Modern Interactive Proofs

by

Dhiraj Holden

B.S., California Institute of Technology (2015)
S.M., Massachussetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author . . . .
Department of Electrical Engineering and Computer Science
January 15, 2022

Certified by . . . .
Shafi Goldwasser
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Modern Interactive Proofs

by

Dhiraj Holden

Submitted to the Department of Electrical Engineering and Computer Science
on January 15, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

In this thesis, we study several extensions of the concept of interactive proofs. First, we consider non-signaling multi-prover interactive proofs. Interacting with multiple non-interacting provers increases the ability of the verifier to check the solution by asking the provers different questions and checking the consistency of their answers. In a non-signaling multi-prover proof, the provers can interact and correlate their answers, but not in an unlimited way: non-signaling provers must make the distribution of answers for any subset of provers only depend on the distribution of questions the verifier sends to that same subset. Non-signaling proofs, have found applications in cryptography and hardness of approximation. An important open problem is characterizing the power of non-signaling proofs. It is known that 2-prover non-signaling proofs are characterized by PSPACE, and that non- signaling proofs with poly(n)-provers are characterized by EXP. However, the power of k-prover non-signaling proofs, for $2 < k < \text{poly}(n)$ remained an open problem. We show that k-prover non-signaling proofs (with negligible soundness) for $k = O(p \log n)$ are contained in PSPACE. We prove this via two different routes that are of independent interest. In both routes we consider a relaxation of non-signaling called sub-non-signaling. Our main technical contribution (which is used in both our proofs) is a reduction showing how to convert any sub-non- signaling strategy with value at least $1 - 2^{-k^2}$ into a non-signaling one with value at least $2^{O(-k^2)}$.

Second, we introduce pseudo-deterministic interactive proofs (psdIP): interactive proof systems for search problems where the verifier is guaranteed with high probability to output the same output on different executions. As in the case with classical interactive proofs, the verifier is a probabilistic polynomial time algorithm interacting with an untrusted powerful prover. We view pseudo-deterministic interactive proofs as an extension of the study of pseudo-deterministic randomized algorithms: the goal of the latter is to find canonical solutions to search problems whereas the goal of the former is to prove that a solution to a search problem is canonical to a probabilistic polynomial time verifier. Alternatively, one may think of the powerful prover as aiding the probabilistic polynomial time verifier to find canonical solutions to search problems, with high probability over the randomness of the verifier. The challenge is that pseudo- determinism should hold not only with respect to the randomness, but also with respect to the prover: a malicious prover should not be able to cause the verifier to output a solution other than the unique canonical one. The IP = PSPACE characterization implies that psdIP = IP. The challenge is to find constant round pseudo-deterministic interactive proofs for hard search prob-lems. We show a constant round pseudo-deterministic interactive proof for the graph isomorphism problem: on any input pair of isomorphic graphs (G_0, G_1), there exist a unique isomorphism from G_0 to G_1 (although many isomorphism many exist) which will be output by the verifier with high probability, regardless of any dishonest prover strategy. In contrast, we show that it is unlikely that psdIP proofs with constant rounds exist for NP-complete problems by showing that if any NP-complete problem has a psdIP protocol, then the polynomial hierarchy collapses.

Third, we define doubly-efficient pseudo-deterministic proofs for polynomial time search problems: pseudo-deterministic proofs with the extra requirement that the prover runtime is polynomial and the verifier runtime to verify that a solution is canonical is significantly lower than the complexity of finding any solution, canonical or otherwise. Naturally this question is particularly interest-ing for search problems for which a lower bound on its worst case complexity is known or has been widely conjectured.

We show doubly-efficient pseudo-deterministic algorithms for a host of natural problems whose complexity has long been conjectured. In particular, linear programming and a variety of problems studied at the center of the fine grained complexity study.

Thesis Supervisor: Shafi Goldwasser
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

This thesis is based on the following papers: Pseudo-Deterministic Proofs [40], Doubly-Efficient Pseudo-Deterministic Proofs [34], and Non-Signaling Proofs with $O(\sqrt{\log n})$ provers are in PSPACE [47].

I would like to thank my advisor Shafi Goldwasser for all the help and support she has given me. I would like to thank my thesis committee, Yael Kalai and Ryan Williams. I would like to thank my collaborators Michel Goemans, Ofer Grossmann, Adam Bouland, Prashant Vasudevan, Justin Thaler, and Lijie Chen. I would like to thank Rahul Santhanam for hosting me at Oxford in the summer of 2018. I would like to thank the people who hung out in G5 Lounge on Friday evenings for all the fun times we had.

Contents

Chapter 1

Introduction

This thesis is putting together the work of three papers.

We introduce *pseudo-deterministic interactive proofs* (psdIP): interactive proof systems for search problems where the verifier is guaranteed with high probability to output the same output on different executions. As in the case with classical interactive proofs, the verifier is a probabilistic polynomial time algorithm interacting with an untrusted powerful prover.

We view pseudo-deterministic interactive proofs as an extension of the study of pseudo-deterministic randomized polynomial time algorithms: the goal of the latter is to *find* canonical solutions to search problems whereas the goal of the former is to *prove* that a solution to a search problem is canonical to a probabilistic polynomial time verifier. Alternatively, one may think of the powerful prover as aiding the probabilistic polynomial time verifier to find canonical solutions to search problems, with high probability over the randomness of the verifier. The challenge is that pseudo-determinism should hold not only with respect to the randomness, but also with respect to the prover: a malicious prover should not be able to cause the verifier to output a solution other than the unique canonical one.

The $IP = PSPACE$ characterization implies that psdIP $= IP$. The challenge is to find constant round pseudo-deterministic interactive proofs for hard search problems. We show a constant round pseudo-deterministic interactive proof for the graph isomorphism problem: on any input pair of isomorphic graphs (G_0, G_1), there exist a unique isomorphism ϕ from G_0 to G_1 (although many isomorphism many exist) which will be output by the verifier with high probability, regardless of any dishonest prover strategy. In contrast, we show that it is unlikely that psdIP proofs with constant rounds exist for NP-complete problems by showing that if any NP-complete problem has a psdIP protocol, then the polynomial hierarchy collapses.

In [40] Goldwasser, Grossman and Holden introduced pseudo-deterministic interactive proofs for search problems where a powerful prover can convince a probabilistic polynomial time verifier that a solution to a search problem is canonical. They studied search problems for which polynomial time algorithms are not known and for which many solutions are possible. They showed that whereas there exists a constant round pseudo deterministic proof for graph isomorphism where the canonical solution is the lexicographically smallest isomorphism, the existence of pseudo-deterministic interactive proofs for NP-hard problems would imply the collapse of the polynomial time hierarchy.

In Chapter 4, we turn our attention to studying *doubly-efficient pseudo-deterministic proofs* for polynomial time search problems, as defined in [34]: pseudo-deterministic proofs with the extra requirement that the prover runtime is polynomial and the verifier runtime to *verify* that a solution is canonical is significantly lower than the complexity of *finding* any solution, canonical or otherwise. Naturally this question is particularly interesting for search problems for which a lower bound on its worst case complexity is known or has been widely conjectured.

We show doubly-efficient pseudo-deterministic algorithms for a host of natural problems whose complexity has long been conjectured. In particular,

- We show a doubly efficient pseudo-deterministic proof for **linear programming** where the canonical solution which the prover will provide is the lexicographically greatest optimal solution for the LP. To this end, we show how through perturbing the linear program and strong duality this solution can be both computed efficiently by the prover, and verified by the verifier. The time of the verifier is $O(d^2)$ for a linear program with integer data and at most d variables and constraints, whereas the time to solve such linear program is $\tilde{O}(d^\omega)$

by randomized algorithms [24] for ω the exponent for fast matrix multiplication

.

- We show a doubly efficient pseudo-deterministic proof for **3-SUM** and problems reducible to 3-SUM where the prover is a $O(n^2)$ time algorithm and the verifier takes time $\tilde{O}(n^{1.5})$.
- We show a doubly-efficient pseudo-deterministic proof for the **hitting set problem** where the verifier runs in time $\tilde{O}(m)$ and the prover runs in time $\tilde{O}(m^2)$ where $m = \sum_{S \in \mathcal{S}} |S| + \sum_{T \in \mathcal{T}} |T|$ for inputs collections of sets $\mathcal{S}, \mathcal{T}$.
- We show a doubly-efficient pseudo-deterministic proof for the **Zero Weight Triangle problem** where the verifier runs in time $\tilde{O}(n^{2+\omega/3})$ and the prover runs in randomized time $\tilde{O}(n^3)$. The Zero Weight Triangle problem is equivalent to the **All-Pairs Shortest Path problem**, a well-studied problem that is the foundation of many hardness results in graph algorithms [88, 87], under sub-cubic reductions.

In the next chapter we turn our attention to non-signaling proofs. Non-signaling proofs, motivated by quantum computation, have found applications in cryptography and hardness of approximation. An important open problem is characterizing the power of non-signaling proofs. It is known that 2-prover non-signaling proofs are characterized by PSPACE, and that non-signaling proofs with poly(n)-provers are characterized by EXP. However, the power of k-prover non-signaling proofs, for $2 < k < \text{poly}(n)$ remained an open problem.

In [47], we show that k-prover non-signaling proofs (with negligible soundness) for $k = O(\sqrt{\log n})$ are contained in PSPACE. We prove this via two different routes that are of independent interest. In both routes we consider a relaxation of no-signaling called sub-no-signaling. Our main technical contribution (which is used in both our proofs) is a reduction showing how to convert any sub-no-signaling strategy with value at least $1 - 2^{-\Omega(k^2)}$ into a no-signaling one with value at least $2^{-O(k^2)}$.

In the first route, we show that the classical prover reduction method for converting k-prover games into 2-prover games carries over to the non-signaling setting with the following loss in soundness: if a k-player game has value less than 2^{-ck^2} (for some constant $c > 0$), then the corresponding 2-prover game has value at most $1 - 2^{dk^2}$ (for some constant $d > 0$). In the second route we show that the value of a sub-non-signaling game can be approximated in space that is polynomial in the communication complexity and exponential in the number of provers.

1.1 Pseudo-Deterministic Proofs

In [32], Gat and Goldwasser initiated the study of probabilistic (polynomial-time) search algorithms that, with high probability, output the same solution on different executions. That is, for all inputs x, the randomized algorithm A satisfies $Pr_{r_1,r_2}(A(x, r_1) = A(x, r_2)) \geq 1 - 1/poly(n)$.

Another way of viewing such algorithms is that for a fixed binary relation R, for every x the algorithm associates a canonical solution $s(x)$ satisfying $(x, s(x)) \in R$, and on input x the algorithm outputs $s(x)$ with overwhelmingly high probability. Algorithms that satisfy this condition are called *pseudo-deterministic*, because they essentially offer the same functionality as deterministic algorithms; that is, they produce a canonical output for each possible input (except with small error probability)[1]. In contrast, arbitrary probabilistic algorithms that solve search problems may output different solutions when presented with the same input (but using different internal coin tosses); that is, on input x, the output may arbitrarily distributed among all valid solutions for x (e.g. it may be uniformly distributed).

Several pseudo-deterministic algorithms have been found which improve (sometimes significantly) on the corresponding best known deterministic algorithm. This is the case for finding quadratic non-residues modulo primes, generators for certain cyclic groups, non-zeros of multi-variate polynomials, matchings in bipartite graphs in RNC, and sub-linear algorithms for several problems [36, 39, 32, 44]. For other problems, such as finding unique primes of a given length, pseudo-deterministic algo-

[1]In fact, by amplifying the success probability, one can ensure that as black boxes, pseudo-deterministic algorithms are indistinguishable from deterministic algorithms by a polynomial time machine.

rithms remain elusive (for the case of primes, it has been shown that there exists a subexponential time pseudo-deterministic algorithm which works on infinitely many input sizes [66]).

In this work we extend the study of pseudo-determinism in the context of probabilistic algorithms to the context of interactive proofs and non-determinism. We view pseudo-deterministic interactive proofs as a natural extension of pseudo-deterministic randomized polynomial time algorithms: the goal of the latter is to *find* canonical solutions to search problems whereas the goal of the former is to *prove* that a solution to a search problem is canonical to a probabilistic polynomial time verifier. Alternatively, one may think of the powerful but possibly untrusted prover as aiding the probabilistic polynomial time verifier to find canonical solutions to search problems. This naturally models the cryptographic setting when an authority generates system-wide parameters (e.g. an elliptic curve for all to use or a generator of a finite group) and it must prove that the parameters were chosen properly. Alternatively, one may think of the powerful prover as aiding the probabilistic polynomial time verifier to find canonical solutions to search problems, with high probability over the randomness of the verifier.

1.1.1 Our Contribution

Consider the search problem of finding a large clique in a graph. A nondeterministic efficient algorithm for this problem exists: simply guess a set of vertices C, confirm in polynomial time that the set of vertices forms a clique, and either output C or reject if C is not a clique. Interestingly, in addition to being nondeterministic, there is another feature of this algorithm; on the same input graph there may be many possible solutions to the search problem and any one of them may be produced as output. Namely, on different executions of the algorithm, on the same input graph G, one execution may guess clique C and another execution may guess clique $C' \neq C$, and both are valid accepting executions.

A natural question is whether for each graph with a large clique, there exists a unique canonical large clique C which can be verified by a polynomial time verifier:

that is, can the verifier V be convinced that the clique C is the canonical one for the input graph? We note that natural candidates which come to mind, such as being the lexicographically smallest large clique, are not known to be verifiable in polynomial time (but seem to require the power of Σ_2 computation). Indeed, the work of Hemaspaandra et al [45] implies the collapse of the polynomial time hierarchy if for every satisfiable SAT formula there exists a canonical assignment which is polynomial time verifiable.

In this work, we consider this question in the setting of interactive proofs, going beyond NP proofs to interactive proofs. Interactive proofs extend NP, and we ask whether the interactive proof mechanism enables provers to convince a probailistic verifier of the "uniqueness of their answer" (properly defined) with high probability.

We define *pseudo-deterministic interactive proofs* for a search problem R (consisting of pairs $(instance, solution)$) as a pair of interacting algorithms: a probabilistic polynomial time verifier and a computationally unbounded prover which on a common input instance x engage in rounds of interaction at the end of which with high probability the verifier output a canonical solution y for x if any solution exists and otherwise rejects x. Analogously to the case of interactive proofs for languages, we require that for every input x, there exists an honest prover which sends a correct solution y to the verifier when one exists and for all dishonest provers the probability that the verifier will accept an incorrect solution is small. Most importantly, we are interested in an additional feature: for every input x the verifier is guaranteed with high probability over its randomness to accept a canonical (unique) solution y if any solution exists and otherwise reject. Importantly, a dishonest prover may not cause the verifier to output a solution other than the canonical (unique) one (except with very low probability).

One may think of the powerful prover as aiding the probabilistic polynomial time verifier to find canonical solutions to search problems, with high probability over the randomness of the verifier. The challenge is that pseudo-determinism should hold not only with respect to the randomness, but also with respect to the prover: a malicious prover should not be able to cause the verifier to output a solution other than the canonical unique one. In addition to the intrinsic complexity theoretic interest in this problem, *consistency* or *predictability* of different executions on the same input are natural requirements from protocols.

We define *pseudo-deterministic IP* (psdIP) to be the class of search problems R (relation on inputs and solutions) for which there exists a probabilistic polynomial time verifier for which for every $x \in R_L$, there is a good powerful prover that will convince the verifier to output with high probability a unique witness $s(x)$ (referred to as the "canonical" witness) such that $(x, s(x)) \in R$; and for every x not in R_L (the set of x such that there does not exist a y satisfying $(x, y) \in R$), for all provers the verifier will reject with high probability. Furthermore, for all provers , the probability that on $x \in R_L$, the verifier will output any witness y other than the "canonical" $s(x)$ is small.

