative in the following way - instantiate a list containing only the graph H. Keep sampling random permutations $\pi \in S_n$ and add $\alpha_\pi(H)$ to the list if it is not already contained in the list. From the Orbit-Stabilizer Theorem, we know that there should be $n!/|Aut(H)| = O(n^3)$-many distinct graphs and we stop this procedure once we have that many. This is a classic case of the coupon collector problem, and we can cover the space in $O(n^3 \log n)$ randomized time (Flajolet et al., 1992; Mitzenmacher and Upfal, 2017). Then, we complete our $O(n^3)$ evaluations to compute

$$\bigvee_{\pi \in U_H^R} \mathbb{1}_{\beta_\pi(\{1,2,\ldots,k\}) \text{ is a clique of size } k \text{ in the graph } H}$$

and obtain a polynomial-time randomized algorithm to compute CLIQUE on highly symmetric graph inputs - with $O(n^3)$ index.

When the Graph is Less Symmetric

The second observation, on the other hand is that the operator $\mathcal{N}_{0.49}$ acts on the truth table $\mathtt{T}_f$ to return the corrupted truth table $\mathtt{T}'_f$ with 49% of the entries randomly corrupted. Due to the law of large numbers, and formalized by concentration bounds, all large orbits - of size $\Omega(n^3)$ - of $\mathcal{P}([n])$[10] will have at least 50.5% of their answers correct with high probability (over the randomness of the choice of corruptions). We can simply sample random permutations from S_n and query the corrupted truth table on $\mathtt{T}'_f$ on polynomially many entries isomorphic to H. With high probability, the majority value is correct! With high probability over the randomness of corruptions, we now have a randomized polynomial-time query algorithm for the asymmetric case - when the index is $\Omega(n^3)$.

Distinguishing between the Two Cases

Now, all that remains to be done is to distinguish between the two cases. We have already hinted at a method to compute the index of a subgroup. We noted before that we can sample random permutations $\pi \in S_n$ and list every new graph $\alpha_\pi(H)$ that we have not seen before. Suppose the index is indeed of size $O(n^3)$. Then not only is this list of size $O(n^3)$, but after $O(n^3 \log n)$ samples, this list stops growing. To be safe, we may even use $O(n^4)$ samples and ensure this list actually stops growing. After $O(n^4)$-time, if our list stops growing and is below our $O(n^3)$ size threshold, we use our procedure for highly symmetric graphs. Otherwise, we use the procedure where we query the corrupted truth table $\mathtt{T}'_f$.

[9] In this case, there is only one orbit of sets of size k under action from S_n. Typically, we would need to evaluate over every orbit. Of course, here, for CLIQUE, we know a priori we only need to evaluate for one size - this will generally not be the case.

[10] This is the power set of $[n]$ in this case.

Derandomizing the Recovery Reduction

Now that we have an intuitive feel for how and why our algorithm works, our job remains to give deterministic procedures for the parts of the recovery algorithms that we used randomness in. Namely, we used randomness in our algorithms for the following subroutines:

1. Distinguishing between large and small index automorphism groups and listing coset representatives when the index of the group is small.
2. Querying the corrupted truth table $\mathtt{T}'_f$.

For the first case, luckily for us, much work has been done in the area of computational group theory, for deterministic algorithms for permutation groups. We use the following result.

Lemma 3. ***(Furst et al., 1980; Sims, 1970)***
Suppose we have a positive integer m, and a subgroup G of S_m such that we have a deterministic membership test to determine if any element $\pi \in S_m$ is a member of G in time $T_G = \Omega(m \log m)$. Then,

1. *In $poly(m)T_G$ deterministic time, it is possible to compute the size of the subgroup G, $|G|$.*
2. *For any integer k, in $poly(m)poly(k)T_G$ deterministic time, it is possible to print a list of k distinct (left or right) coset representatives of G.*[11]

Using these algorithms and the fact that simply checking if $\alpha_\pi(H) = H$ is an efficient membership test, we can both determine the size of the automorphism group $Aut(H)$ deterministically in polynomial-time, and compute a set of right coset representatives efficiently if we know the index is polynomial in n.

Now, derandomizing the second part is more challenging. A naive approach is to query the corrupted truth table $\mathtt{T}'_f$ on every possible graph isomorphic to H, but the not-so-subtle issue for almost all graphs is that they have automorphism groups of size 1 and we would be forced to make $n!$ queries (Pólya, 1937; Erdős and Rényi, 1963).

Suppose we use the deterministic right coset listing algorithm in Lemma 3 and construct a list L of polynomially many right coset representatives of the automorphism group $Aut(H)$ - can we query the truth table $\mathtt{T}'_f$ on $\alpha_\pi(H)$ for every $\pi \in L$, take the majority value and hope we are correct?

As it turns out, we show using Chernoff bounds that indeed, with probability $1 - \exp(-poly(n))$ over the randomness of the operator $\mathcal{N}_{0.49}$, for every graph H with $\Omega(n^3)$ automorphism index size, the "query path" described above has at least 50.5% of the answers correct within the corrupted truth table $\mathtt{T}'_f$ under the assumption that L is sufficiently large (but still polynomial). Hence, we can indeed simply do this and take the majority answer!

With these two ingredients, we have fully derandomized the recovery reduction for CLIQUE. Our generalized recovery algorithm for SLNP^S proceeds almost identically.

[11] In standard texts, the algorithm runtime depends on the index of G as $poly([S_m : G])$. However, to print only k coset representatives, it suffices to terminate this procedure after we have listed k coset representatives, which is possible with a polynomial time dependence on k.

3 Preliminaries

3.1 Notation

Throughout, we use $\mathtt{T}_{f_n}$ to refer to the truth table of the function section f_n (of input length parameter n). We use $\mathtt{T}'_{f_n}$ to refer to a corrupted truth table of the function section f_n. We use Σ (outside of summation notation) to denote an alphabet of constant size. Typically, when we say $p(n)$, we refer to a positive function p that is bounded from above by n^c for some constant c. The notation $poly(n)$ is a substitute for a positive function bounded from above by a polynomial in n (n^c for some constant c) and $polylog(n)$ is a substitute for a positive function bounded from above by $(\log n)^c$ for some constant c. The notation ln represents the natural logarithm with base e and log represents the logarithm with base 2. For any positive integer m, $[m]$ denotes the set $\{1, 2, \ldots m\}$. The notation $\exp(y)$ denotes e^y.

Throughout, α_g typically represents the group action of the element g applied on the input to the function f and β_g represents the action of the element g on a member x of the certificate set $\mathcal{C}$. When G' is a subgroup of a group G, the index $[G : G'] = |G|/|G'|$. $Aut_G(y)$ represents the automorphism group of the object y when acted on by the group G and $Orb_G(y)$ represent the orbit of y when acted on by the group G. Throughout, we use U_O^L (U_O^R) to represent the left (right) transversal of the automorphism group of the object O. A left transversal of a subgroup $H \subset G$ is a list consisting of exactly one element of each left coset of H. A right transversal is the analogous object for right cosets.

3.2 Orbit Stabilizer Theorem

The key technical idea from group theory underpinning our techniques is the concept of group actions (Smith, 2015).

Definition 8. **Group Actions**
Given a group G and a set X, a group action $\alpha : G \times X \to X$ is a function satisfying the following axioms:

- If e is the identity of G and x is any element of X, $\alpha(e, x) = x$.
- Given any $g, g' \in G$ and $x \in X$, $\alpha(g', \alpha(g, x)) = \alpha(g'g, x)$.

We will write $\alpha(g, x)$ as $\alpha_g(x)$ going forward.

Ideally, we use group actions to describe the symmetries of a set. A simple example of a group action is left multiplication, $g \cdot x$, where x is a member of the group itself. A more useful example, and one we will indeed use in this paper, is the action of the symmetric group, S_n, on the set of simple n-vertex graphs. The action of a permutation $\pi \in S_n$ on a graph H is to return a graph H' isomorphic to H[12] such that the vertices and edges of H are permuted according to π.

[12]We call these graphs different if their adjacency matrices are not equal.

Definition 9. **Automorphism Group** or **Stabilizer**[13]
Given a group action $\alpha : G \times X \to X$, for any $x \in X$, the automorphism group $Aut_G(x) \subset G$ (or stabilizer G_x) is defined as

$$G_x = Aut_G(x) = \{\, g \in G \mid \alpha_g(x) = x \,\}.$$

$Aut_G(x)$ is to be viewed as the subgroup of elements of G fixing x, or equivalently, the subgroup of transformations or actions under which x is invariant.

Definition 10. **Orbits**
Given a group action $\alpha : G \times X \to X$, for any $x \in X$, the orbit of x, $Orb_G(x) \subset X$ is defined as

$$Orb_G(x) = \{\, \alpha_g(x) \mid g \in G \,\}.$$

The orbit of x is to be seen as the set of elements of X that are isomorphic to it. For example, intuitively, the orbit of a graph H is the set of unique labelled graphs isomorphic to H. The automorphism group of H is the set of permutations or relabellings that conserve the exact labelled edge relations.

We now state the standard version of the Orbit-Stabilizer Theorem.

Lemma 4. ***Orbit-Stabilizer Theorem (Quantitative Version)***
Given a group action $\alpha : G \times X \to X$, for any element $x \in X$, the following relation holds true

$$|Orb_G(x)||Aut_G(x)| = |G|.$$

More precisely and for better intuition, we state a more qualitative version of this theorem.

Lemma 5. ***Orbit-Stabilizer Theorem (Qualitative Version)***
Given a group action $\alpha : G \times X \to X$, for any element $x \in X$, and any left transversal U_x^L of the automorphism group $Aut_G(x)$, we have that

$$Orb_G(x) = \left\{\, \alpha_u(x) \mid u \in U_x^L \,\right\}.$$

Hence, we can view members of the orbit of x, $Orb_G(x)$, as "isomorphic" to the left cosets of the automorphism group $Aut_G(x)$, in one sense. The qualitative version of the theorem implies the quantitative version since this is now just a special case of Lagrange's theorem (Smith, 2015).