Important Remark: We remark that proving uniqueness of a witness for any NP problem can obviously be done in by an interactive proof with a PSPACE prover – the prover can convince the verifier that a witness provided is a lexicographically smallest witness – using an Arthur-Merlin proof which takes a *polynomial number* of rounds of interaction between prover and verifier following the celebrated Sum-Check protocol by [77] (see preliminary section). The interesting question to ask is: do **constant-round** pseudo-deterministic interactive proofs exist for hard problems in NP for which many witnesses exist?
We let psdAM refer to those pseudo-deterministic interactive proofs in which a constant number of rounds is used.

Our Results

Graph Isomorphism is in pseudo-deterministic AM: Theorem 3.1.1: *There exists a pseudo-deterministic constant-round Arthur-Merlin protocol for finding an isomorphism between two given graphs.*

Recall that the first protocol showing graph non-isomorphism is in constant round IP was shown by [37] and later shown to be possible using public coins via the general transformation of private to public coins [42]. Our algorithm finds a unique isomorphism by producing the lexicographically first isomorphism. In order to prove that a particular isomorphism between input graph pairs is lexicographically smallest, the prover will prove in a sequence of sub-protocols to the verifier that a sequence of graphs suitably defined are non-isomorphic. In an alternative construction, we exhibit an interactive protocol that computes the automorphism group of a graph in a verifiable fashion.

SAT is not in pseudo-deterministic AM: Theorem 3.2.2: *if any* NP-*complete problem has a a pseudo-deterministic constant round* AM *protocol, then,* NP $\subseteq$ coNP/*poly and the polynomial hierarchy collapses to the third level, showing that it is unlikely that NP complete problems have pseudo-deterministic constant round* AM *protocols.*

This result extends the work of [45] which shows that if there are polynomial time unique verifiable proofs for SAT, then the polynomial hierarchy collapses. Essentially, their result held for deterministic interactive proofs (i.e., NP), and we extend their result to probabilistic interactive proofs with constant number of rounds (i.e., AM).

Every problem in search-BPP is in subexponential-time pseudo-deterministic MA: Theorem 3.3.3: *For every problem in search-*BPP*, there exists a pseudo-deterministic* MA *protocol where the verifier takes subexponential time on infinitely many input lengths.*

The idea of the result is to use known circuit lower bounds to get pseudo-deterministic subexponential time MA protocols for problems in search-BPP for infinitely many input lengths. We remark that recently Oliveira and Santhanam [66] showed a subexponential time pseudo-deterministic algorithm for infinitely many input lengths for all properties which have inverse polynomial density and are testable in probabilistic polynomial time. (An example of such a property is the property of being prime, as the set of primes has polynomial density.) In their construction, the condition of high density is required in order for the property to intersect with their subexponential-size hitting set. (Subsequent work in [46] also drops this requirement but only results in an average-case pseudo-deterministic algorithm.) In the case of MA, unconditional circuit lower bounds for MA with a verifier which runs in exponential time have been shown by Miltersen et al [64], which allows us to no longer require inverse polynomial density. Hence, we can obtain a pseudo-deterministic MA algorithm from circuit lower bounds. Thus, compared to [66], our result shows a pseudo-derandomization (for a subexponential verifier and infinitely many input sizes n) for all problems in search-BPP (and not just those with high density), but requires a prover.

Pseudo-deterministic NL equals search-NL: Theorem 3.4.1: *For every search problem in search-*NL*, there exists a pseudo-deterministic* NL *protocol.*

We define *pseudo-deterministic NL* to be the class of search problems R (a relation on inputs and solutions) for which there exists log-space non-deterministic algorithm M (Turing machines) such that for every input x, there exists a unique $s(x)$ such that $R(x, s(x)) = 1$ and $M(x)$ outputs $s(x)$ or rejects x. Namely, there are no two accepting paths for input x that result in different outputs.

To prove the above theorem, we look at the problem of directed connectivity (that is, given a directed graph G with two vertices s and t, we find a path from s to t), and we show that it is possible to find the lexicographically first path of shortest length in NL. To do so, we first find the length d of the shortest path, which can be done in NL. Then, we find the lexicographically first outneighbor u of s such that there is a path of length $d-1$ from u to t. This can be done by going in order over all outneighbors of s, and for each of them checking if there is a path of length $d-1$ to t (if there is not such a path, that can be demonstrated in NL since NL $=$ coNL [50, 78]). By recursively applying this protocol to find a path from u to t, we end up obtaining the lexicographically first path of shortest length, which is unique.

Structural Results: We show a few structural results regarding pseudo-deterministic interactive proofs In Section 3.5. Specifically, we show that psdAM equals to the class search$-\mathrm{P}^{\mathrm{promise}-(\mathrm{AM}\cap\mathrm{coAM})}$, where for valid inputs x, all queries to the oracle must be in the promise. We show similar results in the case of pseudo-deterministic MA and pseudo-deterministic NP.

1.1.2 Other Related Work

In their seminal paper on NP with unique solutions, Valiant and Vazirani asked the following question: is the inherent intractability of NP-complete problems caused by the fact that NP-complete problems have many solutions? They show this is not the case by exhibiting a problem – SAT with unique solutions – which is NP-hard under randomized reductions. They then showed how their result enables to show the NP-hardness under randomized reductions for a few related problems such as parity-SAT. We point out that our question is different. We are not restricting our study to problems (e.g. satisfiable formulas) with unique solutions. Rather, we consider hard problems for which there may be exponentially many solutions, and ask if one can focus on one of them and verify it in polynomial time. In the language of satisfiability, ϕ can be any satisfiable formula with exponentially many satisfying assignments; set $s(\phi)$ to be a unique valued function which outputs a satisfying assignment for ϕ. We study whether there exists an s which can be efficiently computed, or which has an efficient interactive proof.

The question of computing canonical labellings of graphs was considered by Babai and Luks [6] in the early eighties. Clearly graph isomorphism is polynomial time reducible to computing canonical labellings of graphs (compute the canonical labeling for your graphs and compare), however it is unknown whether the two problems are equivalent (although finding canonical labellings in polynomial time seems to be known for all classes of graphs for which isomorphism can be computed in polynomial time). The problem of computing a set of generators (of size $O(\log n)$) of the automorphism group of a graph G was shown by Mathon [62] (among other results) to be polynomial-time reducible to the problem of computing the isomorphism of a graph. We use this in our proof that graph isomorphism is in psdAM.

Finally, we mention that recently another notion of uniqueness has been studied in the context of interactive proofs by Reingold et al [71], called *unambiguous interactive proofs* where the prover has a unique successful strategy. This again differs from pseudo-deterministic interactive proofs, in that we don't assume (nor guarantee) a unique strategy by the successful prover, we only require that the prover proves that the solution (or witness) the verifier receives is unique (with high probability).

1.1.3 Subsequent Work

In [46], inspired by this work, Holden shows that for every BPP search problem there exists an algorithm A which for infinitely many input lengths n and for every polynomial-time samplable distribution over inputs of length n runs in subexponential time and produces a unique answer with high probability on inputs drawn from the distribution and over A's random coins.

[46] expands on the work of Oliveira and Santhanam [66] in several ways. Whereas the latter give a pseudo-deterministic algorithm for estimating the acceptance probability of a circuit on inputs of a given length, the former applies to general search-BPP problems, where the input is a string of a given length over some alphabet and algorithm's A goal is to output a solution that satisfies a BPP testable relation with the input string. Holden [46] shows that for infinitely many input lengths, average-case (over the input distribution) pseudo-deterministic algorithms are possible for problems in search-BPP.

1.2 Doubly-Efficient Pseudo-Deterministic Proofs

Pseudo-deterministic algorithms, introduced by Gat and Goldwasser [32], are probabilistic (polynomial-time) algorithms for search problems that, with high probability, *find* a unique output for each input except with negligible error probability. Such output for input x is referred to as the ”canonical” output for x. Algorithms that satisfy the aforementioned condition are of importance whenever uniqueness or "reproducibility" of the answer is important. This is of particular relevance in a distributed or parallel setting when an algorithm is executed by multiple parties for whom it is challenging (for reasons of trust or efficiency requirement) to *agree* on a common sequence of unbiased random coins.

More recently, Goldwasser, Grossman and Holden [40] extended the study to pseudo-deterministic interactive proofs for search problems, denoted psdIP. The new goal was to *prove* to a probabilistic polynomial time verifier that a solution to a search problem is canonical. The motivation was to address those search problems for which polynomial time algorithms are not known and for which many solutions are possible, such as for graph isomorphism. In this case the *search problem* is to find an isomorphism between two graphs if one exists and an example of a *canonical solution* would be the lexicographically smallest isomorphism. One may think of the powerful prover as aiding the probabilistic polynomial time verifier to find canonical solutions to search problems, with high probability over the randomness of the verifier. The challenge is that a malicious prover should not be able to convince the verifier to accept any solution other than the unique canonical one and that the interaction should be constant round. If unbounded number of rounds are allowed, the $IP = PSPACE$ characterization implies that psdIP $= IP$.

In this work, we turn our attention to studying *doubly-efficient pseudo-deterministic proofs*. That is pseudo-deterministic proofs with the extra requirement that the prover is efficient as well. Our aim is to show doubly-efficient pseudo-deterministic proofs for polynomial time problems, where the prover runs in polynomial time in the complexity of the problem and the verifier can *verify* that a solution is canonical significantly more efficiently than solving the problem without the presence of the prover. We remark that in the doubly-efficient pseudo-deterministic proofs below, except for linear programming, the runtime of the prover is at most a constant times the runtime of the best known deterministic algorithm.

1.2.1 Our Results

A new notion: Doubly-efficient pseudo-deterministic interactive proofs

We define *doubly-efficient pseudo-deterministic interactive proofs* for a search problem R of complexity $T(n)$ (consisting of pairs $(instance, solution)$) with associated canonization function c as a pair of interacting algorithms: a probabilistic polynomial time prover which runs in time $poly(T(n))$ and a probabilistic verifier which runs in time $o(T(n))$ which on a common input instance x engage in constant number of rounds of interaction at the end of which with high probability the verifier outputs a canonical solution $y = c(x)$ if any solution exists and otherwise rejects x. Analogously to the case of completeness in interactive proofs for languages, we require that for every input x, there exists an honest prover which can send the correct solution $c(x)$ to the verifier when one exists. Analogously to the case of soundness, no dishonest prover can cause the verifier to output a solution other than $c(x)$ (the canonical one) (except with very low probability).

A few remarks are in order.

- Naturally this question is particularly interesting for search problems for which a lower bound on its worst case complexity $T(n)$ is known or has been widely conjectured. This will drive our choice of problems for which we show doubly efficient pseudo-deterministic proofs.

- Doubly-efficient pseudo-deterministic proofs for search problems R with associated canonization function c are closely related to **computation delegation** of computing $c(x)$ on input x. The delegation problem was posed by Goldwasser, Kalai, and Rothblum [41] and become known under the name doubly-efficient interactive proof systems. The difference in the requirements is that [41] allow the prover to be any polynomial time algorithm and the verifier to run in linear (up to log factors) time and addresses deterministic computations. Doubly-efficient interactive proofs have been shown by [41] for log-space uniform sets in NC (or, more generally, to inputs that are acceptable by log-space uniform bounded-depth circuits, where the number of rounds in the proof system is linearly related to the depth of the circuit). Reingold, Rothblum and Rothblum [71] showed that any set decidable in polynomial-time by an algorithm of space complexity $s(n) \leq n^{0.499}$, has a constant-round interactive proof system in which the prover runs in polynomial time and the verifier runs in time $\tilde{O}(n)$. Finally Goldreich and Rothblum [38] show direct constructions of doubly-efficient interactive proof systems for problems in P that are believed to have relatively high complexity such as t-CLIQUE and t-SUM.

We remark that works on proof systems and delegation did not stay within the realm of theory alone. Rather, they became the theoretical basis for several system implementations of a delegation system as they offered reasonably efficiently realizable protocols. Indeed, there is a flourishing literature surrounding the refinement and implementation of these theoretical protocols [7, 8, 72, 11, 12, 16, 22, 25, 26, 30, 59, 67, 74, 76, 75, 79, 80, 83, 84] (see [85] for a survey). We hope that our proposed study of doubly-efficient pseudo-deterministic proofs can similarly impact practice (and beyond).

- The setting of doubly-efficient interactive proofs naturally models a cryptographic setting where users wish to have access to common cryptographic system-wide keys or parameters, such as a pair (g, p) for Z_p with prime p and generator g for a given input length n. A central authority (with additional computational power) can of course choose the common system-wide parameter and broadcast it to all, but then who is to say that the central party did not chose its randomness in a way that would force an output for which the trusted center knew some "trapdoor" information which would enable it to break the underlying cryptographic security? Viewing the generation of a cryptographic key as a solution to a search problem R per security parameter, a doubly-efficient pseudo-deterministic proof for R would ensure that the prover had no choice in which parameter to broadcast as he could prove that his solution is canonical.

Doubly-efficient pseudo-deterministic algorithms for linear programming and fine-grained complexity problems

Linear Programming: We show a doubly-efficient pseudo-deterministic proof for the linear programming problem. Verifying an optimal solution to a linear programming problem can be done thanks to *strong duality*: there exists a solution to the dual problem with the same value as the solution to the primal problem. We show that a special optimal solution, namely the lexicographically greatest solution, can be efficiently obtained by the prover, and that the prover can convince the verifier that the LP solution it gives to the verifier is indeed the lexicographically greatest solution; this is done through perturbing the linear program and strong duality. More concretely, every linear program (say, where the objective is to maximize) has a corresponding dual linear program, a minimization problem, with the property that (i) (weak duality) any feasible solution to the dual provides an upper bound on the optimal primal value and (ii) (strong duality) there exists an optimal solution to the dual with the same value as the primal optimal solution. Furthermore, there exist compact polynomial-sized solutions to the primal and dual linear programs. Therefore such a polynomial-sized feasible solution to the dual with an equal value as a primal solution provides a compact certificate for the optimality of this primal solution.

The currently best known time to solve a linear program with integer data and at most d variables and constraints is $\tilde{O}(d^{\omega})$ randomized [24] where ω corresponds to the exponent for fast matrix multiplication which is currently at ≈ 2.37 and $\tilde{O}()$ hides polylog factors including a $\log(1/\delta)$ factor to account for the accuracy δ in solving the linear program. The time of the verifier to verify a pair of primal and dual optimal solution is only $O(d^2)$ as this only requires matrix-vector multiplication.

Problems studied in fine-grained complexity: We next show doubly-efficient pseudo-deterministic proofs for several fine-grained complexity problems where the verifier significantly beats the conjectured time. The challenge is to find a proof where the prover's running time does not change too much from the running time of the deterministic algorithm. In the case of two of our problems, making the running time close to the running time of the deterministic algorithm requires the prover to run in a randomized fashion.

The **3-SUM problem** has an easy $O(n^2)$ time algorithm which can be improved by polylogarithmic factors. It is an outstanding open question whether there is an algorithm that significantly improves $O(n^2)$. Finding such an algorithm would yield algorithms for a host of other problems in computational geometry [31, 27]. Here, we show a doubly-efficient pseudo-deterministic proofs that outputs the lexicographically first such triple of elements where the verifier takes time $\tilde{O}(n^{1.5})$. We crucially use the fact that [20] gives a nondeterministic proof that there is no triple of elements that sum to 0 where the verifier takes time $\tilde{O}(n^{1.5})$.

The **hitting set problem** is the problem of finding a set in a collection of sets that intersects every set in a different collection of sets. We show a pseudo-deterministic proof for the hitting set problem where the verifier runs in time $\tilde{O}(m)$ and the prover runs in time $\tilde{O}(m^2)$ where $m = \sum_{S\in\mathcal{S}} |S| + \sum_{T\in\mathcal{T}} |T|$ for inputs $\mathcal{S}, \mathcal{T}$ collections of sets. This problem has been conjectured to take $m^{2-o(1)}$ time [82].