4 The Unified Meta Theorem

In this section, we prove our main theorem, showing a recovery reduction for SLNP^S. We organize this section based on the various ingredients in the algorithm.

[13] In group theoretic literature, this is referred to as the stabilizer, while in combinatorics, especially in graph theory, it is referred to as the automorphism group. Throughout this paper, we will use the term "automorphism group".

4.1 Determining Automorphism Group Size

It follows from the work of Sims (1970) and Furst et al. (1980) that using the efficiently computable group action α as our efficient membership test, we have a polynomial-time deterministic algorithm to compute the size of the automorphism group $Aut_{S_m}(\phi)$ of any instance ϕ.

Lemma 6. *Suppose we have a function f in the class SLNP^S. Given any instance $\phi \in \Sigma^{p(n)}$, there is a poly(n)-time algorithm to determine the order $|Aut_{S_m}(\phi)|$ of the automorphism group of ϕ.*

Proof. From Definition 7 it follows that the variable m is polynomial in n and from Defintion 6, it follows that the group action $\alpha : S_m \times \Sigma^{p(n)} \to \Sigma^{p(n)}$ is computable in $poly(n)$-time. The membership test to check if $\pi \in S_m$ is contained in $Aut_{S_m}(\phi)$, given any instance $\phi \in \Sigma^{p(n)}$ is to verify that $\alpha_\pi(\phi) = \phi$, which can be done in $poly(n)$-time. Hence, Lemma 3 implies that the group order $|Aut_{S_m}(\phi)|$ can be deterministically computed in $poly(n)$-time.

□

4.2 Probabilistic Guarantees for Asymmetric Instances

Now, we prove that for instances with relatively small automorphism groups, we have a deterministic polynomial time procedure that uses queries to the corrupted truth table $\mathtt{T}'_f$.

First, we prove a lemma that allows us to make the queries deterministically and obtain good probabilistic guarantees over the randomness of the noise operator $\mathcal{N}_{0.5-\epsilon}$. Suppose our query strategy for each input ϕ is deterministic and non-adaptive. Our lemma says that with high probability over the choice of corruptions, for every asymmetric input ϕ, the queries we make retrieve correct answers at least $(0.5 + \epsilon/2)$-fraction of the time.

Lemma 7. *Suppose we have a function $f_n : \Sigma^{p(n)} \to \mathcal{D}$ and subsets*

$$T_1, T_2, \ldots, T_{|\Sigma|^{p(n)}} \subset \Sigma^{p(n)},$$

each of size at least $16p(n)\ln(|\Sigma|)/\epsilon^2$. With probability at least $1 - 1/|\Sigma|^{p(n)}$ over the choice of random corruptions of the operator $\mathcal{N}_{1/2-\epsilon}$, the corrupted truth table $\mathtt{T}'_f = \mathcal{N}_{1/2-\epsilon}\mathtt{T}_f$ has at least $(1/2 + \varepsilon/2)$-fraction of instances left uncorrupted within each subset T_i.

Proof. Consider the random variables $(X_\phi)_{\phi \in T}$. The random variable X_ϕ attains the value of 1 if the entry of the corrupted truth table $\mathtt{T}'_f = \mathcal{N}_{1/2-\epsilon}\mathtt{T}_f$ corresponding to the input ϕ is left uncorrupted and 0 otherwise. We define the random variable $X_T = \sum_{\phi \in T} X_\phi$ counting the number of uncorrupted entries within T.

We use the Chernoff bound (Mitzenmacher and Upfal, 2017; Dubhashi and Ranjan, 1996) with negative correlation to obtain that

$$\mathcal{P}\left[X_T/|T| \leq 1/2 + \epsilon/2\right] \leq e^{-\epsilon^2|T|/8}. \tag{1}$$

This applies when each random variable $X_i \in \{0, 1\}$ in the sum is identically distributed on $[0, 1]$, but

$$\mathcal{P}\left[X_j = 1 | X_i = 1\right] \leq \mathcal{P}\left[X_j = 1\right]$$

for all $i \neq j$.

We have that

$$\mathcal{P}\left[X_\phi = 1 | X_{\phi'} = 1\right] = \frac{\delta|\Sigma|^{p(n)} - 1}{|\Sigma|^{p(n)} - 1} < \delta = \mathcal{P}\left[X_\phi = 1\right]$$

for any distinct inputs ϕ and ϕ' from $\Sigma^{p(n)}$. This holds since the sum $\sum_{\phi \in \Sigma^{p(n)}} X_\phi = \delta|\Sigma|^{p(n)}$ is a conserved quantity and if the truth table $\mathtt{T}'_f$ is uncorrupted at the entry ϕ', only $\delta|\Sigma|^{p(n)} - 1$ of the other $|\Sigma|^{p(n)} - 1$ entries can be left uncorrupted and their random variables are still identically distributed under this condition.

Hence, given any subset $T \subset |\Sigma|^{p(n)}$ with size $|T| \geq 16p(n)\ln\left(|\Sigma|\right)/\epsilon^2$, we have, using Equation 1, that over the choice of random corruptions of $\mathcal{N}_{1/2-\epsilon}$,

$$\mathcal{P}\left[\text{The fraction of uncorrupted entries in } T \text{ is less than } 1/2 + \epsilon/2\right] \leq |\Sigma|^{-2p(n)}.$$

Assume we have subsets $T_1, T_2, \ldots, T_{|\Sigma|^{p(n)}} \subset \Sigma^{p(n)}$, each of size at least $16p(n)\ln|\Sigma|/\epsilon^2$. The probability that over the choice of random corruptions of the operator $\mathcal{N}_{1/2-\epsilon}$ that at least one subset T_i with $i \in \left[|\Sigma|^{p(n)}\right]$ has less than $(1/2 + \epsilon/2)$-fraction correct entries is at most

$$|\Sigma|^{p(n)} \cdot |\Sigma|^{-2p(n)} = 1/|\Sigma|^{p(n)},$$

due to the union bound.

□

Now, using this lemma, we provide our deterministic querying algorithm, using the partial coset transversal algorithm in Lemma 3 to help us list distinct isomorphic instances.

Lemma 8. *Suppose we have a function f in the class SLNP^S and $\epsilon > 0$. Given query access to the corrupted truth table $\mathtt{T}'_f = \mathcal{N}_{1/2-\epsilon}\mathtt{T}_f$, there is a deterministic poly$(n, 1/\epsilon)$-time procedure that, with probability at least $1 - 1/|\Sigma|^{p(n)}$ over the choice of corruptions of $\mathcal{N}_{1/2-\epsilon}$, computes $f_n(\phi)$ correctly for every input $\phi \in \Sigma^{p(n)}$ such that $|Aut_{S_m}(\phi)| \leq m!/\left(16p(n)|\Sigma|/\epsilon^2\right)$.*

Proof. Suppose that for each input $\phi \in \Sigma^{p(n)}$, $\mathtt{COSET}_\phi(k)$ is the list of k (possibly incomplete) coset representatives of $Aut_{S_m}(\phi)$, $(g_1, g_2, \ldots, g_k)$ returned by the deterministic procedure for computing a possibly incomplete list of coset representatives implied by Lemma 3.

For each input $\phi \in \Sigma^{p(n)}$ with automorphism group size $|Aut_{S_m}(\phi)| \leq m!/\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right)$[14], suppose

$$\mathtt{COSET}_\phi\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right) = \left(g_1, g_2, \ldots, g_{16p(n)\ln(|\Sigma|)/\epsilon^2}\right)$$

is the list returned by the deterministic procedure in Lemma 3. Then, we define for each input $\phi \in \Sigma^{p(n)}$, the set T_ϕ as

$$T_\phi = \left\{ \alpha_{g_1}(\phi), \alpha_{g_2}(\phi), \ldots, \alpha_{g_{16p(n)\ln(|\Sigma|)/\epsilon^2}}(\phi) \right\}.$$

[14]Otherwise, we set $T_\phi = \Sigma^{p(n)}$ - this just allows Lemma 7 to apply in a blackbox fashion. We will not actually use such a set T_ϕ algorithmically.

Since these are coset representatives of the automorphism group $Aut_{S_m}(\phi)$ and the orbit stabilizer theorem (Lemma 4) guarantees that we have at least $(16p(n)\ln(|\Sigma|)/\epsilon^2)$ distinct orbit members (or coset representatives), it is easy to see that each of the images of group action of $COSET_\phi\,(16p(n)\ln(|\Sigma|)/\epsilon^2)$ is unique. Hence, each set T_ϕ is of size exactly $(16p(n)\ln(|\Sigma|)/\epsilon^2)$. Due to Lemma 7, with probability at least $1-1/|\Sigma|^{p(n)}$ over the choice of corruptions of the operator $\mathcal{N}_{1/2-\epsilon}$, the corrupted truth table $\mathtt{T}'_f = \mathcal{N}_{1/2-\epsilon}\mathtt{T}_f$ has at least a $(1/2+\epsilon/2)$-fraction of entries left uncorrupted over the set T_ϕ for every input $\phi \in \Sigma^{p(n)}$.

The algorithm is defined as follows. Suppose we are given $\phi \in \Sigma^{p(n)}$ such that $|Aut_{S_m}(\phi)| \leq m!/\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right)$. We use the deterministic procedure in Lemma 3 to compute $16p(n)\ln(|\Sigma|)/\epsilon^2$ coset representatives of $Aut_{S_m}(\phi)$, where the membership test is to apply α_π to ϕ and check whether the image is ϕ. Upon computing the list

$$\mathtt{COSET}_\phi\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right) = \left(g_1, g_2, \ldots, g_{16p(n)\ln(|\Sigma|)/\epsilon^2}\right),$$

we compute the set

$$T_\phi = \left\{\, \alpha_{g_1}(\phi), \alpha_{g_2}(\phi), \ldots, \alpha_{g_{16p(n)\ln(|\Sigma|)/\epsilon^2}}(\phi) \,\right\}.$$

Then, we query the corrupted truth table $\mathtt{T}'_f$ in the locations listed in T_ϕ. We take the majority value in $\mathcal{D}$ of all the retrieved values, provided one exists. Under the $\left(1-1/|\Sigma|^{p(n)}\right)$-probability guarantee over the choice of corruptions of the operator $\mathcal{N}_{1/2-\epsilon}$, at least $(1/2+\epsilon/2)$-fraction of answers are uncorrupted. Hence, under these guarantees, the majority is well-defined and equal to $f(\phi)$ due to Lemma 2.

The entire procedure requires $poly(m)poly(1/\epsilon)poly(n)$ deterministic time and $O(p(n)/\epsilon^2)$ queries to the corrupted truth table $\mathtt{T}'_f$. Due to definition of the class SLNP^S (Definition 7), $m = poly(n)$ and $p(n) = poly(n)$. Hence, the total deterministic runtime of this procedure is polynomial in n and $1/\epsilon$.

□

4.3 Quick Computability for Symmetric Instances

Now, we aim to handle the case where the automorphism group of the instance is very large - almost as large as S_m itself on the logarithmic scale. Here, we make the following observation about the structure of the class SLNP^S. This follows from the definition of the class SLNP^G (Definition 6).

$$f_n(\phi) = \bigodot_{x\in\mathcal{C}_n} h_n(\phi, x) = \bigodot_{i\in[j(n)]} \left(\bigodot_{x\in\mathcal{C}^\beta_{n,i}} h_n(\phi, x) \right). \tag{2}$$

Note that, by definition, $j(n)$ is a function that is polynomial in n. Now, for convenience of writing, let

$$f_{n,i}(\phi) = \bigodot_{x\in\mathcal{C}^\beta_{n,i}} h_n(\phi, x), \tag{3}$$

where $\mathcal{C}^\beta_{n,i}$ is the i^{th} orbit of $\mathcal{C}$ as defined in Definition 6.

We want to be able to compute each $f_{n,i}(\phi)$ efficiently by some process, leveraging the fact that the instance ϕ is highly symmetric and then compute

$$f_n(\phi) = \bigodot_{i \in [j(n)]} f_{n,i}(\phi) \tag{4}$$

in polynomial time since $j(n) = O\left(poly(n)\right)$.

Indeed, we do compress the brute-force enumeration process by using the symmetries of the instance ϕ. First, making the following observation.

Observation 1. *Given that $h_n\left(\alpha_g(\phi), \beta_g(x)\right) = h_n(\phi, x)$ for every $g \in G_n$, $\phi \in \Sigma^{p(n)}$, $x \in \mathcal{C}_n$, for every element g in the automorphism group $Aut_{S_m}(\phi)$, we have that*

$$h_n\left(\phi, \beta_g(x)\right) = h_n(\phi, x).$$

This follows immediately from the definition of the automorphism group (Definition 9).

Now, we prove our main lemma making our algorithm for high-symmetry instances possible. This lemma, in essence, says that we can reduce the time used to verify all possible proofs by not checking proofs we know the validity of due to symmetry.

Lemma 9. ***Compressing Brute-Force via Symmetry***
Suppose we have a function f contained in the class SLNP^S. Then, for any $n \in \mathbb{N}$, $i \in [j(n)]$, $g \in S_{m(n)}$, and given any $y_i \in \mathcal{C}^{\beta}_{n,i}$, we have that

$$f_{n,i}(\phi) = \bigodot_{g \in S_m} h_n\left(\phi, \beta_g(y_i)\right)$$

and

$$f_{n,i}(\phi) = \bigodot_{u \in U^R_\phi} h_n\left(\phi, \beta_u(y_i)\right),$$

where U^R_ϕ is the coset right transversal of the automorphism group $Aut_{S_m}(\phi)$.

Proof. 1. Now, suppose $Aut_{S_m}(x)$ is the automorphism group of any element $x \in \mathcal{C}_n$ and U^L_x is the left transversal of $Aut_{S_m}(x)$. Due to the Orbit-Stabilizer theorem (Lemma 5), we have that for each $x' \in Orb_{S_m}(x)$, there is a coset representative $u \in U^L_x$ such that $x' = \beta_u(x)$. And hence, may write $h_n\left(\phi, x'\right) = h_n\left(\phi, \beta_u(x)\right)$. Since $d \odot d = d$ for every $d \in \mathcal{D}$, we may write $h_n\left(\phi, x'\right) = h_n\left(\phi, \beta_u(x)\right) \odot h_n\left(\phi, \beta_u(x)\right) \odot \cdots \odot h_n\left(\phi, \beta_u(x)\right)$ any non-zero number of times. Since $\beta_{ug}(x) = \beta_u(x)$ for every $g \in Aut_{S_m}(x)$, we have that

$$h_n\left(\phi, x'\right) = \bigodot_{g \in Aut_{S_m}(x)} h_n\left(\phi, \beta_{ug}(x)\right).$$

Subsequently, due to the Orbit-Stabilizer theorem (Lemma 5), enumerating over all members x' of the orbit of x, we have that

$$\begin{aligned}\bigodot_{x' \in Orb_{S_m}(x)} h_n\left(\phi, x'\right) = \bigodot_{u \in U^L_x} h_n\left(\phi, \beta_u(x)\right) &= \bigodot_{u \in U^L_x} \left(\bigodot_{g \in Aut_{S_m}(x)} h_n\left(\phi, \beta_{ug}(x)\right) \right) \\ &= \bigodot_{g \in S_m} h_n\left(\phi, \beta_g(x)\right).\end{aligned}$$

Applying this to the orbit $\mathcal{C}^{\beta}_{n,i}$ and its member y_i, we get that

$$f_{n,i}(\phi) = \bigodot_{g \in S_m} h_n\left(\phi, \beta_g\left(y_i\right)\right).$$

2. Using the previous part that $f_{n,i}(\phi) = \bigodot_{g \in S_m} h_n\left(\phi, \beta_g\left(y_i\right)\right)$, we rewrite this as

$$f_{n,i}(\phi) = \bigodot_{g \in S_m} h\left(\phi, \beta_g\left(y_i\right)\right) = \bigodot_{u \in U^R_\phi} \left(\bigodot_{g \in Aut_{S_m}(\phi)} h_n\left(\phi, \beta_{gu}(x)\right) \right)$$

using the Orbit-Stabilizer theorem (Lemma 5). Now, keep in mind that $\beta_{gu}(x) = \beta_g\left(\beta_u(x)\right)$. Using Observation 1, we get that for every $g \in Aut_{S_m}(\phi)$,

$$h_n\left(\phi, \beta_{gu}(x)\right) = h_n\left(\phi, \beta_g\left(\beta_u(x)\right)\right) = h_n\left(\phi, \beta_u(x)\right).$$

Using the fact that $d \odot d = d$ for every $d \in \mathcal{D}$, we can now show that

$$\bigodot_{g \in Aut_{S_m}(\phi)} h_n\left(\phi, \beta_{gu}(x)\right) = \bigodot_{g \in Aut_{S_m}(\phi)} h_n\left(\phi, \beta_u(x)\right) = h_n\left(\phi, \beta_u(x)\right).$$

Subsequently, we have that

$$f_{n,i}(\phi) = \bigodot_{u \in U^R_\phi} \left(\bigodot_{g \in Aut_{S_m}(\phi)} h_n\left(\phi, \beta_{gu}(x)\right) \right) = \bigodot_{u \in U^R_\phi} h_n\left(\phi, \beta_u(y_i)\right).$$

□

And now, we prove the existence of our polynomial-time algorithm to compute each subfunction $f_{n,i}$.

Lemma 10. *Suppose we have a function f in the class SLNP^S. Then given any input $\phi \in \Sigma^{p(n)}$ such that the automorphism group size is $|Aut_{S_m}(\phi)| \geq m!/\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right)$ and $i \in [j(n)]$, $f_{n,i}(\phi)$ is deterministically computable in $poly(n, 1/\epsilon)$-time.*

Proof. From the definition of the class SLNP^G (Definition 6), for each $i \in [j(n)]$, we can compute a member x_i of the orbit $\mathcal{C}^{\beta}_{n,i}$ in $poly(n)$ time. From Definition 7, we have that the time complexity of computing x_i is bounded by a polynomial in n. Using part 2 of Lemma 9, we have that

$$f_{n,i}(\phi) = \bigodot_{u \in U^R_\phi} h_n\left(\phi, \beta_u(x_i)\right).$$

We compute the right transversal U^R_ϕ using the deterministic procedure specified in Lemma 3 for coset representatives. Since $|Aut_{S_m}(\phi)| \geq m!/\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right)$, due to the Orbit-Stabilizer theorem (Lemma 5), we have that $\left|U^R_\phi\right| \leq 16p(n)\ln(|\Sigma|)/\epsilon^2$. Hence, due to Lemma 3, U^R_ϕ is computable in $poly(n, 1/\epsilon)$-time and contains $poly(n, 1/\epsilon)$ members.

Now, we enumerate over U^R_ϕ and compute $h_n\left(\phi, \beta_u\left(x_i\right)\right)$ for each $u \in U^R_\phi$. We can compute each of these answers in $poly(n)$-time due to the definition of the class SLNP^S (Definition 6 and Definition 7). Using all these answers, we can compute $f_{n,i}(\phi) = \bigodot_{u \in U^R_\phi} h_n\left(\phi, \beta_u\left(x_i\right)\right)$. Since the right transversal U^R_ϕ contains polynomially many elements, we only evaluate the function h_n and perform the operation $\odot$ polynomially many times.