The **All-Pairs Shortest Path** problem is a well-studied problem that is the foundation of many hardness results in graph algorithms [88, 87]. In particular, the **Zero Weight Triangle** problem is equivalent to the All-Pairs Shortest Path problem under subcubic reductions. We show a doubly efficient pseudo-deterministic proof for the Zero Weight Triangle problem where the verifier runs in time $\tilde{O}(n^{2+\omega/3})$ and the prover runs in randomized time $\tilde{O}(n^3)$.

Techniques

All our results take on the following flavor: For a search problem R, the pseudo-deterministic algorithm, given x, finds the lexicographically first y such that $R(x, y)$.

To do this, it asks whether there exists y' such that $(x, 0y') \in R$, y' such that $(x, 1y') \in R$, etc. and finds the first y such that $R(x, y)$ recursively. The notion of "lexicographically first" can be easily generalized to allow other orderings and other encodings of the input. This suggests that more generally doubly-efficient pseudo-deterministic proofs for search are the ones where there is a doubly-efficient proof of existence and a doubly-efficient proof of nonexistence of solutions to said search problem. A more general theorem (Lemma 4.1.3) follows under general conditions.

1.3 Non-Signaling

Shortly after interactive proofs were introduced, multi-party interactive proofs were introduced by Ben-Or, Goldwasser, Kilian and Wigderson [10]. In a multi-prover interactive proof (MIP) a verifier is interacting with several non-communicating provers. This class was proven to be extremely powerful, by Babai, Fortnow and Lund, who showed that MIP = NEXP [5]. The power of this class stems from the assumption that the provers behave locally, namely, they see only the messages sent to them, and do not have any information about messages sent to the other provers.

In reality, however, it is not clear how to ensure that the provers behave locally. Even if the provers are placed in different rooms, with no communication channels between them, they may share quantum entanglement, which can cause their strategies to be correlated, and non-local. These attacks can be powerful, even though at first they may seem to be benign [23].

These quantum strategies motivated the notion of no-signaling strategies, which is the subject of this work. The notion of no-signaling strategies was first studied in physics in the context of Bell inequalities by Khalfin and Tsirelson [58] and Rastall [70], and it has gained much attention after it was reintroduced by Popescu and Rohrlich [69]. No-signaling attacks are more general than quantum attacks. In a no-signaling attack the cheating provers can collude, and thus each answer can be a function of *all* the queries. The only restriction is that for any subset of provers the answers provided by these provers should not convey any information about the queries given to the other provers. Namely, the only restriction that is placed on the (possibly colluding) cheating provers, is that their answers cannot be seen as “evidence" that information has travelled between them.

If we think of the provers as being placed very far away from each other (say in different planets), then the no-signaling restriction allows the provers to behave arbitrarily, as long as they adhere to the physical principle that information cannot travel faster than light, a consequence of Einstein’s special relativity theory. In particular, all the strategies that can be realized by provers that share entangled quantum states are no-signaling strategies. Moreover, the principle that information cannot travel faster than light is a central principle in physics, and is likely to remain valid in any future ultimate theory of nature, since its violation means that information could be sent from future to past. Therefore, soundness against no-signaling strategies is likely to ensure soundness against provers that obey a future ultimate theory of physics, and not only the current physical theories that we have, that are known to be incomplete.

Importantly, although no-signaling strategies are motivated by quantum entanglement, they found compelling applications outside the realm of quantum physics. In particular, they have been proved to be instrumental for constructing succinct delegation schemes (under standard cryptographic assumptions), and in the realm of hardness of approximation.

The applicability of no-signaling to computation delegation. Kalai, Raz, and Rothblum [56] demonstrated the significance of no-signaling by showing that any MIP that is secure against no-signaling attacks[2] can be converted into a single-prover one-round proof system (with computational soundness). More specifically, they show that the PIR (or FHE) heuristic, proposed by Biehl, Meyer, and Wetzel *et. al.* [13], for converting any MIP to a *single prover* one-round proof system is sound if the underlying MIP has no-signaling soundness.

In [57], the same authors constructed an MIP that is secure against no-signaling attacks for every language in EXP, thus yielding the first one-round delegation scheme for *all* deterministic computations, under standard cryptographic assumptions. This application of no-signaling to computation delegation has proved to be very fruitful, and yielded numerous followup works (e.g., [54, 15, 9])

The applicability of no-signaling to hardness of approximation. Kalai, Raz and Regev [55] showed the significance of no-signaling to hardness of approximation. In particular, they showed that it is hard to approximate the value of a linear program in space $2^{\log n^{o(1)}}$, even if the polytope is *fixed* (i.e., even if the algorithm has unbounded

[2]To be precise, [56] considered a slightly more relaxed notion, which they called statistical no-signaling. We neglect this difference here.

time to preprocess the polytope), and even if all the coefficients are non-negative (which is the regime where hardness of approximation is most meaningful). More specifically, they showed that there exists a fixed polytope (corresponding to the set of all possible non-signaling strategies) such that approximating the value of a linear program (with positive coefficients) is P-complete with a polylog-space reduction. Prior work [28, 73, 29, 61], demonstrated such hardness of approximation for the case where the polytope was not fixed (and preprocessing is not allowed).
The importance of the notion of no-signaling, gives rise to the following fundamental question:

What is the power of multi-prover proofs that are sound against no-signaling strategies?

This is precisely the question we study in this work. In what follows, we denote the class of one-round multi-prover interactive proofs with no-signaling soundness by NS MIP. We denote by k-prover NS MIP the class of one-round k-prover interactive proofs with no-signaling soundness.

1.3.1 Prior Work

Ito, Kobayashi and Matsumoto [53] proved that 2-prover NS MIP contains PSPACE (by proving that the 2-prover scheme of Cai, Condon, and Lipton [18] is in fact secure against no-signaling strategies). Shortly after, Ito [51] proved that 2-prover NS MIP is contained in PSPACE, thus characterizing the power of 2-prover NS MIP. The power of k-prover NS MIP, for $k \geq 2$, remained open.

It is known that NS MIP is contained in EXP since one can find the best no-signaling strategy by solving an exponential-size linear program. Therefore, the power of a k-prover NS MIP lies between PSPACE and EXP. More recently, Kalai, Raz and Rothblum [57] showed that there exists a constant $c \in \mathbb{N}$ such that for every $k \geq \log^c n$, k-prover NS MIP contains EXP, thus characterizing the power of k-prover NS MIP for $k \geq \log^c n$.

These works left open the following question: What is the power of k-prover NS

MIP for $2 < k < \log^c n$?

1.3.2 Our Results

Throughout this manuscript, we assume that an MIP has completeness at least $1 - \text{negl}(n)$, and has soundness $\text{negl}(n)$, for some negligible function $\text{negl}(n)$.[3] This assumption is standard in cryptography. We mention that often in the definition of interactive proofs (or possibly in the definition of MIPs), completeness is required to be greater than $2/3$ and soundness at most $1/3$; this is because it is well known that this gap can be amplified to $1 - \text{negl}(n)$ and $\text{negl}(n)$ via parallel repetition (where the verifier accepts if $2/3 - \epsilon$ of the repetitions are accepted, for some small constant $\epsilon > 0$). In the no-signaling regime we do not have a parallel repetition theorem [49], and hence cannot amplify soundness via parallel repetition.

We prove that k-prover NS MIP with $k = O(\sqrt{\log n})$ is contained in PSPACE. More generally, we prove the following theorem.

Theorem 1.3.1 (Informal). *There exist constants $c, d > 0$ such that any k-prover* MIP *with no-signaling soundness at most 2^{-ck^2} and completeness at least $1 - 2^{-dk^2}$, is contained in* SPACE $\left(poly(n, 2^{k^2})\right)$.

We emphasize that this theorem holds only for MIPs that have negligible soundness and almost perfect completeness. In particular, we don't rule out the existence of a k-prover MIP with NS soundness $1/3$ and completeness $2/3$ for EXP, with $k = \log n$. However, the soundness and completeness gap of such MIPs could not be amplified (to $1 - \text{negl}(n)$) without adding provers.

We present two alternative routes for proving Theorem 1.3.1, each is of independent interest. Both routes consider the more relaxed notion of *sub-no-signaling*, as defined in [60] (for the goal of obtaining a parallel repetition theorem for no-signaling strategies). Both rely on the following theorem that asserts that one can convert any sub-no-signaling strategy into a no-signaling one, albeit with a substantial loss in the success probability.

[3]A function $\mu : \mathbb{N} \to \mathbb{N}$ is said to be negligible if approaches zero faster than the inverse of any polynomial.

Theorem 1.3.2 (Informal)**.** *There exist constants $c, d > 0$ such that for any k-prover* MIP*, if there exists a sub-no-signaling strategy that succeeds with probability at least $1 - 2^{-dk^2}$, then there exists a no-signaling strategy that succeeds with probability at least 2^{-ck^2}.*

The proof of this theorem contains the bulk of technical difficulty of this work, and is used as a building block in both proofs of Theorem 1.3.1. See Section **??** for the proof idea, and see Section 5.2 for the precise theorem statement and proof.

We note that a related theorem was proven by Lancien and Winter [60], who showed that, for every game, if there exists a sub-no-signaling strategy that succeeds with probability at least $1 - \epsilon$ then there exists a no-signaling strategy that succeeds with probability at least $1 - \Gamma\epsilon$, where Γ may be as large as exponential in the communication complexity. This bound does not seem to be tight enough in order to obtain Theorem 1.3.1.

We next present our two alternative routes for proving Theorem 1.3.1 (using Theorem 1.3.2). The first is via a prover reduction method, and the second is via approximating the sub-no-signaling value efficiently.

Reducing the number of provers. We show that (a slight variant of) the classical prover reduction method for converting a k-prover MIP into a 2-prover MIP, carries over to the no-signaling setting, albeit a substantial loss in soundness (which depends on k).

More specifically, in the seminal work of Ben-Or, Goldwasser, Kilian and Wigderson [10], they presented a general method for converting a k-prover MIP into a 2-prover MIP, where in the resulting 2-prover MIP the verifier sends one prover the queries $(q_1, \ldots, q_k)$ corresponding to *all* the k provers in the underlying k-prover scheme, and expects to get back k answers $(a_1, \ldots, a_k)$; he sends the other prover a single query q_i corresponding to a random index $i \in [k]$, and gets back an answer a'_i. The verifier accepts if and only if $a'_i = a_i$ and if the verifier in the k-prover MIP accepts the answers $(a_1, \ldots, a_k)$.

In the no-signaling setting, we slightly modify this transformation, by having the verifier in the 2-prover MIP send the second prover a *subset* of queries $\{q_i\}_{i\in S}$ for a randomly chosen subset $S \subset [k]$ (as opposed to a single query q_i corresponding to a single index $i \in [k]$), and accepts if and only if the answers $(a_1, \ldots, a_k)$ of the first prover are accepted by the verifier in the k-prover MIP and if the answers of the second prover, denote by $(a'_i)_{i\in S}$ satisfy that $a'_i = a_i$ for every $i \in S$.

Theorem 1.3.3 (Informal)**.** *There exist constants $c, d > 0$ such that for every k-prover* MIP $\Pi = (P_1, \ldots, P_k, V)$ *with no-signaling soundness at most 2^{-ck^2}, the 2-prover* MIP*, obtained by performing the prover reduction transformation (described above) on Π, has no-signaling soundness at most $1 - 2^{-dk^2}$.*

We prove Theorem 1.3.3 by using Theorem 1.3.2. We refer the reader to Section **??** for the proof idea, and Section 5.1.1 for the precise theorem statement and proof.

We next argue that Theorem 1.3.3 implies Theorem 1.3.1. Let c, d be the constants from Theorem 1.3.3. We prove Theorem 1.3.1 with constants $c' = c$ and $d' = 2d$. To this end, fix any k-prover MIP for a language L with no signaling soundness 2^{-ck^2} and completeness $1 - 2^{-2dk^2}$. Use Theorem 1.3.3 to convert this MIP into a 2-prover MIP with no-signaling soundness $1 - 2^{-dk^2}$. By [51], the no-signaling value of any 2-player game can be approximated up to an additive factor of ϵ in space $\text{poly}(n, 1/\epsilon)$. Setting $\epsilon = 2^{-2dk^2}$, there exists an algorithm $\mathcal{A}$ that runs in space $\text{poly}(n, 2^{k^2})$, such that on input an element $x \in \{0,1\}^n \cap L$ it outputs a value $v \geq 1 - 2 \cdot 2^{-2dk^2}$, and on input an element $x \in \{0,1\}^n \setminus L$ it outputs a value $v \leq 1 - 2^{-dk^2} + 2^{-2dk^2}$. This algorithm can be used to decide whether $x \in L$ (assuming without loss of generality that $d > \frac{2}{k^2}$), implying that $L \in \text{SPACE}(\text{poly}(n, 2^{k^2}))$.

Approximating the sub-no-signaling value. We next present an alternative route for proving Theorem 1.3.1, without going through the prover reduction method presented above. Instead we prove the following theorem, which is of independent interest.

Theorem 1.3.4 (Informal)**.** *The sub-no-signaling value of any k-prover* MIP *can be approximated up to an additive factor ϵ by a poly$(n, 2^k, 1/\epsilon)$-space algorithm.*

In particular this theorem implies the following corollary.

Corollary 1.3.5 (Informal). *k-prover* subNS MIP *is contained in* SPACE $\big(poly(n, 2^k)\big)$.

We mention that a related (yet weaker) theorem was proven in [48], where it was shown that given an MIP, one can distinguish between the case that its value is 1 (i.e., there exists a local strategy that is accepted with probability 1) and the case that its sub-no-signaling value is at most $1 - \delta$, in space $\text{poly}(n, 2^k, 1/\delta)$. This does not seem to be strong enough for us to use in order to obtain Theorem 1.3.1.

We next argue that Theorem 1.3.4 and Theorem 1.3.2 imply Theorem 1.3.1. To this end, let $c, d > 0$ be the constants from Theorem 1.3.2. We prove Theorem 1.3.1 with any constants c', d' such that $c' > c$ and $d' = 2d$. Fix any k-prover MIP with soundness at most $2^{-c'k^2} < 2^{-ck^2}$ and completeness at least $1 - 2^{2dk^2}$. By Theorem 1.3.2 for every $x \in \{0,1\}^n \setminus L$ the sub-no-signaling value the MIP on input x must be less than $1 - 2^{-dk^2}$. By Theorem 1.3.4, applied with $\epsilon = 2^{-2dk^2}$, there exists an algorithm $\mathcal{A}$, that given any $x \in \{0,1\}^n$, runs in space $\text{poly}(n, 2^{k^2})$ and approximates the sub-no-signaling value of this MIP on input x up to an additive factor 2^{-2dk^2}. Therefore for every $x \in \{0,1\}^n \setminus L$, the algorithm $\mathcal{A}(x)$ outputs an element $v \leq 1 - 2^{-dk^2} + 2^{-2dk^2}$, and for every $x \in \{0,1\}^n \cap L$ the algorithm $\mathcal{A}(x)$ outputs an element $v \geq 1 - 2 \cdot 2^{-2dk^2}$. This algorithm can be used to decide whether $x \in L$ (assuming without loss of generality that $d > \frac{2}{k^2}$), implying that $L \in \text{SPACE}(\text{poly}(n, 2^{k^2}))$.

Chapter 2

Preliminaries

Here are the definitions that will be used throughout this thesis.

2.1 Pseudo-determinism

2.1.1 Definitions of Pseudo-determinism

In this section, we define pseudo-determinism in the context of nondeterminism and interactive proofs. We begin by defining a search problem. Intuitively speaking, a search problem is a problem where for an input, there may be multiple possible outputs.

Definition 2.1.1 (Search Problem)**.** A *search problem* is a relation R consisting of pairs (x, y). We define L_R to be the set of x's such that there exists a y satisfying $(x, y) \in R$. An algorithm solving the search problem is an algorithm that, when given $x \in L_R$, finds a y such that $(x, y) \in R$. When L_R contains all strings, we say that R is a *total* search problem. Otherwise, we say R is a *promise* search problem.

We now define pseudo-determinism in the context of interactive proofs. Intuitively speaking, we say that an interactive proof is pseudo-deterministic if an honest prover causes the verifier to output the same unique solution with high probability (canonical completeness), and dishonest provers can only cause the verifier to output either the unique solution or $\perp$ with high probability (canonical soundness). In other words, dishonest provers cannot cause the verifier to output an answer which is not the unique answer. Additionally, we have the condition that for an input x with no solutions, for all provers the verifier will output $\perp$ with high probability (standard soundness) We note that we use psdIP, psdAM, psdNP, psdMA, and so on, to refer to a class of promise problems, unless otherwise stated.

Definition 2.1.2 (Pseudo-deterministic IP)**.** A search problem R is in *pseudo-deterministic* IP (often denoted psdIP) if there exists a function s where all $x \in L_R$ satisfy $(x, s(x)) \in R$, and an interactive protocol between a probabilistic polynomial time verifier algorithm V and a prover (unbounded algorithm) P such that for every $x \in L_R$:

1. (Canonical Completeness) There exists a P such that $\Pr_r[(P, V)(x, r) = s(x)] \geq \frac{2}{3}$. (We use $(P, V)(x, r)$ to denote the output of the verifier V when interacting with prover P on input x using randomness r).
2. (Canonical Soundness) For all P', $\Pr_r[(P', V)(x, r) = s(x) \text{ or } \perp] \geq \frac{2}{3}$.

And (Standard Soundness) for every $x \notin L_R$, for all provers P', $\Pr_r[(P', V)(x, r) \neq \perp] \leq \frac{1}{3}$.

One can similarly define pseudo-deterministic MA, and pseudo-deterministic AM, where MA is a 1-round protocol, and AM is a 2-round protocol. One can show that any constant-round interactive protocol can be reduced to a 2-round interactive protocol [3]. Hence, the definition of pseudo-deterministic AM captures the set of all search problems solvable in a constant number of rounds of interaction. Furthermore, in the definition of pseudo-deterministic AM, we use public coins. One can show that any protocol using private coins can be simulated using public coins[1][42].

Definition 2.1.3 (Pseudo-deterministic AM)**.** A search problem R is in *pseudo-deterministic* AM (often denoted psdAM) if there exists a function s where all $x \in L_R$ satisfy $(x, s(x)) \in R$, a probabilistic polynomial time verifier algorithm V, and polynomials p and q, such that for every $x \in L_R$:

[1]In the case where the prover is not unbounded, private coins may be more powerful than public coins.

1. $\Pr_{r\in\{0,1\}^{p(n)}}(\exists z \in \{0,1\}^{q(n)}\, V(x,r,z) = s(x)) \geq \frac{2}{3}$
2. $\Pr_{r\in\{0,1\}^{p(n)}}(\forall z \in \{0,1\}^{q(n)}\, V(x,r,z) \in \{s(x),\perp\}) \geq \frac{2}{3}$.

And for every $x \notin L_R$, we have $\Pr_{r\in\{0,1\}^{p(n)}}(\forall z \in \{0,1\}^{q(n)}\, V(x,r,z) = \{\perp\}) \geq \frac{2}{3}$.

Definition 2.1.4 (Pseudo-deterministic MA)**.** A search problem R is in *pseudo-deterministic* MA (often denoted psdMA) if there exists a function s where all $x \in L_R$ satisfy $(x, s(x)) \in R$ and $|s(x)| \leq \mathit{poly}(x)$, a probabilistic polynomial time verifier V such that for every $x \in L_R$[2]:

1. There exists a message M of polynomial size such that $\Pr_r[V(x,M,r) = s(x)] \geq \frac{2}{3}$.
2. For all M', $\Pr_r[V(x,M',r) = s(x) \text{ or } \perp] > \frac{2}{3}$.

And for every $x \notin L_R$, for all M', $\Pr_r[V(x,M',r) \neq \perp] \leq \frac{1}{3}$.

Pseudo-determinism can similarly be defined in the context of nondeterminism (which can be viewed as a specific case of an interactive proof):

Definition 2.1.5 (Pseudo-deterministic NP)**.** A search problem R is in *pseudo-deterministic* NP (often denoted psdNP) if there exists a function s where all $x \in L_R$ satisfy $(x, s(x)) \in R$ and $|s(x)| \leq \mathit{poly}(x)$, and there is a deterministic polynomial time verifier V such that for every $x \in L_R$:

1. There exists a message M of polynomial size such that $V(x,M) = s(x)$.
2. For all M', $V(x,M') = s(x)$ or $V(x,M') = \perp$.

And for every $x \notin L_R$, for all M', we have $V(x,M') = \perp$.

A similar definition for pseudo-deterministic NL follows naturally:

Definition 2.1.6 (Pseudo-deterministic NL)**.** A search problem R is in *pseudo-deterministic* NL (often denoted psdNL) if there exists a function s where all $x \in L_R$ satisfy $(x, s(x)) \in R$ and $|s(x)| \leq \mathit{poly}(x)$, there is a nondeterministic log-space machine V such that for every $x \in L_R$:

[2] We remark that we use M to denote the proof sent by the prover Merlin, and not the algorithm implemented by the prover.

1. There exist nondeterministic choices N for the machine such that such that $V(x, N) = s(x)$.

2. For all possible nondeterministic choices N', $V(x, N') = s(x)$ or $V(x, N') = \perp$.

And for every $x \notin L_R$, for all nondeterministic choices N', $V(x, N') = \perp$.

It is valuable to contrast the above definition with the definition of search-NL, in which it is possible to have different nondeterministic guesses of the machine result in different answers:

Definition 2.1.7 (search-NL)**.** A search problem R is in *search*-NL if there is a nondeterministic log-space machine V such that for every $x \in L_R$,

1. There exist nondeterministic choices N for the machine such that such that $V(x, N) = y$, and $(x, y) \in R$.

2. For all possible nondeterministic choices N', $(x, V(x, N')) \in R$, or $V(x, N') = \perp$.

And for every $x \notin L_R$, for all nondeterministic choices N', $V(x, N') = \perp$.

Intuitively speaking, in the case of search-NL, it is okay for the algorithm to output different correct solutions when using different nondeterministic choices. In the case of pseudo-deterministic-NL, there should not be two nondeterministic choices for the algorithm which result in different answers (on input x, every set of nondeterministic choices which leads to an answer which is not $\perp$ should lead to the same answer $s(x)$).

We will also use the following definition of search-BPP, the class of search problem solvable (and verifiable) in probabilistic polynomial time:

Definition 2.1.8 (Search-BPP)**.** A binary relation R is in *search-BPP* if there exist probabilistic polynomial-time algorithms A, B such that

1. Given $x \in R_L$, A outputs a y such that with probability at least $2/3$, $(x, y) \in R$.

2. If y is output by A when run on x, and $(x, y) \notin R$, then B rejects on (x, y) with probability at least $2/3$. Furthermore, for all $x \in L_R$, with probability at least $1/2$ B accepts on (x, y) with probability at least $1/2$.

When $x \notin R_L$, A outputs $\perp$ with probability at least 2/3.

The intuition of the above definition is that A is used to find an output y, and then B can be used to verify y, and amplify the success probability.

2.1.2 Polynomial-round pseudo-determinism

Consider what happens in the polynomial-round case. From [77], we know that IP = PSPACE. It is also known that finding the lexicographically first witness to any NP problem is in PSPACE. Thus, we have the following lemma:

Lemma 2.1.9. *Every language $L \in \mathrm{NP}$ has an polynomial-round* psdIP *protocol.*

Proof. Let us consider the function $f(x)$ which outputs the lexicographically first witness that $x \in L$ if $x \in L$ or $\perp$ otherwise. It is easy to see that determining whether $f(x) = y$ is in PSPACE. As a result, there is an polynomial-round IP protocol to determine whether $f(x) = y$. Then, the psdIP protocol is as follows; the prover gives the verifier y and then they run the protocol, and the verifier accepts and outputs y if the protocol accepts. This satisfies the conditions for pseudo-determinism because of the completeness and soundness properties of the IP protocol. □

Definition 2.1.10. A linear program is, given a matrix A and vectors $\mathbf{b}, \mathbf{c}$, the problem $\max\{\mathbf{c}^\top \mathbf{x}\}$ subject to the constraints $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$. Its dual is the linear program $\min\{\mathbf{b}^\top \mathbf{y}\}$ subject to the constraints $A^\top \mathbf{y} \geq \mathbf{c}$ and $\mathbf{y} \geq \mathbf{0}$.

Theorem 2.1.11. *(Weak duality) If $\mathbf{x}, \mathbf{y}$ are feasible solutions to a linear program given by $\max\{\mathbf{c}^\top \mathbf{x}\}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ and its dual respectively, then $\mathbf{c}^\top \mathbf{x} \leq \mathbf{b}^\top \mathbf{y}$. (Strong duality) Furthermore $\mathbf{x}, \mathbf{y}$ are optimal solutions if and only if $\mathbf{c}^\top \mathbf{x} = \mathbf{b}^\top \mathbf{y}$.*

2.2 Doubly-Efficient Pseudo-Deterministic Proofs

In this section we will introduce concepts needed to give pseudo-deterministic proofs that improve on the best known deterministic algorithms for problems studied in fine-grained complexity.

Definition 2.2.1 (Search Problem)**.** A *search problem* is a relation R consisting of pairs (x, y) and we define L_R to be the set of x such that $\exists y(x, y) \in R$.

The goal of an algorithm solving a search problem is to find a y such that $(x, y) \in R$. The focus of pseudo-determinism is to give algorithms for search problems that find canonical solutions; a pseudo-deterministic algorithm will output the same solution to a search problem with high probability over its randomness. [40] extended the notion of pseudo-determinism to interactive proofs and brought the concept of NP search problems with unique answers under the umbrella of pseudo-determinism. We will refer to this work's definition of a pseudo-deterministic proof. The pseudo-deterministic proofs in our setting will always either output the unique solution or $\perp$.

Definition 2.2.2 (Pseudo-deterministic proof [40])**.** A search problem R is in *pseudo-deterministic* IP (often denoted psdIP) if there exists a function s where all $x \in L_R$ satisfy $(x, s(x)) \in R$, and an interactive protocol between a probabilistic polynomial time verifier algorithm V and a prover (unbounded algorithm) P such that for every $x \in L_R$:

1. (Canonical Completeness) There exists a P such that $\Pr_r[(P, V)(x, r) = s(x)] \geq \frac{2}{3}$. (We use $(P, V)(x, r)$ to denote the output of the verifier V when interacting with prover P on input x using randomness r).

2. (Canonical Soundness) For all P', $\Pr_r[(P', V)(x, r) = s(x) \text{ or } \perp] \geq \frac{2}{3}$.

And (Standard Soundness) for every $x \notin L_R$, for all provers P', $\Pr_r[(P', V)(x, r) \neq \perp] \leq \frac{1}{3}$.

This is analogous to the definition of pseudo-deterministic NP, except we allow the prover and verifier to interact. In the setting we consider, the prover and verifier both run in polynomial time, with the prover given more time than the verifier. Our goal is to construct pseudo-deterministic proofs for problems such that the verifier runs in time faster than the best known deterministic algorithm for the problem.

2.3 Non-Signaling Interactive Proofs

2.3.1 No-Signaling Games

Definition 2.3.1. A *k-prover, one-round game* is a tuple $\mathcal{G} = (Q_1, ..., Q_k, A_1, ..., A_k, V, \pi)$, where $Q_1, ..., Q_k$ are sets of queries, $A_1, ..., A_k$ are sets of answers,

$$V : Q_1 \times Q_2 \times ... \times Q_k \times A_1 \times A_2 \times ... \times A_k \to \{0, 1\}$$

is a polynomial-time computable function, and π is a polynomial-time sampleable probability distribution over $(Q_1, ..., Q_k)$.

Notation. We denote by $\mathcal{Q} \triangleq Q_1 \times Q_2 \times ...Q_k$ and $\mathcal{A} \triangleq A_1 \times A_2 \times ... \times A_k$. We also denote by $\mathcal{Q}_S \triangleq Q_{s_1} \times Q_{s_2} \times ... \times Q_{s_{|S|}}$, where $S = \{s_1, s_2, ..., s_{|S|}\}$, and similarly for $\mathcal{A}_S$.

We denote by $[k] = \{1, \ldots, k\}$. For every $q = (q_1, \ldots, q_k) \in \mathcal{Q}$, every $a = (a_1 \ldots, a_k) \in \mathcal{A}$, and every $S \subseteq [k]$, we denote by $q_S = (q_i)_{i \in S}$ and $a_S = (a_i)_{i \in S}$.

Definition 2.3.2. A *strategy* for a game $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$ is a family of probability distributions $\{p_q\}_{q \in \mathcal{Q}}$ over $\mathcal{A} \cup \{\perp\}$.

For any $q \in \mathcal{Q}$, any $a \in \mathcal{A}$, and any subset $S \subseteq [k]$, we denote by

$$p_q(a_S) \triangleq \sum_{a^* \in \mathcal{A}: a^*_S = a_S} p_q(a^*).$$

Definition 2.3.3. A strategy $\{p_q\}_{q \in \mathcal{Q}}$ for a k-player game $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$ is said to be *no-signaling* if there exists a family of probability distributions $\{\mathrm{Sim}_{S,q_S}\}_{S \subseteq [k], q_S \in \mathcal{Q}_S}$, where each Sim_{S,q_S} is a distribution over $\mathcal{A}_S$, such that for every $q \in \mathcal{Q}$, every $S \subseteq [k]$, and every $a_S \in \mathcal{A}_S$,

$$p_q(a_S) = \mathrm{Sim}_{S,q_S}(a_S).$$

Namely, a strategy is no-signaling if the marginal distributions of the answers are the same regardless of the other queries. Note that if $\{p_q\}_{q \in \mathcal{Q}}$ is a no-signaling strategy then for every $q \in \mathcal{Q}$,

$$\sum_{a_S} \Pr[p_q = a_S] = \sum_{a_S} \Pr[\mathrm{Sim}_{S,q_S} = a_S] = 1,$$

which implies that $\Pr[p_q = \perp] = 0$.

Two relaxations of the notion of no-signaling were considered in the literature: the first is the notion of *sub-no-signaling*, by Lancien and Winter [60], and the second is the notion of *honest-referee no-signaling* by Holmgren and Yang [49]. In both cases these relaxed notions were motivated by the goal of proving a parallel repetition theorem for no-signaling strategies. We begin by defining the latter notion.

Definition 2.3.4. A strategy $\{p_q\}_{q\in\mathcal{Q}}$ for a k-player game $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$ is said to be *honest-referee no-signaling* if the no-signaling condition holds for every $q \in \mathcal{Q}$ such that $\Pr[\pi = q] > 0$ (and is not required to hold for queries that are not in the support of π).

Formally, $\{p_q\}_{q\in\mathcal{Q}}$ is a honest-referee no-signaling strategy for $\mathcal{G}$ if there exists a family of probability distributions $\{\mathrm{Sim}_{S,q_S}\}_{S\subseteq[k],q_S\in\mathcal{Q}_S}$, where each Sim_{S,q_S} is a distribution over $\mathcal{A}_S$, such that for every $q \in \mathcal{Q}$ in the support of π, every $S \subseteq [k]$, and every $a_S \in \mathcal{A}_S$,

$$p_q(a_S) = \mathrm{Sim}_{S,q_S}(a_S).$$

Definition 2.3.5. A strategy $\{p_q\}_{q\in\mathcal{Q}}$ for a k-player game $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$ is said to be *sub-no-signaling* if there exists a family of probability distributions $\{\mathrm{Sim}_{S,q_S}\}_{S\subseteq[k],q_S\in\mathcal{Q}_S}$, where each Sim_{S,q_S} is a distribution over $\mathcal{A}_S$, such that for every $q \in \mathcal{Q}$, every $S \subseteq [k]$, and every $a_S \in \mathcal{A}_S$,

$$p_q(a_S) \leq \mathrm{Sim}_{S,q_S}(a_S).$$

If $\{p_q\}_{q\in\mathcal{Q}}$ is a sub-no-signaling set of probability distributions, then for every $q \in \mathcal{Q}$, if

$$\sum_{a_S} p_q(a_S) < \sum_{a_S} \mathrm{Sim}_{S,q_S}(a_S) = 1,$$

then in the remaining probability p_q outputs $\perp$.