□

Now, we combine these $f_{n,i}$'s to obtain our value $f_n(\phi)$.

Lemma 11. *Suppose we have a function f contained in the class SLNP^S and are given an input $\phi \in \Sigma^{p(n)}$ with the promise that the automorphism group size is $|Aut_{S_m}(\phi)| \geq m!/\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right)$. Then, $f_n(\phi)$ is deterministically computable in poly $(n, 1/\epsilon)$-time.*

Proof. Using the $poly(n, 1/\epsilon)$-algorithm in Lemma 10, we enumerate $f_{n,i}(\phi)$ for every $i \in [j(n)]$. Following this, we compute

$$f_n(\phi) = \bigodot_{i\in[j(n)]} f_{n,i}(\phi).$$

This can be done in $poly(n, 1/\epsilon)$-time since $j(n) = poly(n)$ (Definition 7).

□

4.4 Deterministic Reductions for SLNP^S

We state our main theorem, giving deterministic polynomial-parameter recovery reductions for our class of functions.

Theorem 5. ***(Theorem 1 Restated)***
Suppose we have a function f in the class SLNP^S. Then, for every $\epsilon > 0$, we have a $(poly\,(n, 1/\epsilon)\,, 0.5 - \epsilon, \exp\left(-\Omega(poly(n))\right))$-recovery reduction in the random noise model for f.

Proof. First, given the input $\phi \in \Sigma^{p(n)}$, we compute the size of the automorphism group $Aut_{S_m}(\phi)$ using the deterministic procedure implied by Lemma 6. This takes $poly(n)$-time. If $|Aut_{S_m}(\phi)| \geq m!/\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right)$, we use the deterministic procedure for highly symmetric instances specified in Lemma 11 to compute $f_n(\phi)$ in $poly(n, 1/\epsilon)$-time. Otherwise, if $|Aut_{S_m}(\phi)| < m!/\left(16p(n)\ln(|\Sigma|)/\epsilon^2\right)$, then we use the $poly(n, 1/\epsilon)$-time and $poly(n, 1/\epsilon)$-query procedure specified in Lemma 8 that returns the correct answer with probability at least $1 - 1/|\Sigma|^{p(n)} = 1 - \exp\left(-\Omega(poly(n))\right)$ over the choice of random corruptions of the operator $\mathcal{N}_{0.5-\epsilon}$.

□

Remark 3. Our paradigm so far has been to query on isomorphic instances wherever possible and compute the answer ourselves when we cannot obtain probabilistic guarantees. We note here that this paradigm falls short for some functions and does not always hold as a black-box paradigm. The example is the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n+1}$ multiplying two n bit integers together. Consider the task of multiplying two n-bit prime numbers p and q larger than 2^{n-1}. If we wanted to use the "query isomorphic instances" idea, we can only use the inputs (p, q) and (q, p) since these are the only factorizations of pq with both inputs representable in n bits. The probabilistic guarantees here are poor. If we apply $\mathcal{N}_{0.49}$ to the multiplication table, with probability $1 - 0.51^2 = 0.7399$, at least one of the truth table entries for the inputs (p, q) and (q, p) is false and looking for a majority is no longer a fruitful approach. Hence, we must manually compute this answer ourselves. Due to the work of Harvey and van der Hoeven (2021), there is a $O(n \log n)$ time algorithm for multiplying

2 n bit integers. Under the *Network Coding Conjecture* (Li and Li, 2004; Langberg and Médard, 2009), this algorithm was shown to be optimal (Afshani et al., 2019). If integer multiplication has a non-trivial recovery reduction ($o(n \log n)$-time) under our paradigm, then for every n-bit prime pair p and q, both larger than 2^{n-1}, we would be able to multiply them in $o(n \log n)$-time. It seems unlikely to us that multiplying prime numbers should be asymptotically easier than multiplying arbitrary n bit numbers.

5 Random Noise Reductions for NP-Hard Problems

We now give the lemma proving that our NP-hard problems are indeed contained in the class SLNP^S. The proof is straightforward, but tedious, so we provide the proof in Appendix B.

Lemma 12. *The following functions are contained in the class* SLNP^S *(as defined in Definition 7).*

1. SAT
2. kSAT
3. kCSP
4. *Max-*kCSP
5. INDSET
6. VERTEXCOVER
7. CLIQUE
8. kCOLOR
9. HAMCYCLE
10. HAMPATH

Now, we complete the proof of our main theorem, putting all the ingredients together.

Theorem 6. ***(Theorem 2 Restated)***
For every $\epsilon > 0$*, the following functions have deterministic* $(poly\left(n, 1/\epsilon\right), 0.5 - \epsilon, \exp\left(-\Omega\left(poly(n)\right)\right))$*-recovery reductions in the random noise model:*

1. SAT
2. kSAT
3. kCSP
4. *Max-*CSP
5. INDSET

6. VERTEXCOVER

7. CLIQUE

8. kCOLOR

9. HAMCYCLE

10. HAMPATH

Proof. We note that the semilattices in question are $(\{0,1\},\vee)$ and $(\mathbb{Z},\max)$ and $b\vee b=b$ for every $b\in\{0,1\}$ and $\max\{r,r\}=r$ for every $r\in\mathbb{Z}$. Hence, this follows immediately from Lemma 12 and Theorem 1.

□

6 Random Noise Reductions for Fine-Grained Problems

Now, we provide the claimed recovery reductions for fine-grained problems. We start with the OV problem.

Theorem 7. ***(Theorem 4, Part 1 Restated)***
For every $\epsilon = 1/polylog(n)$ and dimension $d = O(n^{1-\gamma})$ for some $\gamma > 0$, there is a $\left(\tilde{O}(nd), 0.5-\epsilon, 1-\exp(-\Omega(nd))\right)$-recovery reduction in the random noise model for the OV *problem in d dimensions.*

Proof. Suppose the input is given as n, dimension d vectors $V=(v_1,v_2,\ldots,v_n)$. First, we remark that we use the group action $\alpha : S_n \times \{0,1\}^{nd} \to \{0,1\}^{nd}$ such that $\alpha_\pi(V) = \left(v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}\right)$.

Notice that if V had four or more distinct vectors, then the index of the automorphism group $Aut_{S_n}(V)$ is at least $\Omega(n^3)$. Hence, if the input V has at most three distinct vectors, in $O(nd)$-time, we can scan the input, list the (at most three) distinct vectors and compute $O(1)$ dot products in time $O(nd)$. If one of the possible dot products is 0, we return 1 for the OV problem.

Now, suppose the input V has four or more distinct vectors - we can check this in $O(nd)$-time. Then the automorphism group is of size $O(n!/n^3) = o(n!/n^2)$ for all sufficiently large n. Now, suppose we have the corrupted truth table $\mathtt{T}'_f = \mathcal{N}_{0.5-\epsilon}\mathtt{T}_f$. Suppose for each instance V with automorphism group index $\omega(n^2)$, we define $S_V = Orb_{S_n}(V)$[15]. Now, due to Lemma 7, with probability $1-2^{-nd}$, every such set S_V with $Aut_{S_n}(V)$ index size at least $\omega(n^2) > 16nd\ln(2)/\epsilon^2$ has at least $(0.5+\epsilon/2)$-fraction of its instances correct. Now, our strategy is to sample $O(\log n/\epsilon)$ random permutations from S_n, query the corrupted truth table $\mathtt{T}'_f$ on the input $\alpha_\pi(V)$ for each sampled permutation π. Due to the Chernoff bound (Mitzenmacher and Upfal, 2017), with probability at least $1-1/n > 2/3$, the majority is

[15]If the index of the automorphism group $Aut_{S_n}(V)$ is $O(n^2)$, we set $S_V = \{0,1\}^{nd}$ as a formality.

the correct answer. This takes $\tilde{O}(nd/\epsilon^2) = \tilde{O}(nd)$-time. Hence, we have our randomized recovery reduction.

□

Now, we provide our recovery reduction for the *Parity k-Clique* Problem.

Theorem 8. ***(Theorem 4, Part 2 Restated)***
For any constant $k > 0$ and $\epsilon = 1/polylog(n)$, we have a $\left(\tilde{O}(n^2), 0.5 - \epsilon, 1 - 2^{-\binom{n}{2}}\right)$-recovery reduction in the random noise model for Parity k-Clique, the problem of computing the lowest order bit of the number of k cliques in a graph.

Proof. First, we note that due to Lemma 17 and Lemma 18 in Appendix C, we have that for sufficiently large n, if a graph has automorphism group size $|Aut(H)| = \omega\left(n!/n^3\right)$, then it is one of twelve graphs, each of which we can recognize and count the number of k-cliques on (and hence the parity bit) in $\tilde{O}(n^2)$-time.

If it is not one of those twelve graphs, then $|Aut(H)| = O\left(n!/n^3\right)$, and hence, the index of the automorphism group is $\Omega\left(n^3\right) > 16\ln(2)n^2/\epsilon^2$. In this case, Lemma 7 says that with probability at least $1 - 2^{-\binom{n}{2}}$ over the choice of corruptions, the corrupted truth table $\mathtt{T}'_f = \mathcal{N}_{0.5-\epsilon}\mathtt{T}_f$ has at least $(0.5 + \epsilon/2)$-fraction of entries unflipped over every predefined set S_H of size at least $16\ln(2)n^2/\epsilon^2$. Hence, for each graph H with $Aut(H) = O\left(n!/n^3\right)$, we define $S_H = Orb_{S_n}(H)$ and $S_H = \{0,1\}^{\binom{n}{2}}$ for the other twelve cases. Hence, since $Aut(H) = O\left(n!/n^3\right)$, we can query the truth table on $O\left(\log n/\epsilon\right)$ random members of $Orb_{S_n}(H)$ and return the majority value. We do this by randomly sampling permutations π from S_n and querying $\mathtt{T}'_f$ on $\alpha_\pi(H)$ for the $O\left(\log n/\epsilon\right)$ randomly chosen permutations π. Due to the Chernoff bound (Mitzenmacher and Upfal, 2017), with probability $1 - 1/n > 0$ over the randomness of our algorithm, we get the correct majority answer for *Parity k-Clique* on the input H.