Definition 2.3.6. Let $\mathrm{NS}(\mathcal{G})$ be the set of no-signaling strategies of a k-prover game $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$. The *no-signaling value* of $\mathcal{G}$ is

$$\mathcal{V}_{\mathrm{NS}}(\mathcal{G}) = \max_{\{p_q\}_{q\in\mathcal{Q}}\in\mathrm{NS}(\mathcal{G})} \sum_{q\in\mathcal{Q}} \pi(q) \sum_{a\in\mathcal{A}} p_q(a)V(q,a).$$

Similarly, let $\mathrm{hrNS}(\mathcal{G})$ be the set of honest-referee no-signaling strategies of $\mathcal{G}$. The *honest-referee no-signaling value* of $\mathcal{G}$ is

$$\mathcal{V}_{\mathrm{hrNS}}(\mathcal{G}) = \max_{\{p_q\}_{q\in\mathcal{Q}}\in\mathrm{hrNS}(\mathcal{G})} \sum_{q\in\mathcal{Q}} \pi(q) \sum_{a\in\mathcal{A}} p_q(a)V(q,a).$$

Let $\mathrm{subNS}(\mathcal{G})$ be the set of sub-no-signaling strategies of $\mathcal{G}$. The *sub-no-signaling value* of $\mathcal{G}$ is

$$\mathcal{V}_{\mathrm{subNS}}(\mathcal{G}) = \max_{\{p_q\}_{q\in\mathcal{Q}}\in\mathrm{subNS}(\mathcal{G})} \sum_{q\in\mathcal{Q}} \pi(q) \sum_{a\in\mathcal{A}} p_q(a)V(q,a).$$

2.3.2 Linear Programming

Definition 2.3.7 ([63])**.** Fix any linear program given by $\max \mathbf{c}^\top \mathbf{x}$ subject to $\mathbf{x}_S \geq 0$, $\mathbf{x}_T$ unrestricted, $A_U\mathbf{x} \leq \mathbf{b}_U$, and $A_V\mathbf{x} = \mathbf{b}_V$, where S, T are disjoint and $S \cup T = [n]$, where $n = |x|$, and where U, V are disjoint and $U \cup V = [m]$ where m is the number of rows of A, where A is defined to be the matrix whose rows are the rows of A_U and the rows of A_V.

The dual of this linear program is defined by $\min \mathbf{b}^\top \mathbf{y}$, where $|\mathbf{y}| = m$, subject to $\mathbf{y}_U \geq 0$, $\mathbf{y}_V$ unrestricted, $A_S^\top \mathbf{y} \geq \mathbf{c}_S$, $A_T^\top \mathbf{y} = \mathbf{c}_T$.

Theorem 2.3.8 (Strong duality [63])**.** *If the value of a linear program is finite then it is equal to the value of its dual.*

Definition 2.3.9 ([89])**.** A mixed packing and covering problem is a pair of non-negative matrices A, C and a pair of non-negative vectors b, d. A solution to a mixed packing and covering problem is a vector x such that $x \geq 0$, $Ax \leq b$, and $Cx \geq d$.

Theorem 2.3.10 ([89]). *Let (A, b, C, d) be a mixed packing and covering problem. Then, there exists an algorithm running in space poly*$(\log(|(A, b, C, d)|), 1/\epsilon)$ *to determine whether there does not exist a solution to the mixed packing and covering problem or to output a solution to the mixed packing and covering problem* $(A, b(1+\epsilon), C, d)$.

Chapter 3

Pseudo-Deterministic Proofs

3.1 Pseudo-deterministic-AM algorithm for graph isomorphism

In this section we give an algorithm for finding an isomorphism between two graphs in AM that outputs the same answer with high probability. The way this algorithm works is that the prover will send the lexicographically first isomorphism to the verifier and then prove that it is the lexicographically first isomorphism. To prove that the isomorphism is the lexicographically first isomorphism, we label the graph and run a sequence of graph non-isomorphism protocols to show no lexicographically smaller isomorphism exists. We present an alternate proof of the same result in the appendix (the proof in the appendix is more group theoretic, whereas the proof below is more combinatorial).

Theorem 3.1.1. *Finding an isomorphism between graphs can be done in* psdAM.

Proof. Let the vertices of G_1 be $v_1, v_2, \ldots, v_n$, and the vertices of G_2 be $u_1, u_2, \ldots, u_n$. We will show an AM algorithm which outputs a unique isomorphism ϕ. Our algorithm will proceed in n stages (which we will later show can be parallelized). After the kth stage, the values $\phi(v_1), \phi(v_2), \ldots, \phi(v_k)$ will be determined.

Suppose that the values $\phi(v_1), \phi(v_2), \ldots, \phi(v_k)$ have been determined. Then we will determine the smallest r such that there exists an isomorphism ϕ^* such that for

$1 \leq i \leq k$, we have $\phi^*(v_i) = \phi(v_i)$, and in addition, $\phi^*(v_{k+1}) = u_r$. If we find r, we can set $\phi(v_{k+1}) = \phi^*(v_{k+1})$ and continue to the $k+1^{th}$ stage.

To find the correct value of r, the (honest) prover will tell the verifier the value of r and ϕ. Then, to show that the prover is not lying, for each $r' < r$, the prover will prove that there exists no isomorphism ϕ' such that for $1 \leq i \leq k$, we have $\phi'(v_i) = \phi(v_i)$, and in addition, $\phi'(v_{k+1}) = u_{r'}$. To prove this, the verifier will pick G_1 or G_2, each with probability 1/2. If the verifier picked G_1, he will randomly shuffle the vertices $v_{k+2}, \ldots, v_n$, and send the shuffled graph to the prover. If the verifier picked G_2, he will set $u'_i = \phi(v_i)$ for $1 \leq i \leq k$, and $u'_{k+1} = u_{r'}$, and shuffle the rest of the vertices. If the prover can distinguish between whether the verifier initially picked G_1 or G_2, then that implies there is no isomorphism sending v_i to $\phi(v_i)$ for $1 \leq i \leq k$, and sending v_{k+1} to $u_{r'}$. The prover now can show this for all $r' \leq r$ (in parallel), as well as exhibit the isomorphism ϕ, thus proving that r is the minimum value such that there is an isomorphism sending v_i to $\phi(v_i)$ for $1 \leq i \leq k$, and sending v_{k+1} to u_r.

The above n stages can be done in parallel in order to achieve a constant round protocol. To do so, in the first stage, the prover sends the isomorphism ϕ to the verifier. Then, the verifier can test (in parallel) for each k whether under the assumption that $\phi(v_1), \phi(v_2), \ldots, \phi(v_k)$ are correct, $\phi(v_{k+1})$ is the lexicographically minimal vertex which v_{k+1} can be sent to. The correctness of this protocol follows from the fact that multiple AM protocols can be performed in parallel (as shown in appendix C.1 of [35]). □

We note that in the above protocol, the prover only needs to have the power to solve graph isomorphism (and graph non-isomorphism). Also, we note that the above protocol uses private coins. While the protocol can be simulated with a public coin protocol [42], the simulation requires the prover to be very powerful. It remains open to determine whether there is a pseudo-deterministic AM protocol for graph isomorphism which uses public coins, and uses a "weak" prover (one which is a polynomial time machine with access to an oracle solving graph isomorphism).

3.2 Lower bound on pseudo-deterministic AM algorithms

In this section, we establish that if any NP-complete problem has an AM protocol that outputs a unique witness with high probability, then the polynomial hierarchy collapses. To do this we show the analog of AM $\subseteq$ NP/*poly* for pseudo-deterministic algorithms, and then use this fact to get a NP/*poly* algorithm with a unique witness. We can then use [45] to show that NP $\subseteq$ coNP/*poly*.

We begin by proving that psdAM $\subseteq$ psdNP/*poly*:

Lemma 3.2.1. *Suppose that there is a* psdAM *protocol for a search problem R, which on input $x \in L_R$, outputs $f(x)$. Then, the search problem R has a* psdNP/*poly algorithm which, on input x, outputs $f(x)$.*

Proof. The idea of the proof is similar to the proof that AM $\subseteq$ NP/*poly* (which in itself uses techniques similar to those of Adleman's theorem [1], showing BPP $\subseteq$ P/*poly*).

Consider the psdAM protocol, and suppose that on input $x \in L_R$, it outputs $f(x)$.

Since we are guaranteed that when the verifier of the the psdAM accepts, it will output $f(x)$ with high probability, we can use standard amplification techniques to show that the verifier will output $f(x)$ with probability $1 - o(\exp(-n))$, assuming an honest prover, and will output anything other than $f(x)$ with probability $o(\exp(-n))$, even with a malicious prover. Then, by a union bound, there exists a choice of random string r that makes the verifier output $f(x)$ for all inputs $x \in \{0,1\}^n$ of size n with an honest prover, and that for malicious provers, the verifier will either reject or output $f(x)$. We encode this string r as the advice string for the NP/*poly* machine.

The NP/*poly* machine computing f can read r off the advice tape and then guess the prover's message, and whenever the verifier accepts, $f(x)$ will be output by that branch. Thus $f(x)$ can be computed by an NP/*poly* machine. $\square$

Next, we show that if an NP-complete problem has a pseudo-deterministic-NP/*poly* algorithm, then the polynomial hierarchy collapses.

Theorem 3.2.2. *Let $L \in \mathrm{NP}$ be an* NP*-complete problem. Let R be a polynomial time algorithm such that there exists a polynomial p so that $x \in L$ if and only if $\exists y \in \{0,1\}^{p(|x|)} R(x,y)$. Suppose that there is a* psdAM *protocol that when given some $x \in L$, outputs a unique $f(x) \in \{0,1\}^{p(|x|)}$ such that $R(x, f(x)) = 1$. Then,* $\mathrm{NP} \subseteq \mathrm{coNP}/poly$ *and the polynomial hierarchy collapses to the third level.*

Proof. From Lemma 3.2.1, we have that there exists psdNP/*poly* algorithm that given $x \in L$, outputs a unique witness $f(x)$ for x. Refer to [45] for the proof that a NP/*poly* function that outputs a unique witness implies that $\mathrm{NP} \subseteq \mathrm{coNP}/poly$. □

3.3 Pseudo-deterministic derandomization for BPP in subexponential time MA

In this section, we will show how to use known circuit lower bounds to get pseudo-deterministic subexponential time (time $O(2^{n^\epsilon})$ for every ϵ) MA protocols for problems in search-BPP for infinitely many input lengths. In [66], it is shown that there is a subexponential time pseudo-deterministic ZPP algorithm for infinitely many input lengths for all properties which have inverse polynomial density, and are testable in polynomial time[1]. A notable example of such a property is primality. So as a corollary, in [66] it is shown that given some integer n, one can find a prime greater than n in pseudo-deterministic subexponential time, for infinitely many input lengths.

In their construction, the condition of high density is required because they show that either there exists a subexponential sized hitting set for infinitely many input lengths which can be used to find strings with the property deterministically, or a complexity collapse happens which implies circuit lower bounds which give pseudo-deterministic algorithms. In the case of MA, unconditional circuit lower bounds for Merlin-Arthur proofs where the verifier is allowed at least half-exponential time have been shown [17, 64], which means that inverse polynomial density is no longer

[1]We believe that their analysis can be improved to get a half-exponential time bounded-error pseudo-deterministic algorithm, which is better than our result for properties of inverse polynomial density.

required. Hence, we can obtain a pseudo-deterministic MA algorithm directly from circuit lower bounds. Compared to [66], our result manages to show a certain pseudo-derandomization for all problems in search-BPP (and not just those with high density), but requires a prover.

Our pseudo-deterministic algorithm uses the Nisan-Wigderson pseudo-random generator. The main lemma that we will use is the lemma that shows that given the truth-table of a hard function, you can get a pseudorandom generator with the right parameters.

Lemma 3.3.1 (Lemma 2.4 of [65])**.** *Let m, n, l be integers; let $f : \{0,1\}^m \to \{0,1\}$ be a boolean function that cannot be approximated with error $\frac{1}{2} - \frac{1}{2n^2}$ by any circuit of size n^2.*

To obtain the best running time for our pseudo-deterministic algorithm, we will need the iterated exponential functions first used in complexity theory by [64]. We will be considering functions with half-exponential growth, i.e. functions f such that $f(f(n)) \in O(2^{n^k})$ for some k.

Definition 3.3.2 (Fractional exponentials [64])**.** The fractional exponential function $e_\alpha(x)$ will be defined as $A^{-1}(A(x) + \alpha)$, where A is the solution to the functional equation $A(e^x - 1) = A(x) + 1$. In addition, we can construct such functions so that $e_\alpha(e_\beta(x)) = e_{\alpha+\beta}(x)$. It is clear from this definition that $e_1(n) = O(2^n)$ as desired.

Theorem 3.3.3. *Given a problem R in search-*BPP*, it is possible to obtain a pseudo-deterministic* MA *algorithm for R where the verifier takes subexponential time for infinitely many input lengths.*

Proof. From [64], we see that MA $\cap$ coMA where the verifier runs in half-exponential time cannot be approximated by polynomial-size circuits. Using the Lemma from [65], this means that in half-exponential time MA, we can construct a pseudorandom generator with half-exponential stretch which is secure against any given polynomial-size circuit for infinitely many input lengths. Let T be the truth-table of the hard function that the pseudorandom generator uses. Then, let R be a relation in search-BPP. Recall from Definition 2.1.8 that there is an algorithm A that given x, produces

y such that $(x, y) \in R$ with high probability if $x \in R_L$. We will now describe the protocol. First, the prover sends T to the verifier and proves that it is indeed the truth table of the hard function in half-exponential time MA (which can be done in half-exponential time). With T in hand, the verifier can then compute the output of the pseudorandom generator. The verifier loops through the seeds in lexicographic order and uses the pseudorandom generator on each seed to create pseudo-random strings, which the verifier then uses as the randomness for A. Each time, the verifier tests whether $(x, A(x, r)) \in R$ (which can be done in BPP, and hence also in MA) and returns the first such valid output.

This will output the same solution whenever the verifier both gets the correct truth-table for the PRG, and succeeds in testing for each PRG output whether the output it provides is valid. Both of these happen with high probability, and thus this is a pseudo-deterministic subexponential-time MA algorithm for any problem in search-BPP which succeeds on infinitely many input lengths. $\square$

3.4 Uniqueness in NL

In this section, we prove that every problem in search-NL can be made pseudo-deterministic:

Theorem 3.4.1 (Pseudo-determinism NL). *Every search problem in search-*NL *is in* psdNL.

One can think of the complete search problem for NL as: given a directed graph G, and two vertices s and t such that there is a path from s to t, find a path from s to t. Note that the standard nondeterministic algorithm of simply guessing a path will result in different paths for different nondeterministic guesses. Our goal will be to find a unique path, so that on different nondeterministic choices, we will not end up with a path which is not the unique one.

The idea will be to find the lexicographically first shortest path (i.e, if the min-length path from s to t is of length d, we will output the lexicographically first path of length d from s to t). To do so, first we will determine the length d of the min-length path from s to t. Then, for each neighbor of s, we will check if it has a path of length $d-1$ to t, and move to the first such neighbor. Now, we have reduced the problem to finding a unique path of length $d-1$, which we can do recursively.