□

Remark 4. Note that in both cases, we use randomness so we can perform our reductions in almost-linear time. This is because the deterministic permutation algorithms of Sims (1970) and Furst et al. (1980) tend to add large polynomial overheads to the algorithm.

Acknowledgments

We are grateful to Emanuele Viola for useful feedback on presentation and for pointing us to Akavia (2008) and Akavia et al. (2006).

References

Abboud, A. and Williams, V. V. (2014). Popular conjectures imply strong lower bounds for dynamic problems. In *IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 434–443. 1

Afshani, P., Freksen, C. B., Kamma, L., and Larsen, K. G. (2019). Lower Bounds for Multiplication via Network Coding. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 10:1–10:12. 20

Ajtai, M. (1996). Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 99–108. 3

Akavia, A. (2008). *Learning noisy characters, multiplication codes, and cryptographic hardcore predicates*. PhD thesis, Massachusetts Institute of Technology. 2, 22

Akavia, A., Goldreich, O., Goldwasser, S., and Moshkovitz, D. (2006). On basing one-way functions on NP-hardness. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 701–710. 3, 22

Angluin, D. and Laird, P. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370. 2

Asadi, V. R., Golovnev, A., Gur, T., and Shinkar, I. (2022). Worst-case to average-case reductions via additive combinatorics. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1566–1574. 1

Asadi, V. R., Golovnev, A., Gur, T., Shinkar, I., and Subramanian, S. (2024). Quantum worst-case to average-case reductions for all linear problems. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2535–2567. 1

Babai, L. and Laplante, S. (1999). Stronger separations for random-self-reducibility, rounds, and advice. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity (CCC)*, pages 98–104. 3

Ball, M., Rosen, A., Sabin, M., and Vasudevan, P. N. (2017). Average-case fine-grained hardness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 483–496. 1, 6

Berman, L. and Hartmanis, J. (1977). On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322. 5, 7

Bogdanov, A. and Trevisan, L. (2006). On worst-case to average-case reductions for NP problems. *SIAM Journal on Computing*, 36(4):1119–1159. 1, 2, 3, 6

Boix-Adserà, E., Brennan, M., and Bresler, G. (2019). The average-case complexity of counting cliques in Erdös–Rényi hypergraphs. In *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1256–1280. 1, 6

Cai, J., Pavan, A., and Sivakumar, D. (1999). On the hardness of permanent. In *16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 90–99. 1

Cesa-Bianchi, N., Dichterman, E., Fischer, P., Shamir, E., and Simon, H. U. (1999). Sample-efficient strategies for learning in the presence of noise. *Journal of the ACM*, 46(5):684–719. 2

Chen, J., Huang, X., Kanj, I. A., and Xia, G. (2006). Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367. 7

Dalirrooyfard, M., Lincoln, A., and Williams, V. V. (2020). New techniques for proving fine-grained average-case hardness. In *IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 774–785. 1, 6

Demillo, R. A. and Lipton, R. J. (1978). A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195. 1

Dubhashi, D. and Ranjan, D. (1996). Balls and bins: A study in negative dependence. *Basic Research in Computer Science (BRICS)*. 14

Erdős, P. and Rényi, A. (1963). Asymmetric graphs. *Acta Mathematica Academiae Scientiarum Hungaricae*, 14:295–315. 2, 11

Feige, U. and Lund, C. (1996). On the hardness of computing the permanent of random matrices. *Computational Complexity*, 6:101–132. 1

Feigenbaum, J. and Fortnow, L. (1993). Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005. 2, 6

Feigenbaum, J., Fortnow, L., Lund, C., and Spielman, D. (1992). The power of adaptiveness and additional queries in random-self-reductions. In *Proceedings of the Seventh Annual Structure in Complexity Theory Conference*, pages 338–346. 3

Flajolet, P., Gardy, D., and Thimonier, L. (1992). Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39(3):207–229. 10

Furst, M., Hopcroft, J., and Luks, E. (1980). Polynomial-time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 36–41. 11, 14, 22

Gemmell, P. and Sudan, M. (1992). Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174. 1

Goldreich, O. (2020). On counting t-cliques mod 2. *Electronic Colloquium on Computational Complexity*, Report No. 104(Revision 3):1–6. 1, 6

Goldreich, O. and Rothblum, G. (2018). Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In *IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 77–88. 1, 7

Harvey, D. and van der Hoeven, J. (2021). Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193:563–617. 19

Impagliazzo, R. and Paturi, R. (2001). On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375. 8

Impagliazzo, R., Paturi, R., and Zane, F. (2001). Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530. 8

Kalai, A. and Servedio, R. A. (2003). Boosting in the presence of noise. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, pages 195–205. 2

Kane, D. M. and Williams, R. R. (2019). The orthogonal vectors conjecture for branching programs and formulas. In *10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 48:1–48:15. 1

Langberg, M. and Médard, M. (2009). On the multiple unicast network coding, conjecture. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing*, pages 222–227. 20

Levin, L. A. (1986). Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286. 1

Li, Z. and Li, B. (2004). Network coding: The case of multiple unicast sessions. In *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing*. 20

Micciancio, D. (2004). Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. *SIAM Journal on Computing*, 34(1):118–169. 3

Mitzenmacher, M. and Upfal, E. (2017). *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, second edition. 10, 14, 21, 22

Naik, A., Ogiwara, M., and Selman, A. (1993). P-selective sets, and reducing search to decision vs. self-reducibility. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 52–64. 3

Pólya, G. (1937). Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen. *Acta Mathematica*, 68:145–254. 2, 11

Schwartz, J. T. (1980). Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717. 1

Sims, C. C. (1970). Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra*, pages 169–183. Pergamon. 11, 14, 22

Smith, J. D. H. (2015). *Introduction to Abstract Algebra*. CRC Press, second edition. 12, 13

Sudan, M. (1996). Maximum likelihood decoding of Reed Solomon codes. In *Proceedings of 37th Conference on Foundations of Computer Science (FOCS)*, pages 164–172. 1

Sudan, M., Trevisan, L., and Vadhan, S. (2001). Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266. 1

Williams, R. (2005). A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365. 7

Williams, V. V. (2018). On some fine-grained questions in algorithms and complexity. *Proceedings of the International Congress of Mathematicians (ICM)*, pages 3447–3487. 1

Williams, V. V. and Williams, R. R. (2018). Subcubic equivalences between path, matrix, and triangle problems. *Journal of the ACM*, 65(5):27:(1–38). 1

Zhang, Y., Guo, Z., and Rekatsinas, T. (2020). A statistical perspective on discovering functional dependencies in noisy data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 861–876. 2

Zippel, R. (1979). Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation (EUROSAM)*, pages 216–226. 1

A Properties of Our Generalized Functions

Lemma 13. *(Lemma 1 restated)*
The complexity class NP *is contained in the function class* SLNP.

Proof. Note that $(\{0,1\}, \vee)$ is a semilattice where $y_1 \vee y_2$ is always computable in constant time. From the definition of NP, for a language $L \in NP$, we can write

$$f(\phi) = \bigvee_{x \in \mathcal{C}} M(\phi, x),$$

where f is the indicator function for L, $\mathcal{C}$ is the set of polynomial-sized certificates, and the verifier M runs in $poly(n)$ time. The set $\mathcal{C}$ of certificates is of size $2^{|x|} = 2^{poly(n)}$.

□

Lemma 14. ***Isomorphism Invariance Property of*** SLNP^G *(Lemma 2 restated)*
Given an indicator function $f \in \mathsf{SLNP}^G$, with group sequence $G_1, G_2, \ldots$ and group actions α and β, we have that for every element $g \in G_n$ and input $\phi \in \Sigma^{p(n)}$,

$$f_n\left(\alpha_g(\phi)\right) = f_n(\phi).$$

Proof. First we expand out $f_n\left(\alpha_g(\phi)\right)$ as

$$f_n\left(\alpha_g(\phi)\right) = \bigodot_{x \in \mathcal{C}_n} h_n\left(\alpha_g(\phi), x\right) = \bigodot_{x \in \mathcal{C}_n} h_n\left(\alpha_g(\phi), \beta_g\left(\beta_{g^{-1}}(x)\right)\right).$$

Since the group action $\beta_{g^{-1}}$ is a permutation of the set $\mathcal{C}_n$ and the operator $\odot$ is commutative, we can rearrange this "big sum" as

$$f_n\left(\alpha_g(\phi)\right) = \bigodot_{x \in \mathcal{C}_n} h_n\left(\alpha_g(\phi), \beta_g(x)\right).$$

Since, pointwise for every $x \in \mathcal{C}_n$, by definition of SLNP^G (Definition 6), we have that $h_n\left(\alpha_g(\phi), \beta_g(x)\right) = h_n(\phi, x)$, we rewrite this sum as

$$f_n\left(\alpha_g(\phi)\right) = \bigodot_{x \in \mathcal{C}_n} h_n\left(\alpha_g(\phi), \beta_g(x)\right) = \bigodot_{x \in \mathcal{C}_n} h_n\left(\phi, x\right) = f_n(\phi),$$

and subsequently prove the desired equality.

□

B Our NP-Hard Problems are in SLNP^S

Lemma 15. *(Lemma 12 restated)*
The following functions are in the class SLNP^S as defined in Definition 7.