The full proof is given below:

Proof. Given a problem in search-NL, consider the set of all min-length computation histories. We will find the lexicographically first successful computation history in this set.

To do so, we first (nondeterministically) compute the length of the min-length computation history. This can be done because coNL = NL (so if the shortest computation history is of size T, one can show a history of size T. Also, because it is coNL to show that there is no history of size up to $T-1$, we can show that there is no history of size less than T in NL).

In general, using the same technique, given a state S of the NL machine, we can tell what is the shortest possible length for a successful computation history starting at S.

Our algorithm will proceed as follows. Given a state S (which we initially set to be the initial configuration of the NL machine), we will compute T, the length of the shortest successful computation path starting at S. Then, for each possible nondeterministic choice, we will check (in NL) whether there exists a computation history of length $T-1$ given that nondeterministic choice. Then, we will choose the lexicographically first such nondeterministic choice, and recurse.

This algorithm finds the lexicographically first computation path of minimal length which is unique. Hence, the algorithm will always output the same solution (or reject), so the algorithm is pseudo-deterministic. □

3.5 Structural Results

In [36], Goldreich et al showed that the set of total search problems solved by pseudo-deterministic polynomial time randomized algorithms equals the set of total search problems solved by deterministic polynomial time algorithms, with access to an oracle to decision problems in BPP. In [39], this result was extended to the context of RNC. We show analogous theorems here. In the context of MA, we show that for total search problems, psdMA = search$-$P$^{\mathrm{MA}\cap\mathrm{coMA}}$.[2] In other words, any pseudo-deterministic MA algorithm can be simulated by a polynomial time search algorithm with an oracle solving decision problems in MA $\cap$ coMA, and vice versa.

In the case of search problems that are not total, we show that psdMA equals to the class search$-$P$^{\mathrm{promise}-(\mathrm{MA}\cap\mathrm{coMA})}$, where when the input x is in L_R, all queries to the oracle must be in the promise. We note that generally, when having an oracle to a promise problem, one is allowed to query the oracle on inputs not in the promise, as long as the output of the algorithm as a whole is correct for all possible answers the oracle gives to such queries. In our case, we simply do not allow queries to the oracle to be in the promise. Such reductions have been called *smart* reductions [43].

We show similar theorems for AM, and NP. Specifically, we show psdAM = search$-$P$^{\mathrm{promise}-(\mathrm{AM}\cap\mathrm{coAM})}$ and psdNP = search$-$P$^{\mathrm{promise}-(\mathrm{NP}\cap\mathrm{coNP})}$, where the reductions to the oracles are smart reductions.

In the case of total problems, one can use a similar technique to show psdAM = search$-$P$^{\mathrm{AM}\cap\mathrm{coAM}}$ and psdNP = search$-$P$^{\mathrm{NP}\cap\mathrm{coNP}}$, where the oracles can only return answers to total decision problems.

Theorem 3.5.1. *The class* psdMA *equals the class* search$-$P$^{\mathrm{promise}-(\mathrm{MA}\cap\mathrm{coMA})}$*, where on any input $x \in L_R$, the all queries to the oracle are in the promise.*

Proof. The proof is similar to the proofs in [36] and [39] which show similar reductions to decision problems in the context of pseudo-deterministic polynomial time algorithms and pseudo-deterministic NC algorithms.

First, we show that a polynomial time algorithm with an oracle for promise$-$(MA$\cap$ coMA) decision problems which only asks queries in the promise has a corresponding pseudo-deterministic MA algorithm. Consider a polynomial time algorithm A which uses an oracle for promise$-$(MA $\cap$ coMA). We can simulate A by an MA protocol

[2]What we call search$-$P is often denoted as FP.

where the prover sends the verifier the proof for every question which A asks the oracle. Then, the verifier can simply run the algorithm from A, and whenever he accesses the oracle, he instead verifies the proof sent to him by the prover.

We note that the condition of a smart reduction is required in order for the prover to be able to send to the verifier the list of all queries A will make to the oracle. If A can ask the oracle queries not in the promise, it may be that on different executions of A, different queries will be made to the oracle (since A is a adaptive, and the queries A makes may depend on the answers returned by the oracle for queries not in the promise), so the prover is unable to predict what queries A will need answered.

We now show that a pseudo-deterministic MA algorithm B has a corresponding polynomial time algorithm A that uses a promise$-(\mathrm{MA} \cap \mathrm{coMA})$ oracle while only querying on inputs in the promise. On input $x \in L_R$, the polynomial time algorithm can ask the promise$-(\mathrm{MA} \cap \mathrm{coMA})$ oracle for the first bit of the unique answer given by B. This is a decision problem in promise$-(\mathrm{MA} \cap \mathrm{coMA})$ since it has a constant round interactive proof (namely, run B and then output the first bit). Similarly, the algorithm A can figure out every other bit of the unique answer, and then concatenate those bits to obtain the full output.

Note that it is required that the oracle is for promise$-(\mathrm{MA} \cap \mathrm{coMA})$, and not just for promise$-$MA, since if one of the bits of the output is 0, the verifier must be able to convince the prover of that (and this would require a promise$-$coMA protocol). $\square$

A very similar proof shows the following:

Theorem 3.5.2. *The class* psdNP *equals the class* search$-\mathrm{P}^{\text{promise}-(\mathrm{NP}\cap\mathrm{coNP})}$*, where on any input $x \in L_R$, all queries to the oracle are in the promise.*

We now prove a similar theorem for the case of AM protocols. We note that this is slightly more subtle, since it's not clear how to simulate a search$-\mathrm{P}^{\text{promise}-(\mathrm{AM}\cap\mathrm{coAM})}$ protocol using only a constant number of rounds of interaction, since the search-P algorithm may ask polynomial many queries in an adaptive fashion.

Theorem 3.5.3. *The class* psdAM *equals the class* search$-\mathrm{P}^{\text{promise}-(\mathrm{AM}\cap\mathrm{coAM})}$*, where on any input $x \in L_R$, the all queries to the oracle are in the promise.*

Proof. First, we show that a polynomial time algorithm with an oracle for promise$-$(AM$\cap$ coAM) decision problems where the queries are all in the promise has a corresponding pseudo-deterministic AM algorithm. We proceed similarly to the proof of Theorem 3.5.1. Consider a polynomial time algorithm A which uses an oracle for promise$-$(AM$\cap$ coAM). The prover will internally simulate that algorithm A, and then send to the verifier a list of all queries that A makes to the promise$-$(AM$\cap$coAM) oracle. Then, the prover can prove the answer (in parallel), to all of those queries.

To prove correctness, suppose that the prover lies about at least one of the oracle queries. Then, consider the first oracle query to which the prover lied. Then, by a standard simulation argument, one can show that it can be made overwhelmingly likely that the verifier will discover that the prover lied on that query.

Once all queries have been answered by the verifier the algorithm B can run like A, but instead of querying the oracle, it already knows the answer since the prover has proved it to him.

The proof that a pseudo-deterministic MA algorithm B has a corresponding polynomial time algorithm A that uses an promise$-$(AM $\cap$ coAM) oracle is identical to the proof of Theorem 3.5.1 □

As a corollary of the above, we learn that private coins are no more powerful than public coins in the pseudo-deterministic setting:

Corollary 3.5.4. *A pseudo-deterministic constant round interactive proof using private coins can be simulated by a pseudo-deterministic constant round interactive proof using public coins.*

Proof. By Theorem 3.5.3, we can view the algorithm as an algorithm in search$-$P$^{\text{AM}\cap\text{coAM}}$.

By a similar argument to that in Theorem 3.5.3, one can show that psdIP = search$-$P$^{\text{IP}\cap\text{coIP}}$, where in this context IP refers to *constant* round interactive proofs using *private coins*, and AM refers to constant round interactive proofs using *public coins*. Since promise$-$(AM $\cap$ coAM) = promise$-$(IP $\cap$ coIP), since every constant round *private coin* interactive proof for decision problems can be simulated by a constant round interactive proof using *public coins* [42], we have:

$$\mathrm{psdAM} = \mathrm{search}{-}\mathrm{P}^{\mathrm{promise}-(\mathrm{AM}\cap\mathrm{coAM})} = \mathrm{search}{-}\mathrm{P}^{\mathrm{promise}-(\mathrm{IP}\cap\mathrm{coIP})} = \mathrm{psdIP}.$$

□

3.6 Discussion and Open Problems

Pseudo-determinism and TFNP: The class of total search problems solvable by pseudo-deterministic NP algorithms is a very natural subset of TFNP, the set of all total NP search problems. It is interesting to understand how the set of total psdNP problems fits in TFNP. For example, it is not known whether TFNP = psdNP. It would be interesting either to show that every problem in TFNP has a pseudo-deterministic NP algorithm, or to show that under plausible assumptions there is a problem in TFNP which does not have a pseudo-deterministic NP algorithm.

Similarly, it is interesting to understand the relationship of psdNP to other subclasses of TFNP. For example, one can ask whether every problem in PPAD has a pseudo-deterministic NP algorithm (i.e., given a game, does there exists a pseudo-deterministic NP or AM algorithm which outputs a Nash Equilibrium), or whether under plausible assumptions this is not the case. Similar questions can be asked for CLS, PPP, and so on.

Pseudo-determinism in Lattice problems: There are several problems in the context of lattices which have NP (and often also NP$\cap$coNP) algorithms [2]. Notable examples include gap-SVP and gap-CVP, for certain gap sizes. It would be interesting to show pseudo-deterministic interactive proofs for those problems. In other words, one could ask: does there exists an AM protocol for gap-SVP so that when a short vector exists, the *same* short vector is output every time. Perhaps more interesting would be to show, under plausible cryptographic assumptions, that certain such problems *do not* have psdAM protocols.

Pseudo-determinism and Number Theoretic Problems: The problem of generating primes (given a number n, output a prime greater than n), and the problem of finding primitive roots (given a prime p, find a primitive root mod p) have efficient randomized algorithms, and have been studied in the context of pseudo-determinism [44, 32, 66], though no polynomial time pseudo-deterministic algorithms have been found. It is interesting to ask whether these problems have polynomial time psdAM protocols.

The Relationship between psdAM **and** search−BPP**:** One of the main open problems in pseudo-determinism is to determine whether every problem in search−BPP also has a polynomial time pseudo-deterministic algorithm. This remains unsolved. As a step in that direction (and as an interesting problem on its own), it is interesting to determine whether search−BPP $\subseteq$ psdAM. In this chapter, we proved a partial result in this direction, namely that search−BPP $\subseteq$ $i.o.\text{psdMA}_{\text{SUBEXP}}$.

Zero Knowledge Proofs of Uniqueness: The definition of pseudo-deterministic interactive proofs can be extended to the context of Zero Knowledge. In other words, the verifier gets no information other than the answer, and knowing that it is the unique/canonical answer. It is interesting to examine this notion and understand its relationship to psdAM.

The Power of the Prover in pseudo-deterministic interactive proofs: Consider a search problem which can be solved in IP where the prover, instead of being all-powerful, is computationally limited. We know that such a problem can be solved in psdIP if the prover has unlimited computational power (in fact, one can show it is enough for the prover to be in PSPACE). In general, if the prover can be computationally limited for some IP protocol, can it also be computationally limited for a psdIP protocol for the same problem? It is also interesting in general to compare the power needed for the psdIP protocol compared to the power needed to solve the search problem non-pseudo-deterministically. Similar questions can be asked in the context of AM.

The Power of the Prover in pseudo-deterministic private vs public coins proofs: In our psdAM protocol for Graph Isomorphism, the verifier uses private coins, and the prover is weak (it can be simulated by a polynomial time machine with an oracle for graph isomorphism). If using public coins, what power would the prover need? In general, it is interesting to compare the power needed by the prover when using private coins vs public coins in psdAM and psdIP protocols.

Pseudo-deterministic interactive proofs for setting cryptographic global system parameters: Suppose an authority must come up with global parameters for a cryptographic protocol (for instance, a prime p and a primitive root g of p, which would be needed for a Diffie-Hellman key exchange). It may be important that other parties in the protocol know that the authority did not come up with these parameters because he happens to have a trapdoor to them. If the authority proves to the other parties that the parameters chosen are canonical, the other parties now know that the authority did not just pick these parameters because of a trapdoor (instead, the authority had to pick those parameters, since those are the canonical ones). It would be interesting to come up with a specific example of a protocol along with global parameters for which there is a pseudo-deterministic interactive proof showing the parameters are unique.

3.7 Alternate Algorithm for Graph Isomorphism in pseudo-deterministic AM

In this section, we present another psdAM algorithm for Graph Isomorphism, this one more group theoretic (as opposed to the more combinatiorial approach of the algorithm in Section 3.1). The method we use to do this involves finding the lexicographically first isomorphism using group theory. In particular, the verifier will obtain the automorphism group of one of the graphs from the prover and verify that it is indeed the automorphism group, and then the verifier will convert an isomorphism obtained from the prover into the lexicographically first isomorphism between the two graphs. We will define the group-theoretic terms used below.

Definition 3.7.1 (Automorphism Group)**.** The *automorphism group* $Aut(G)$ of a graph is the set of permutations $\phi : G \to G$ such that for every $u, v \in V(G)$, $(u, v) \in E(G) \iff (\phi(u), \phi(v)) \in E(G)$ (i.e., ϕ is an automorphism of G).

Definition 3.7.2 (Stabilize)**.** Given a set S and elements $\alpha_1, \alpha_2, ..., \alpha_i \in S$, we say that a permutation $\phi : S \to S$ *stabilizes* $\{\alpha_1, \alpha_2, ..., \alpha_k\}$ iff $\phi(\alpha_i) = \alpha_i$ for $i \in \{1, ..., k\}$. We also say that a group G *stabilizes* $\{\alpha_1, \alpha_2, ..., \alpha_k\}$ when every $\phi \in G$ stabilizes $\{\alpha_1, \alpha_2, ..., \alpha_k\}$.

Definition 3.7.3 (Stabilizer)**.** The *stabilizer* of an element s in S for a group G acting on S is the set of elements of G that stabilize s.

Lemma 3.7.4. *Suppose that we are given a tuple (G_1, G_2, H, ϕ) where G_1 and G_2 are graphs, $H = Aut(G_1)$ is represented as a set of generators, and ϕ an isomorphism between G_1 and G_2. Then, in polynomial time, we can compute a unique isomorphism ϕ^* from G_1 to G_2 independent of the choice of ϕ and the representation of H.*

Proof. We use the algorithm given in [19] to compute a canonical coset representative, observing that the set of isomorphisms between G_1 and G_2 is a coset of the automorphism group of G_1. Let $\alpha_1, ..., \alpha_t$ be a basis of H, i.e., a set such that any $h \in H$ fixing $\alpha_1, ..., \alpha_t$ is the identity. Let H_i be the subgroup of H that stabilizes $\alpha_1, ..., \alpha_{i-1}$. Now, let U_i be a set of coset representatives of H_{i+1} in H_i. Given the generators of H_i, we can calculate U_i, and by Schreier's theorem we can calculate the generators for H_{i+1}. In this fashion, we can get generators and coset representatives for all the H_i. To produce ϕ^*, we do the following. FIND-FIRST-ISOMORPHISM $\phi^* = \phi$ For $i = 1, ..., t$ Let $P_i = \{\phi^* u | u \in U_i\}$. Set $\phi^* = \arg\min_{\phi \in P_i}(\phi(\alpha_i))$. To see that this produces a unique isomorphism that does not depend on ϕ, observe that $\phi^*(\alpha_1)$ is the minimum possible value of $\phi(\alpha_1)$ over all isomorphisms of G_1 to G_2 as U_1 is a set of coset representatives for the stabilizer of α_1 over H. Also, if $\phi^*(\alpha_i)$ is fixed for $i \in \{1, ..., k\}$, then $\phi^*(\alpha_{k+1})$ is the minimum possible value of $\phi(\alpha_{k+1})$ over all isomorphisms which take α_1 to $\phi^*(\alpha_1)$, α_2 to $\phi^*(\alpha_2)$,..., and α_k to $\phi^*(\alpha_k)$, as U_{i+1}

stabilizes $\alpha_1, ..., \alpha_k$, so everything in P_{i+1} takes α_1 to $\phi^*(\alpha_1)$, α_2 to $\phi^*(\alpha_2)$,..., and α_k to $\phi^*(\alpha_k)$. This implies that ϕ^* does not depend on ϕ and is unique. □

Given this result, this means that it suffices to show a protocol that lets the verifier obtain a set of generators for the automorphism group of G_1 and an isomorphism that are correct with high probability, as by the above lemma this can be used to obtain a unique isomorphism between G_1 and G_2 independent of the isomorphism or the generators.

Theorem 3.7.5. *There exists an interactive protocol for graph isomorphism such that with high probability, the isomorphism that is output by the verifier is unique, where in the case of a cheating prover the verifier fails instead of outputting a non-unique isomorphism. In other words, finding an isomorphism between graphs can be done in* psdAM*.*

Proof. From Lemma 3.7.4, it suffices to show an interactive protocol that computes the automorphism group of a graph in a verifiable fashion. [62] reduces the problem of computing the generators of the automorphism group to the problem of finding isomorphisms. Using this reduction, we can make a constant-round interactive protocol to determine the automorphism group by finding the isomorphisms in parallel. The reason we can do this in parallel is that [62] implies that there are $O(n^4)$ different pairs of graphs to check and for each pair of graphs we either run the graph isomorphism protocol or the graph non-isomorphism protocol. In the case of the graph isomorphism protocol, the verifier need only accept with an isomorphism in hand; for graph non-isomorphism, the messages sent to the prover are indistinguishable between the two graphs when they are isomorphic, so since the graphs and permutations are chosen independently, there is no way for the prover to correlate their answers to gain a higher acceptance probability for isomorphic graphs. Thus this means that the verifier can determine the automorphism group of a graph and verify that it is indeed the entire automorphism group. Using Lemma 3.7.4 we then see that the prover just has to give the verifier an isomorphism, and verifier can compute a unique isomorphism using the automorphism group. □

Chapter 4

Doubly-Efficient Pseudo-Deterministic Proofs

4.1 Doubly-efficient pseudo-deterministic proofs

We want to extend the concept of pseudo-deterministic proofs to the setting where the prover also runs in polynomial time, and we want to extend the concept of doubly-efficient interactive proofs to the setting where the verifier outputs a unique solution. Both of these tasks are accomplished by introducing *doubly-efficient pseudo-deterministic proofs* : proofs where both the verifier and prover run in polynomial time, the verifier running in time asymptotically faster, and where the verifier will output a unique solution given an input.

Definition 4.1.1. A $(t_1(n), t_2(n))$ pseudo-deterministic proof is a pseudo-deterministic proof where the verifier V runs in (probabilistic) time $t_1(n)$ and the prover P runs in (probabilistic) time $t_2(n)$.

Ideally, we want the prover to run in time almost equal to the deterministic running time of the problem, as this means the total work is not much more than the work of solving this problem deterministically. However, we say that pseudo-deterministic proof is *non-trivial* as long as the verifier runs faster than the deterministic running time of the problem. To demonstrate the concept, we will consider the pseudo-deterministic proof for graph isomorphism. The prover from [40] only needs the power to compute n^2 instances of graph isomorphism. We know from [4] that graph isomorphism is in quasi-polynomial time. Thus, the result of [40] about graph isomorphism can be restated as:

Corollary 4.1.2. *Graph Isomorphism has a $(poly(n), quasipoly(n))$ pseudo-deterministic proof.*

A large class of pseudo-deterministic algorithms have the following format: for a search problem R, the pseudo-deterministic algorithm, given x, finds the lexicographically first y such that $R(x, y)$. To do this, it asks whether there exists y' such that $(x, 0y') \in R$, y' such that $(x, 1y') \in R$, etc. and finds the first y such that $R(x, y)$ recursively. For instance, [32] gives a pseudo-deterministic algorithm for testing if a polynomial is non-zero by finding the lexicographically first non-zero solution. Given $p(x_1, ..., x_n)$, the algorithm tests if $p(0, ..., x_n)$ is zero everywhere. If it is not zero everywhere, then the algorithm checks if $p(0, 0, ..., x_n)$ is zero everywhere, and otherwise the algorithm checks if $p(1, ..., x_n)$ is zero everywhere. This continues recursively until the algorithm finds the first element that is non-zero or rejects. Also, [40] provides a pseudo-deterministic proof for graph isomorphism where the verifier outputs the lexicographically first isomorphism by going recursively. The algorithm starts by figuring out where the first vertex is mapped in the lexicographically first isomorphism by looping through the vertices, then where the second vertex is mapped, and so on until the lexicographically first isomorphism has been found.

We will use a structure similar to this to define doubly-efficient pseudo-deterministic proofs for a large class of problems studied within the fine grained complexity literature.

Lemma 4.1.3. *Suppose we have a search problem $R(x, y)$ such that $|y| = poly(x)$, finding the lexicographically first y given x such that $R(x, y)$ takes time $t_1(n)$, computing $R(x, y)$ takes time $t_2(n)$, and y can be written as $y_1...y_k$ such that the following holds:*

- *Given $x, y_1, ..., y_i$, the problem $\exists z_i < y_i \exists y_{i+1}, ..., y_k R(x, y_1, ..., y_{i-1}, z_i, y_{i+1}...y_k)$*

can be solved in co-nondeterministic time $t_3(n)$ where the prover runs in time $t_4(n)$.

Then there exists a $(t_2(n)+k*t_3(n), t_1(n)+k*t_4(n))$ pseudo-deterministic proof that outputs the lexicographically first y such that $R(x,y)$.

Proof. Our algorithm proceeds in two stages: in the first stage, the prover gives y, taking time $t_1(n)$ to find the lexicographically first y such that $R(x,y)$, and the verifier checks whether $R(x,y)$ and outputs $\perp$ otherwise; this takes time $t_2(n)$. In the next stage we prove that y is the lexicographically first such y; that is, for all $z <_{lex} y$, $\neg R(x,z)$. To do so, we only need to check that there is no i such that $\exists z_i < y_i \exists z_{i+1}, ..., z_k$ such that $R(y_1, y_2, ..., y_{i-1}, z_i, z_{i+1}, ..., z_k)$ for $1 \leq i \leq k$. Since we have to do this k times, the total time of this stage is $k*(t_2(n))$ for the verifier and $k*t_4(n)$ for the prover. Our algorithm clearly outputs the lexicographically first y such that $R(x,y)$, and a cheating prover cannot make the verifier output a different y. $\square$

Now that we have shown a general framework for constructing pseudo-deterministic proofs, we will proceed to show a number of problems for which there exist pseudo-deterministic proofs where the verifier runs in time faster than the best known deterministic algorithms. In addition, there is evidence suggesting that the best known deterministic algorithms are nearly optimal; in particular, there is evidence against being able to turn these pseudo-deterministic proofs into deterministic algorithms where the running time of the deterministic algorithm is the same as the running time of the verifier for the pseudo-deterministic proof.

4.2 Linear programming

In [40], we use the fact that graph non-isomorphism has an AM proof to give a pseudo-deterministic AM proof for graph isomorphism. Here we show a pseudo-deterministic proof for linear programming. Linear programming is the class of optimization problems with linear constraints and a linear objective function. We exploit the fact that linear programming admits a good characterization, a compact way of certifying the optimality of a solution. Indeed every linear program (say, where the objective is to maximize) has a corresponding dual linear program, a minimization problem, with the property that (i) (weak duality) any feasible solution to the dual provides an upper bound on the optimal primal value and (ii) (strong duality) there exists an optimal solution to the dual with the same value as the primal optimal solution. Furthermore, there exist compact polynomial-sized solutions to the primal and dual linear programs. Therefore such a polynomial-sized feasible solution to the dual with an equal value as a primal solution provides a compact certificate for the optimality of this primal solution.

In order to be able to turn this into a pseudo-deterministic proof, we need the prover to identify a special, unique optimal solution (as there could be a continuum of primal optimal solutions), and provide a way for the verifier to efficiently verify it. As special solution, we use the *lexicographically greatest* optimal solution to the primal. Among all optimal solutions, the lexicographically greatest first maximizes x_1, then x_2, and so on; see below for a precise definition. To verify it, one option would be to provide dual optimal solutions to a squence of dual linear programs corresponding to the definition of lexicographically greatest maximal solution. A better (more efficient) way, which we describe in this section, is to show that we can perturb the objective function of the primal linear program in such a way that there is a unique optimal solution and that this solution is the unique lexicographically greatest optimal solution for the unperturbed linear program.

We start with basic notation and linear programming fundamentals.

Definition 4.2.1. A linear program is the problem $\max\{\mathbf{c}^\top \mathbf{x}\}$ subject to the constraints $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$. Its dual is the linear program $\min\{\mathbf{b}^\top \mathbf{y}\}$ subject to the constraints $A^\top \mathbf{y} \geq \mathbf{c}$ and $\mathbf{y} \geq \mathbf{0}$.

Theorem 4.2.2. *(Weak duality) If $\mathbf{x}, \mathbf{y}$ are feasible solutions to a linear program given by $\max\{\mathbf{c}^\top \mathbf{x}\}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ and its dual respectively, then $\mathbf{c}^\top \mathbf{x} \leq \mathbf{b}^\top \mathbf{y}$. (Strong duality) Furthermore $\mathbf{x}, \mathbf{y}$ are optimal solutions if and only if*

$\mathbf{c}^\top \mathbf{x} = \mathbf{b}^\top \mathbf{y}$.

Furthermore, there exist optimal solutions of polynomial size, since any extreme point (which cannot be expressed as a strict convex combination of feasible points) has this property.

Theorem 4.2.3 ([33])**.** *Let P be the linear program given by $\max \mathbf{c}^\top \mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$, where all inputs are integers and A is an $m \times n$ matrix. Define $L = m + n + \log(\max_{A'} |det(A')|) + \log(\max_i |b_i|) + \log(\max_j |c_j|)$, where A' range over all square submatrices of A. Then any extreme point $\mathbf{x}$ of P is of the form $x_i = \frac{p_i}{q}$ where q and p_i's are integers satisfying $1 \leq q < 2^L$ and $0 \leq p_i < 2^L$ for all i.*

This quantity L is often used when referring to efficiency of linear programming algorithms, and can be seen (see [33]) to be polynomially related to the binary encoding of all the input data.

Definition 4.2.4. The *lexicographically greatest* optimal solution $\mathbf{x}^*$ to a linear program $\max\{\mathbf{c}^\top \mathbf{x}\}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ is the solution that satisfies (i) feasibility: $A\mathbf{x}^* \leq \mathbf{b}$ and $\mathbf{x}^* \geq \mathbf{0}$, (ii) optimality: $\mathbf{c}^\top \mathbf{x}^* = \max_{A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}}\{\mathbf{c}^\top \mathbf{x}\}$, and (iii) for every $\mathbf{x} \in \arg\max_{A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}}\{\mathbf{c}^\top \mathbf{x}\}$, either $\mathbf{x} = \mathbf{x}^*$ or there exists $i \leq n$ with $x_i < x_i^*$ and $x_j = x_j^*$ for $j < i$.

Now that we have defined the necessary terminology, we can proceed to proving that linear programming has a pseudo-deterministic interactive proof. To do so, we perturb our linear program so that the only optimal solution to the new linear program is the lexicographically greatest solution to the original program, and then use the dual linear program to prove that the solution given to the verifier is optimal.

Theorem 4.2.5. *Let P be the linear program given by $\max \mathbf{c}^\top \mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$, with L defined as above. Then, the linear program P' given by $\max \mathbf{c}^\top \mathbf{x} + \epsilon x_1 + \epsilon^2 x_2 + ... + \epsilon^n x_n$, where $\epsilon = 2^{-3L-2}$, has a unique solution which is the lexicographically greatest solution of P.*

Proof. First consider the unperturbed linear program P, and two extreme point solutions $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, with corresponding denominators q_1 and q_2 respectively (see Theorem 4.2.3).

If $\mathbf{c}^\top \mathbf{x}^{(1)} > \mathbf{c}^\top \mathbf{x}^{(2)}$ then $\mathbf{c}^\top \mathbf{x}^{(1)} - \mathbf{c}^\top \mathbf{x}^{(2)} \geq \frac{1}{q_1 q_2} > 2^{-2L}$. Let $\mathbf{c}'$ be the perturbed $\mathbf{c}$ (by adding the vector $(\epsilon, \epsilon^2, \cdots, \epsilon^n)$). Then

$$\mathbf{c}'^\top \mathbf{x}^{(1)} - \mathbf{c}'^\top \mathbf{x}^{(2)} > 2^{-2L} + \sum_{i=1}^{n} \epsilon^i (x_i^{(1)} - x_i^{(2)}) > 2^{-2L} - 2^L \sum_{i=1}^{n} \epsilon^i > 2^{-2L} - 2^L \epsilon/(1-\epsilon) > 0,$$

given our choice of ϵ. This shows that, after perturbation, we still have that $\mathbf{x}^{(1)}$ has a greater objective value than $\mathbf{x}^{(2)}$.

Suppose, on the other hand, that $\mathbf{c}^\top \mathbf{x}^{(1)} = \mathbf{c}^\top \mathbf{x}^{(2)}$ and that $\mathbf{x}^{(1)}$ is lexicographically greater than $\mathbf{x}^{(2)}$, i.e. that $x_i^1 > x_i^2$ while $x_j^1 = x_j^2$ for $j < i$. Then

$$\mathbf{c}'^\top \mathbf{x}^{(1)} - \mathbf{c}'^\top \mathbf{x}^{(2)} = \sum_{k=i}^{n} \epsilon^k (x_k^1 - x_k^2) \geq \epsilon^i \left(\frac{1}{q_1 q_2} - \sum_{\ell=1}^{n-i} \epsilon^\ell 2^L \right) > \epsilon^i \left(2^{-2L} - \frac{\epsilon}{1-\epsilon} 2^L \right) > 0,$$

showing that, after perturbation, the lexicographically greater solution $\mathbf{x}^{(1)}$ has greater (perturbed) objective function value. Together, this shows that the unique optimal solution to the perturbed problem is the lexicographically greatest solution to P. $\square$

Observe that the parameter L' of the perturbed linear program increases polynomially to $O(nL)$, but the precision needed to solve the linear program approximately in order to be able to recover the unique extreme point solution is still $2^{-O(L)}$, as this represents a lower bound on the difference in value between any two extreme point solutions.

Theorem 4.2.6. *There exists a $(O(d^2 \log(1/\delta)), \tilde{O}(d^\omega \log(1/\delta)))$ pseudo-deterministic interactive proof for finding an optimal solution to a linear program P.*

Linear programs with at most d variables and constraints can be solved within an error of δ in time $\tilde{O}(d^{2.5} \log(1/\delta))$ deterministically [81], and in time $\tilde{O}(d^\omega \log(1/\delta))$ randomized [24] with ω (currently ~ 2.37) corresponds to the exponent for fast matrix multiplication. The notation $\tilde{O}$ hides polylog factors. The time to verify a pair of primal and dual optimal solution is only $O(d^2)$ (with a $\log(1/\delta)$ factor for bit complexity) as this only requires matrix vector multiplication. So, verification is currently more efficient than finding the solution.