1. SAT
2. kSAT
3. kCSP
4. *Max-k*CSP
5. INDSET
6. VERTEXCOVER
7. CLIQUE
8. kCOLOR
9. HAMCYCLE
10. HAMPATH

Proof. 1. Suppose we are given SAT formulae of length at most $m(n)$ (polynomial in n) (in any measure) on n variables. Since SAT is in NP, due to Lemma 1, SAT is in SLNP. We use the group S_n to act on the input ϕ and the assignment x as follows. For any permutation $\pi \in S_n$, $\alpha_\pi(\phi)$ relabels every variable x_i as $x_{\pi(i)}$. Given the assignment $x' = (x'_1, x'_2, \ldots, x'_n) \in \{0,1\}^n$, the permutation $\pi \in S_n$ acts on x' as $\beta_\pi\left(x'_1, x'_2, \ldots, x'_n\right) = \left(x'_{\pi^{-1}(1)}, x'_{\pi^{-1}(2)}, \ldots, x'_{\pi^{-1}(n)}\right)$ - the entry in the j^{th} position moves to the $\pi(j)^{\text{th}}$ position. In the view that $f(\phi) = \bigodot_{x \in \{0,1\}^n} M(\phi, x)$, it is easy to see that for every $\pi \in S_n$,

$$M\left(\alpha_\pi(\phi), \beta_\pi(x)\right) = M(\phi, x),$$

and that both group actions α and β are computable in $poly(n)$-time. The challenge remains to show that $\{0,1\}^n$ partitions into $poly(n)$ orbits under the action of β and that we can deterministically sample a list of elements of $\{0,1\}^n$ in each orbit in $poly(n)$-time. This is, indeed, the case. The group action β partitions $\{0,1\}^n$ into $n+1$ partitions on the basis of the number of 0s in the string. A representative of the orbit with k 0s is $0^k 1^{n-k}$. Hence, with all relevant polynomial parameters, SAT is in SLNP^S.

2. This proof follows almost immediately from the proof for SAT. The only difference here is that the input ϕ has clause-width at most k. We can also represent ϕ as a string in $\{0,1\}^{\sum_{j\in[k]}\binom{2n}{k}}$ choosing or not choosing one of the $\sum_{j\in[k]}\binom{2n}{j}$ unrestricted clauses of length at most k.

3. For kCSP, we have a clause $C:\Sigma^k\to\{0,1\}$ with an arbitrary truth table. One can see that any $C(x_1,\ldots,x_k)$ is computable in $O(1)$ time. We construct our function $h_n:\{0,1\}^{(q(n))^k}\times\mathcal{C}_n\to\{0,1\}$ where $\mathcal{C}_n=\Sigma^{q(n)}$ and q is a polynomial representing the length of the certificate. Here, we have that

$$h_n(\phi,y)=\bigwedge_{(i_1,i_2,\ldots,i_k)\in[q(n)]^k} C_{\text{select}}\left(\phi,y_{i_1},y_{i_2},\ldots,y_{i_k}\right),$$

where

$$C_{\text{select}}(\phi,y_1,y_2,\ldots,y_k)=\begin{cases}C\left(y_1,y_2,\ldots,y_k\right), & \text{if } \phi \text{ selects the tuple } (i_1,i_2,\ldots,i_k);\\ 1, & \text{otherwise.}\end{cases}$$

Note that h can be computed in polynomial time since q is a polynomial in n. We can see ϕ as a $(q(n))^k$-bits long string selecting and deselecting clauses in the kCSP instance. Our string y is a proof from the set $\Sigma^{q(n)}$. Our kCSP can be represented as

$$f_n(\phi)=\bigvee_{y\in\Sigma^{q(n)}} h_n(\phi,y).$$

Now, we describe our group actions $\alpha:S_{q(n)}\times\{0,1\}^{(q(n))^k}\to\{0,1\}^{(q(n))^k}$ and $\beta:S_{q(n)}\times\Sigma^{q(n)}\to\Sigma^{q(n)}$. The simpler group action β acts on $y=\left(y_1,y_2,\ldots,y_{q(n)}\right)$ to give $\beta_\pi(y)=\left(y_{\pi(1)},y_{\pi(2)},\ldots,y_{\pi(q(n))}\right)$ for every permutation $\pi\in S_{q(n)}$. The group action α acts to preserve the relation, with the bit $\alpha_\pi(\phi)_{(i_1,i_2,\ldots,i_k)}=\phi_{(\pi^{-1}(i_1),\pi^{-1}(i_2),\ldots,\pi^{-1}(i_k))}$ for every permutation $\pi\in S_{q(n)}$. It can be seen that the relation $h\left(\alpha_\pi(\phi),\beta_\pi(y)\right)=h(\phi,y)$ is preserved for every $\pi\in S_{q(n)}$, instance $\phi\in\{0,1\}^{(q(n))^k}$, and certificate $y\in\Sigma^{q(n)}$. Note that q is a polynomial and $S_{q(n)}$ is the group and hence, this is in SLNP^S for every possible clause $C:\Sigma^k\to\{0,1\}$.

4. The proof proceeds identically to the above case, except we use $(\mathbb{Z},\max)$ as the semi-lattice.

For cases 5 through 10, due to Lemma 1, it is easy to see that these are in SLNP. Now, we prove that they are in SLNP^S due to action from S_n on n vertex graph inputs. The group action α always acts on a graph H by permuting the graph according to permutation π, where the graph $\alpha_\pi(H)$ contains an edge between $v_{\pi(i)}$ and $v_{\pi(j)}$ if and only if the graph H has an edge between v_i and v_j. Now, we prove that for each of these problems, we have a group action β preserving symmetry and splitting the certificate space $\mathcal{C}$ into $poly(n)$-partitions, each of which we can deterministically produce one canonical certificate from. Since one can verify the parameters, this is sufficient to show that these functions are in SLNP^S.

5. Here, the certificate is a set $S \subset [n]$. The group action β_π acts on S by giving us a set $\beta_\pi(S)$ that contains $\pi(i)$ if and only if S contains i. One can see that the set $\beta_\pi(S)$ is an independent set for the graph $\alpha_\pi(H)$ if and only if the set S is an independent set for the graph H. Here, the orbits partition the power set of $[n]$, $\mathcal{P}([n])$ into $n+1$ partitions on the basis of size. We can easily sample the set $\{1, 2, \ldots, k\}$ as the canonical certificate of size k. Here, our function h, on input k, graph H and set S checks if S encodes an independent set for the graph H and that $|S| = k$.

6. Here, this follows identically to the above case except the function h checks if the set S is a vertex cover of size k for the graph H.

7. As above, the proof proceeds identically, except the function h checks if the set S forms a clique of size k in the graph H.

8. Here, our certificate is coloring $C = (c_1, c_2, \ldots, c_n) \in [k]^n$ of the vertices. The function h checks if the coloring C is a valid k-coloring of the graph H, with no monochromatic edges. The group action β acts on the coloring C by producing $\beta_\pi(C) = \left(c_{\pi^{-1}(1)}, c_{\pi^{-1}(2)}, \ldots, c_{\pi^{-1}(n)}\right)$. One can see that $\beta_\pi(C)$ is a valid coloring of the graph $\alpha_\pi(H)$ is and only if C is a valid coloring for the graph H. It can also be seen that there are $\binom{n+k-1}{k-1} = O\left(n^{k-1}\right)$ orbit partitions on the basis of the number of each color $c \in [k]$ present in the coloring C - we can sample one canonical member of each orbit in polynomial-time.

9. Here, the certificate is a sequence $I = (i_1, i_2, \ldots, i_n)$ where each distinct $i_k \in [n]$ and this represents a cycle $i_1 \to i_2 \to \cdots \to i_n \to i_1$. The function h checks if the sequence is a valid Hamiltonian cycle for the graph H. We note the redundancy but introduce it for the sake of symmetry. The action β_π acts on the sequence I by returning $\beta_\pi(I) = \left(i_{\pi^{-1}(1)}, i_{\pi^{-1}(2)}, \ldots, i_{\pi^{-1}(n)}\right)$. One can see that there is only one orbit and that the relation is preserved upon group action.

10. Here, the certificate is once again a sequence $I = (i_1, i_2, \ldots, i_n)$ as before, representing the path $i_1 \to i_2 \to \cdots \to i_n$. Once again, we note that we introduce the redundancy in certification to allow for symmetry. The function h checks if the path encoded by the sequence I is a Hamiltonian path in the graph H. We use the group action β identically to before and there is only one orbit, with the group actions preserving the relation.

□

C A Classification of Graphs With $|Aut(H)| = \omega(n!/n^3)$

First, we will prove Lemma 16 that gives the properties of the n vertex graphs with $|Aut(H)| = \omega(n!/n^3)$ in terms of the number of partitions based on the degree of vertices, the size of such partitions, and the degree distribution of the vertices.

Lemma 16. *For sufficiently large n, any graph H with $|Aut(H)| = \omega(n!/n^3)$ satisfies the following properties.*

1. *If the vertices of H are partitioned based on degree, then there are at most three partitions.*

2. *No partition can be simultaneously larger than 2 and smaller than $n-2$.*

3. *The degree of any vertex v can be in the set $\{0, 1, 2, n-2, n-1\}$.*

Proof. Suppose that we have $m \geq 4$ partitions of size $(\alpha_i)_{i\in[m]}$, in ascending order, with each $\alpha_i \geq 1$. The probability that $\pi \in S_n$ is in $Aut(H)$ is bounded from above by

$$\frac{\prod_{i\in[m]} \alpha_i!}{n!} \leq \frac{(\sum_{i\in[m-1]} \alpha_i)!\alpha_m!}{n!},$$

since π is not allowed to permute vertices across partitions. Since $\alpha_m \geq n/m$ due to the pigeonhole principle, and $\alpha_m \leq n-(m-1)$ due to each α_i being positive, we have that

$$\frac{\left(\sum_{i\in[m-1]} \alpha_i\right)!\alpha_m!}{n!} = \frac{1}{\binom{n}{\alpha_m}} \leq \frac{1}{\binom{n}{3}} = O\left(\frac{1}{n^3}\right).$$

If this is the case, then $|Aut(H)|$ is upper bounded by $O(n!/n^3)$, leading to a contradiction. This proves the first statement of the lemma.

If there is a partition of size α, then the probability that $\pi \in S_n$ is in $Aut(H)$ is upper bounded by $1/\binom{n}{\alpha} = O\left(1/n^3\right)$ for the forbidden range. This implies the second statement of the lemma.

Let us consider that the degree m of v in H is greater than 2 and less than $n-2$. Let the neighbors of v be $(u_i)_{i\in[m]}$. The probability that $\pi \in S_n$ is in $Aut(H)$ is bounded from above by

$$\frac{n(m!)(n-1-m)!}{n!} = \frac{1}{\binom{n-1}{m}} \leq \frac{1}{\binom{n-1}{3}} = O(1/n^3),$$

since n is the maximum number of vertices v could map to, $m!$ is the number of ways the neighbors of v could distribute themselves among the neighbors of the image of v, and $(n-m-1)!$ is the number of ways the remaining vertices can distribute. Also, m is between 3 and $n-3$. Due to a similar argument as before, this implies the third statement of the lemma.

□

Now, using Lemma 16, we will prove Lemma 17 that gives the structure of the graphs with $|Aut(H)| = \omega(n!/n^3)$.

Lemma 17. *Only the following graphs have $|Aut(H)| = \omega(n!/n^3)$.*

1. *K_n and its complement.*

2. K_n *with one edge missing and its complement.*

3. K_{n-1} *with an isolated vertex and its complement.*

4. K_{n-1} *with one vertex of degree* 1 *adjacent to it and its complement.*

5. K_{n-2} *with two isolated vertices and its complement.*

6. K_{n-2} *with two vertices of degree* 1 *adjacent to each other and its complement.*

Proof. Using Lemma 17, the only possible partition sizes based on degree we can have are (n), $(n-1,1)$, $(n-2,1,1)$ and $(n-2,2)$. Now, by a case-by-case analysis, we will determine which graphs can have such large automorphism groups. Since $Aut(H) = Aut\left(\overline{H}\right)$, we will categorize by the degree of the largest partition and assume that the degree is less than or equal to 2. This way, we will either allow a graph and its complement or reject both. We will also assume that n is sufficiently large, say $n \geq 100$.

Case 1: The Largest Partition Degree is 0***.***
Now, for the (n) partition, the graph is either empty or the complete graph K_n. Clearly,

$$|Aut(H)| = n! = \omega\left(\frac{n!}{n^3}\right),$$

in both cases, so we allow both.

When the partition is $(n-1,1)$, this is technically not allowed since even the vertex of the partition of size 1 must have degree zero, meaning such a partition with these degrees cannot exist.

When the partition is $(n-2,1,1)$, this cannot exist since the partitions of size 1 must have the same degree.

When the partition is $(n-2,2)$, the only allowed case is that both the vertices in the partition of size 2 are adjacent. Otherwise, they would also have degree 0, and we would have (n) again. The other case is K_n with one edge missing. Both of them have

$$|Aut(H)| = 2(n-2)! = \frac{n!}{O(n^2)} = \omega\left(\frac{n!}{n^3}\right),$$

hence, we allow them both.

From this case, we allow the graphs as described in statements 1 and 2 of the lemma.

Case 2: The Largest Partition Degree is 1***.***
For the partition type (n), this is only allowed when n is even due to the handshake lemma. When so, the vertices arrange themselves in pairs. Visually, we have $n/2$ "sticks". We can permute these sticks in $(n/2)!$ ways and flip them in $2^{n/2}$ ways. In particular, the size of the automorphism group is

$$|Aut(H)| = \left(\frac{n}{2}\right)! \cdot 2^{n/2} \leq \frac{n!}{n^3} = O\left(\frac{n!}{n^3}\right),$$

for sufficiently large n. Hence, we reject this case.

For the partition type $(n-1,1)$, we have the following possibilities: The vertex in the partition of size 1 may have possible degrees $n-1$, $n-2$, 2, or 0.

1. The vertices in the $n-1$-partition are all adjacent to the vertex in the 1-partition. This is allowed, with

$$|Aut(H)| = (n-1)! = \frac{n!}{n} = \omega\left(\frac{n!}{n^3}\right),$$

and hence, we allow K_{n-1} with an isolated vertex and its complement graph. This covers the case where the 1-partition vertex has degree $n-1$.

2. If the degree of the 1-partition vertex is $n-2$, this is disallowed for the following reason: The vertex in the $n-1$-partition not adjacent to the 1-partition vertex must be adjacent to one of the other vertices, if it needs a degree of 1. This creates a vertex of degree 2 in the $n-1$-partition.

3. The degree of the 1-partition vertex is 2. In this case, we have two vertices u and v in the $n-1$-partition that are adjacent to the 1-partition vertex. The others are arranged similarly to the (n) case for degree 1. Here, the automorphism group size is

$$|Aut(H)| = 2\left(\frac{n-3}{2}\right)! \cdot 2^{(n-3)/2} = O\left(\frac{n!}{n^3}\right),$$

which for sufficiently large n is too small; hence we reject this case when n is odd. The graph is not possible when n is even

4. If the degree of the 1-partition vertex is 0 and the others have degree 1, this suffers from the same pitfalls as the (n) case, having an automorphism group of size

$$|Aut(H)| = \left(\frac{n-1}{2}\right)! \cdot 2^{(n-1)/2} = O\left(\frac{n!}{n^3}\right),$$

and hence, we reject this case as well when n is odd. The graph is not possible when n is even.

For the partition type $(n-2,1,1)$, let the two 1-partition vertices be u_1 and u_2 with degrees d_1 and d_2, respectively. Without loss of generality, assume that $d_2 > d_1$. Since the unique degrees d_1 and d_2 are different, the $n-2$-partition implicitly partitions itself into three parts: The partition that is adjacent to the vertex u_1 of size α_1, the partition that is adjacent to the vertex u_2 of size α_2, and the remaining vertices that pair themselves. These partitions are rigid in that no π from the automorphism group can map vertices across the partition. Hence, assuming the correct parity for n, the probability that a random π from S_n is in the automorphism group is

$$\begin{aligned}
\frac{|Aut(H)|}{n!} &\leq \frac{\alpha_1!\alpha_2!\left(\frac{n-\alpha_1-\alpha_2-2}{2}\right)!2^{(n-\alpha_1-\alpha_2-2)/2}}{n!} \\
&\leq \frac{d_1!d_2!\left(\frac{n-\alpha_1-\alpha_2-2}{2}\right)!2^{(n-\alpha_1-\alpha_2-2)/2}}{n!} \\
&= O\left(\frac{1}{n^3}\right),
\end{aligned}$$

if d_1 and d_2 are both from the set $\{0, 2\}$. Therefore, d_2 is either $n-1$ or $n-2$. The value of d_2 cannot be $n-1$, since then u_2 is connected to all the other vertices, forcing $d_1 = 1$, which is not allowed. The only possibility that remains is $d_2 = n-2$. If $d_1 = 0$, then the vertex u_1 is isolated, and u_2 is connected to all vertices in the $n-2$-partition. In this case, we have

$$|Aut(H)| = (n-2)! = \frac{n!}{O(n^2)} = \omega\left(\frac{n!}{n^3}\right),$$

so that we allow this graph. We also allow the complement of this graph, a K_{n-1} with a vertex of degree 1 adjacent to it.

If $d_2 = n-2$ and $d_1 = 2$, then the vertex u_2 is connected to all but one vertex in the $n-2$-partition, and it is also connected with the vertex u_1. The vertex u_1 is also connected with the isolated vertex in the $n-2$-partition. In this case, we have

$$|Aut(H)| = (n-3)! = \frac{n!}{O(n^3)} = O\left(\frac{n!}{n^3}\right),$$

so that this graph is rejected.

For the $(n-2, 2)$ case, we have two vertices v_1, and v_2 of degree d. We have the following cases.

1. If $d = 0$, we have a case similar to that of (n) with degree 1, where the automorphism group size is

   $$|Aut(H)| = 2 \cdot 2^{(n-2)/2}\left(\frac{n-2}{2}\right)! = O\left(\frac{n!}{n^3}\right).$$

   We disallow this case when n is even. When n is odd, the graph is not possible.

2. If $d = 1$, this is not allowed since we have defined the partition class this way.

3. For $d = 2$, this implicitly partitions the $n-2$-partition into two parts: Adjacent to a vertex of degree 2 and not adjacent to a vertex of degree 2. Suppose that these vertices are partitioned into partitions of size α_1 and α_2, respectively, the probability that $\pi \in S_n$ is in the automorphism group is

   $$\frac{|Aut(H)|}{n!} \leq \frac{2 \cdot \alpha_1!\alpha_2!}{n!} = \frac{2}{n(n-1)} \cdot \frac{1}{\binom{n-2}{\alpha_1}} = O\left(\frac{1}{n^3}\right),$$

   since α_1 and α_2 are necessarily positive. If they were not, we would either have v_1 and v_2 have very high degree, or degree 0 or 1. We reject this graph.

4. For $d = n-2$ and $d = n-1$, this is not allowed since at least one vertex from the $n-2$-partition would have to have a degree larger than 1.

This case covers the statements 3 and 4 of the lemma.

Case 3: The Largest Partition Degree is 2**.**

For the (n) case, for sufficiently large n, we must have v_1, v_2, v_3, v_4, v_5, and v_6 such that v_1 and v_2 are adjacent, v_2 and v_3 are adjacent, v_4 and v_5 are adjacent, and v_5 and v_6 are adjacent. If we pick a random permutation π from S_n, the probability that it is in the automorphism group is

$$\frac{|Aut(H)|}{n!} \leq \frac{n \cdot 2 \cdot (n-3) \cdot 2 \cdot (n-6)!}{n!} = O\left(\frac{1}{n^4}\right),$$

since v_2 can map to at most n vertices, v_1 and v_3 can only swap their positions as a neighbor of v_2; similarly, v_5 can map to at most $n-3$ vertices, v_4 and v_6 can only swap their positions as a neighbor of v_5, and the remaining vertices can map freely to give an upper bound. Hence, we reject this case.

When we have the partition type $(n-1, 1)$, we have the following cases, based on the degree d of the 1-partition vertex u.

1. If $d = 0$, this graph suffers the same pitfalls as the (n) partition case and has the automorphism group size of

   $$|Aut(H)| \leq \frac{(n-1) \cdot 2 \cdot (n-4) \cdot 2 \cdot (n-7)!}{n!} = O\left(\frac{1}{n^5}\right),$$

   hence, we reject this case.

2. If $d = 1$, suppose v_1 is in the $(n-1)$-partition and adjacent to u. If v_2 is adjacent to v_1, we must find a v_3 adjacent to v_2 since v_3 cannot be adjacent to any of the vertices we already numbered. Otherwise, u's degree would be too high, and a similar case would go for v_1 and v_2. Once we continue this process and reach v_{n-1}, this vertex has no chance of having a degree 2 since all other vertices have their promised degrees. Such a graph does not exist.

3. If $d = 2$, we violate the definition of our partition structure.

4. If $d = n - 2$, we only have one possibility: Suppose u is adjacent to v_1 through v_{n-2}. The vertex v_{n-1} is adjacent to v_1 and v_2. From $i = 1$ onwards, v_{2i+1} is also adjacent to v_{2i+2}. The automorphism group of this graph is of the size of

   $$|Aut(H)| = 2 \cdot 2^{(n-4)/2} \left(\frac{n-4}{2}\right)! = O\left(\frac{n!}{n^3}\right),$$

   since the vertices v_1 and v_2 can swap themselves; and all other remaining $(n-4)/2$ pairs can swap and rearrange themselves. We reject this case when n is even. When n is odd, the graph is not possible.

5. If $d = n - 1$, then the structure would be u, connected to each v_i and the v_i's forming pairs again, like the sticks. The automorphism group is of the size of

   $$|Aut(H)| = 2^{(n-1)/2} \left(\frac{n-1}{2}\right)! = O\left(\frac{n!}{n^3}\right),$$

for sufficiently large n, and we reject this case when n is odd. The graph is not possible when n is even.

When we have a partition structure $(n-2,2)$, we have the following cases, where d is the degree of the 2-partition.

1. If $d=0$, then this suffers from the same asymptotic pitfalls as the (n)-case for degree 2 and we reject this case:
$$|Aut(H)| \leq \frac{2\cdot(n-2)\cdot 2\cdot(n-5)\cdot 2\cdot(n-8)!}{n!} = O\left(\frac{1}{n^6}\right).$$
2. If $d=1$, suppose u_1 and u_2 are from the 2-partition. If u_1 and u_2 are adjacent, this suffers from the same pitfall as the (n)-case again (as shown in case 1 above), and we reject this case. If they are not adjacent, then suppose that v_1 is adjacent to u_1. The vertex v_1 is adjacent to v_2. The vertex v_2 cannot be adjacent to any of the vertices we visited, so we require a new vertex v_3. Similarly, we go on until v_{n-2}. The vertex v_{n-2} must be adjacent to u_2, since all the others already have the promised degree. This resulting graph has an automorphism group size of 2: Only reflectional symmetry. Another alternative is one chain from u_1 to u_2, and a cover of cycles. Once again, the u_1, u_2 component with the chain only has reflectional symmetry, so we have an automorphism group of size
$$|Aut(H)| \leq 2\cdot(n-3)! = \frac{n!}{O\left(n^3\right)} = O\left(\frac{n!}{n^3}\right),$$
and we reject this case.
3. The case of $d=2$ is again not allowed.
4. If $d=n-2$, we have two cases:
   - If u_1 and u_2 are not adjacent, then they are connected to each vertex of the $n-2$-partition. This graph has an automorphism group of size
   $$|Aut(H)| = 2\cdot(n-2)! = \frac{n!}{O\left(n^2\right)} = \omega\left(\frac{n!}{n^3}\right),$$
   and we accept this and its complement: K_{n-2} with the other component being an edge.
   - If u_1 and u_2 are adjacent, then u_1 and u_2 are adjacent to $n-3$ vertices each in the $n-2$-partition. The vertices v_3 through v_{n-2} are adjacent to both, and v_1 (adjacent to u_1) is adjacent to v_2 (adjacent to u_2). The automorphism group size is
   $$|Aut(H)| = 2\cdot(n-4)! = O\left(\frac{n!}{n^4}\right),$$
   and hence, we reject it.

5. If $d = n-1$, then this graph is a complement of K_{n-2} along with two isolated vertices. This graph has an automorphism group size of

$$|Aut(H)| = 2 \cdot (n-2)! = \frac{n!}{O(n^2)} = \omega\left(\frac{n!}{n^3}\right),$$

and we accept this and its complement.

When we have a partition structure of $(n-2, 1, 1)$, we reject. We can categorize this graph as follows: The vertices in the $n-2$-partition form a cycle within the partition, there is a chain starting at u_1 and ending at u_2, or starting at u_i and ending at u_i (for $i = 1$ or 2). There must be at least one such cycle containing some u_i, since otherwise, d_1 would be equal to d_2. Let a be the length of the chain and b be the number of such isomorphic chains. The number of permutations in the automorphism group is

$$|Aut(H)| \leq 2^a a!(n-ab)! = O\left(\frac{n!}{n^3}\right),$$

since b is at least 1 and a is at least 3.

This case covers the statements 5 and 6 of the lemma.

□

Due to the above case-by-case analysis, we have the following lemma.

Lemma 18. *For sufficiently large n, in $\tilde{O}(n^2)$-time, given an n-vertex undirected simple graph H, we can check whether $Aut(H) = \omega(n!/n^3)$ and also compute the number of k-cliques for any $k > 2$.*

Proof. Our algorithm will proceed as follows. If the number of edges in H is larger than $\binom{n}{2}/2$, then we check if it is one of the large-clique structures. If not, then we compute $\overline{H}$ and check for one of the large clique structures. Both counting edges and computing the complement of H requires $O(n^2)$-time. Hence, assuming $U'_n = H$ or $\overline{H}$ with at least $\binom{n}{2}/2$ edges, our algorithm proceeds as follows.

1. *Checking if U'_n is K_n:* Simply check if every entry in U'_n is 1 confirms this. If this test is passed, if $U'_n = U_n$, then the number of k-cliques is $\binom{n}{k}$. If $U'_n = \overline{H}$, then the number of k-cliques is 0. If U'_n does not pass this test, then we move to the next test.

2. *Checking if U'_n is K_n with one missing edge:* Simply checking if exactly one entry in U'_n is 0 confirms this. If the test is passed and $U'_n = H$, then the number of k-cliques is $\binom{n}{k} - \binom{n-2}{k-2}$, since the subtracted number is the number of k-cliques that, in K_n, would contain the excluded edge. If $U'_n = \overline{H}$, then the number of k-cliques is 0. If this test fails, then we move to the next test.

3. *Checking if U'_n is K_{n-1} with an isolated vertex:* It suffices to check if $n-1$ vertices have degree $n-2$ and one has degree 0. If U'_n passes this test and $U'_n = H$, then the number of k-cliques is $\binom{n-1}{k}$. If $U'_n = \overline{H}$, then the number of k-cliques is 0. If this test fails, then we move to the next test.

4. *Checking if U'_n is K_{n-1} with one vertex of degree 1 adjacent to it:* First, we count the degrees of the vertices. If there is agreement with the expected number of vertices of each degree, then we are done, since all vertices with degree $n-2$ must form a K_{n-2} subgraph, all adjacent to the vertex of degree $n-1$, since the vertex of degree $n-1$ is already adjacent to the vertex of degree 1. If this test passes and $U'_n = H$, then the number of k-cliques is $\binom{n-1}{k}$. If $U'_n = \overline{H}$, then the number of k-cliques is 0. If this test fails, then we move to the next test.

5. *Checking if U'_n is K_{n-2} with two isolated vertices:* It suffices in this case to check alignment with the expected degrees of the vertices. If the test passes and $U'_n = H$, then the number of k-cliques is $\binom{n-2}{k}$. If $U'_n = \overline{H}$, then the number of k-cliques is 0. If this test fails, then we move to the next test.

6. *Checking if U'_n is K_{n-2} with two vertices of degree 1 adjacent to each other:* First, we compute the degrees of the vertices and check that the vertices of degree 1 are adjacent to each other. This forces the other $n-2$ vertices to form an $n-2$-clique. If this test passes and $U'_n = H$, then the number of k-cliques is $\binom{n-2}{k}$. If $U'_n = \overline{H}$, then the number of k-cliques is 0. If this test fails as well, and after all other tests, we know from our classification that

$$|Aut(H)| = O\left(\frac{n!}{n^3}\right).$$

In all six cases, in $\tilde{O}(n^2)$-time (depending on one's preferred model of computation), we can determine the appropriate classification if

$$|Aut(H)| = \omega\left(\frac{n!}{n^3}\right),$$

and also compute the number of k-cliques in $\tilde{O}(n^2)$-time. If not, we can determine that

$$|Aut(H)| = O\left(\frac{n!}{n^3}\right).$$

□

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il