Proof. By Theorem 4.2.5, we can perturb the objective function of P and obtain a linear program P' which has a unique solution, namely the lexicographically greatest solution of P. Let Q' be the dual linear program to P'. The prover sends over optimal solutions to P' and Q'. Then, the verifier checks to see whether the solutions are feasible and also whether the value of the solution to P' is equal to the value of the solution to Q'. If both of these conditions hold, the verifier outputs the solution to P', otherwise it outputs $\perp$. If the prover is honest, then clearly the verifier will output the solution to P'. A cheating prover cannot make the verifier output a different solution to P, as this would not correspond to an optimal solution of P' since it is unique. $\square$

4.3 Problems studied in fine-grained complexity

4.3.1 3-SUM and problems reducible to 3-SUM

3-SUM is the problem to find 3 numbers that sum to 0, where the numbers are drawn from 3 lists. The 3-SUM problem has an easy $O(n^2)$ time algorithm and this can be improved by polylogarithmic factors [21]. It is an outstanding open question whether there is an algorithm that is much faster than $O(n^2)$, and finding such an algorithm would give faster algorithms for a host of other problems in computational geometry [31, 27]. We will show a pseudo-deterministic proof where the verifier runs in time $\tilde{O}(n^{1.5})$.

Definition 4.3.1. We say the 3-SUM problem is the problem of, given 3 lists $a_1, ..., a_n$, $b_1, ..., b_n$, $c_1, ..., c_n$, of $O(\log n)$ bit integers, finding a triple a_i, b_j, c_k such that $a_i + b_j + c_k = 0$.

In addition, [20] gives a nondeterministic proof that there is no triple of elements that sum to 0 where the verifier takes time $\tilde{O}(n^{1.5})$.

Theorem 4.3.2 ([20])**.** *3-SUM* $\in \text{coNTIME}(\tilde{O}(n^{1.5}))$.

Theorem 4.3.3. *There exists a non-deterministic proof for* $\overline{\textit{3-SUM}}$ *where the verifier runs in time* $\tilde{O}(n^{1.5})$ *and the prover runs in randomized time* $\tilde{O}(n^2)$.

Proof. It takes a bit of work to show that the prover for this algorithm can run in randomized time $O(n^2)$. What the algorithm does is sends a prime p such that fewer than $\tilde{O}(n^{1.5})$ triples sum to 0 mod p, and all of the triples that add to 0 mod p. Since there are n^3 triples and each sum must be a product of at most $\log(n)$ primes, we get that there are $\tilde{O}(n^3)$ pairs (a_i, b_j, c_k, p) such that $a_i + b_j + c_k = 0 \pmod{p}$. Thus, in the first $n^{1.5}$ primes, over half the primes p will have $\tilde{O}(n^{1.5})$ triples that sum to 0 mod p by Markov's inequality. Thus if we sample a random prime in the first $n^{1.5}$ primes, we will get a good prime with high probability. Finding the sums equal to 0 mod p still takes time $\tilde{O}(n^2)$ deterministically. □

With this, we can construct a pseudo-deterministic proof for 3-SUM where the prover runs in time almost equal to the best known deterministic algorithm for 3-SUM.

Theorem 4.3.4. *3-SUM has a* $(\tilde{O}(n^{1.5}), \tilde{O}(n^2))$ *pseudo-deterministic proof.*

Proof. We split the answer y into $y_1 = i$, $y_2 = j$, and $y_3 = k$. We have a $\tilde{O}(n^{1.5})$ algorithm for proving that a list has no 3 integers which sum to 0. To check whether there is a 3-SUM with $z_i < y_i$, we can simply replace the first $i-1$ lists with $y_1, ..., y_{i-1}$ respectively and take out all of the elements of the ith list after and including y_i, and then use a nondeterministic proof to show that there is no 3-SUM in these lists. It takes time $\tilde{O}(n^2)$ to find a 3-SUM and the prover takes randomized time $\tilde{O}(n^2)$ in the nondeterministic proof that there is no 3-SUM. Thus Lemma 4.1.3 implies that there is a pseudo-deterministic proof where the verifier runs in time $\tilde{O}(n^{1.5})$ and the prover runs in randomized time $\tilde{O}(n^2)$. □

Corollary 4.3.5. *Determining whether there are three collinear points in a set of points on the plane has a* $(\tilde{O}(n^{1.5}), \tilde{O}(n^2))$ *pseudo-deterministic proof.*

4.3.2 Hitting Set

The Hitting Set problem is, given two collections of sets, find a set in the first collection that intersects every set in the second collection. The Hitting Set problem is also conjectured to take $m^{2-o(1)}$ time [82]. Here we give a pseudo-deterministic proof in which the verifier runs in linear time.

Definition 4.3.6. The Hitting Set problem is, given two collections $\mathcal{S}, \mathcal{T}$ of sets, find a set S such that $S \cap T \neq \emptyset \forall T \in \mathcal{T}$.

Theorem 4.3.7 ([20])**.** *There is a nondeterministic proof where the verifier runs in time $O(m), m = \sum_{S \in \mathcal{S}} |S| + \sum_{T \in \mathcal{T}} |T|$, and the prover runs in time $O(m^2)$ for the Hitting Set problem and the complement of the Hitting Set problem.*

Theorem 4.3.8. *Hitting Set has a $(O(m), O(m^2))$ pseudo-deterministic proof.*

Proof. We can reduce the problem of showing there is no set S' that is a hitting set before S to Hitting Set by removing all of the sets after S including S and proving that there does not exist a hitting set for T. Then, by Lemma 4.1.3, this implies there exists a pseudo-deterministic proof for Hitting Set where the verifier runs in time $O(m)$ and the prover runs in time $O(m^2)$. □

4.3.3 Model checking of graph properties

A large number of different graph problems can be expressed as model checking of first-order properties as observed by [20]. For instance both the k-Dominating Set problem [68] and asking whether a graph has diameter 2 [14] can be written as model checking problems. [86] shows that given a first-order property of a graph with k quantifiers over vertices, checking whether the graph has this property can be done in time $\tilde{O}(n^{k-3+\omega})$. We extend the work of [20] on sparse graphs to provide pseudo-deterministic proofs.

Definition 4.3.9. We say a graph property is a formula $Q_1 x_1 \in X_1 Q_2 x_2 \in X_2 ... Q_k x_k \in X_k \psi$, where ψ is a quantifier-free formula on edge predicates and the model checking problem for a graph property is to determine whether the property holds for a given graph.

Theorem 4.3.10 ([20])**.** *If a formula with k does not have the form $\exists^{k-1}\forall$, then the model checking problem for the formula can be solved in co-nondeterministic time m^{k-2} where m is the number of edges in the graph.*

Theorem 4.3.11 ([20])**.** *The deterministic complexity of model checking a k-quantifier formula is $O(m^{k-1})$.*

Theorem 4.3.12. *If a formula does not have the form $\exists^{k-1}\forall$, there exists a $(O(m^{k-2}), O(m^{k-1}))$ pseudo-deterministic proof for finding a setting to the first set of existential quantifiers of that formula.*

Proof. If the first i quantifiers are $\exists$, then we can find $x_1, ..., x_i$ such that $Q_{i+1}x_{i+1}...Q_k x_k \psi(x_1, ..., x_i)$ nondeterministically in time $O(m^{k-i})$ for any $1 \leq j \leq i$, and we can check for any $1 \leq j \leq i$ that $\exists x'_j < x_j Q_{j+1}x_{j+1}...Q_k x_k \psi(x_1, ..., x_{j-1})$ in co-nondeterministic time $O(m^{k-2})$ by setting $X'_j = X_j \cap \{x | x < x_j\}$. For both of these checks, the prover has to solve a model checking problem with at most k quantifiers, which has complexity $O(m^{k-1})$. This shows that there is a $O(m^{k-2})$ pseudo-deterministic proof where the prover runs in time $O(m^{k-1})$ for finding a setting to the first set of existential quantifiers of a formula, if the formula does not have the form $\exists^{k-1}\forall$. □

4.3.4 Problems equivalent to All-Pairs Shortest Path

The All-Pairs Shortest Path problem has been the focus of much research in fine-grained complexity. It has been shown by [88, 87] that many problems related to graphs reduce to the All-Pairs Shortest Path problem and vice versa, so finding a faster algorithm for any one of these problems would yield a fast algorithm for a host of graph problems. [20] shows that the Zero Weight Triangle problem, which is equivalent to the All-Pairs Shortest Path problem under subcubic reductions [87], has a $O(n^{3-\epsilon})$ co-nondeterministic algorithm, which is faster than all known deterministic algorithms. We use this to construct a pseudo-deterministic proof for the Zero Weight Triangle problem.

Definition 4.3.13. The Zero Weight Triangle problem is given a graph $G = (V, E)$ and edge weights $e(i, j)$, find $i, j, k \in V$ such that $e(i, j) + e(i, k) + e(j, k) = 0$.

Theorem 4.3.14 ([20]). *The Zero Weight Triangle problem has a nondeterministic proof and a co-nondeterministic proof where the verifier runs in time $O(n^{2+\omega/3})$, where ω is the largest number such that matrix multiplication is in time $O(n^{\omega})$.*

Theorem 4.3.15. *The Zero Weight Triangle problem has an $(\tilde{O}(n^{2+\omega/3}), \tilde{O}(n^3))$ pseudo-deterministic proof.*

Proof. There is an easy reduction from Zero Weight Triangle to Zero Weight Triangle on tripartite graphs. Then, we remove all edges in the first column going from $i' \geq i$ to j, and thus the resulting graph has a triangle with zero weight iff there exists a triangle in the original graph with zero weight and $i' < i$, where i is the smallest vertex in the claimed lexicographically first zero weight triangle. A similar argument as the argument showing the prover for 3-SUM runs in randomized time $\tilde{O}(n^2)$ shows that the prover for the pseudo-deterministic proof of Zero Weight Triangle runs in randomized time $\tilde{O}(n^3)$. By Lemma 4.1.3, this implies that Zero Weight Triangle has a pseudo-deterministic proof where the verifier runs in time $\tilde{O}(n^{2+\omega/3})$ and the prover runs in $\tilde{O}(n^3)$. □

4.4 Conclusions and Open Problems

We defined the notion of doubly-efficient pseudo-deterministic proofs and gave a number of examples of search problems for which we showed doubly-efficient pseudo-deterministic proofs. In all of these cases, the verifier runs faster than the best known probabilistic algorithm for the problem which can offer significant improvements for settings in which a more powerful computer (cloud, special purpose device, centralized authority) can perform the computation first and prove it to a significantly less powerful user. In all these cases the prover's computation increases polynomially from what is necessary to solve the problem without need for a canonical solution. An interesting problem would be show that this is true in general. Namely, that for any doubly-efficient pseudo-eterministic proof the computation of the prover need be no more than whats necessary to find the canonical solution. Finally, we remark that in all the cases we treated, the canonical solution was the lexicographically smallest (or largest as in the LP case) but other canonical solutions are possible.

Chapter 5

Non-Signaling Proofs

5.1 Non-Signaling Games with k Players and $2^{-\Omega(k^2)}$ Soundness are in $\mathrm{SPACE}\left(\mathbf{poly}(n, 2^{k^2})\right)$

In what follows we state our main theorem.

Theorem 5.1.1. *Fix any language $L \notin \mathrm{SPACE}(poly(n, 2^k))$ and any k-prover one-round proof system $(P_1, \ldots, P_k, V)$ for L with completeness $c \geq 1 - 2^{-5k^2}$. For every x consider the game $\mathcal{G}_x = (\mathcal{Q}, \mathcal{A}, V, \pi_x)$, where $\mathcal{Q} = \mathcal{Q}_1 \times \ldots \times \mathcal{Q}_k$ and where $\mathcal{Q}_i$ is the set of possible queries sent by V to prover P_i, $\mathcal{A} = \mathcal{A}_1 \times, \ldots, \mathcal{A}_k$ and where $\mathcal{A}_i$ is the set of possible answers sent by P_i, and π_x is the distribution of queries sent by $V(x)$.*

Then, there exists a constant $a > 0$ and an infinite set $N \subseteq \mathbb{N}$, such that for every $n \in N$ there exists $x \in \{0,1\}^n \setminus L$ such that $\mathcal{V}_{\mathrm{NS}}(\mathcal{G}_x) \geq 2^{-a \cdot k^2}$.

We present two different proofs for Theorem 5.1.1. Both make use of the following theorem which is the main technical contribution of this work.

Theorem 5.1.2. *There exists a constant $c \in \mathbb{N}$, such that for any $k \in \mathbb{N}$ and any k-player game $\mathcal{G}$, if $\mathcal{V}_{\mathrm{subNS}}(\mathcal{G}) \geq 1 - 2^{-4k^2}$ then $\mathcal{V}_{\mathrm{NS}}(\mathcal{G}) \geq 2^{-c \cdot k^2}$.*

We defer the proof of Theorem 5.1.2 to Section 5.2, and refer the reader to Section **??** for the high-level overview of the proof. In what follows we present our proofs for Theorem 5.1.1, both using Theorem 5.1.2 as a building block.

In the first proof, presented in Section 5.1.1, we rely on a prover reduction theorem which shows that one can convert any k-player game with non-signaling value at most $2^{-O(k^2)}$ into a 2-player game with non-signaling value at most $1-2^{-\Omega(k^2)}$. In the second proof, presented in Section 5.1.2, we rely on the fact that one can approximate the sub-non-signaling of a k-player game up to an additive factor ϵ, using an algorithm running in space $\mathrm{poly}(\mathrm{cc}, 2^k, 1/\epsilon)$, where cc is the communication complexity of the game.

5.1.1 From Multi-Prover Non-Signaling Proofs to 2-Prover Non-Signaling Proofs

In the classical setting there is a well known reduction that converts any k-player game into a 2-player. Below we present a slight variant of it, that will be useful in the non-signaling setting.

Let $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$ be a k-player game. Consider the following 2-player game, denoted by $\mathcal{T}(\mathcal{G}) = (\mathcal{Q}^*, \mathcal{A}^*, V^*, \pi^*)$:

- $\mathcal{Q}^* = (\mathcal{Q}_1^*, \mathcal{Q}_2^*)$, where $\mathcal{Q}_1^* = \mathcal{Q}$ and $\mathcal{Q}_2^* = \{S, q_S\}_{S\subseteq[k], q_S\in\mathcal{Q}_S}$.
- $\mathcal{A}^* = (\mathcal{A}_1^*, \mathcal{A}_2^*)$, where $\mathcal{A}_1^* = \mathcal{A}$ and $\mathcal{A}_2^* = \bigcup_{S\subseteq[k]} \mathcal{A}_S$.
- π^* generates $q \leftarrow \pi$ and generates a random subset $S \subseteq [k]$. It outputs $(q, (S, q_S))$.
- $V^*((q, (S, q_S)), (a, a'_S))$ accepts if and only if $V(q, a)$ accepts and $a_i = a'_i$ for every $i \in S$.

Theorem 5.1.3. *Let $\mathcal{G}$ be a k-player game with non-signaling value less than 2^{-ck^2} (where $c \in \mathbb{N}$ is the constant from Theorem 5.1.2). Then the 2-player game $\mathcal{T}(\mathcal{G})$ has non-signaling value at most $1 - 2^{-5k^2}$.*

Proof. Let $\mathcal{G}$ be a k-player game with non-signaling value less than 2^{-ck^2}. Suppose for the sake of contradiction that the non-signaling value of the 2-player game $\mathcal{T}(\mathcal{G})$

is $1-\epsilon$, for $\epsilon < 2^{-5k^2}$. Let $\{p_{q,(S,q_S)}\}$ be a non-signaling strategy that convinces the verifier V^* in the game $\mathcal{T}(\mathcal{G})$ to accept with probability $1-\epsilon$.

Consider the sub-non-signaling strategy $\{p_q\}$ for the k-player game $\mathcal{G}$, where P_q samples answers as follows:

1. For every $S \subseteq [k]$, sample $(a, a'_S) \leftarrow p_{q,(S,q_S)}$.
2. If there exists $S \subseteq [k]$ such that the above answers are rejecting (i.e., $V^*((q,(S,q_S)),(a,a'_S)) = 0$) then output $\perp$.
3. Otherwise, choose a random $S \subseteq [k]$ and output a corresponding to this S.

Claim 5.1.4. *$\{p_q\}$ is a sub-non-signaling strategy for the k-player game $\mathcal{G}$.*

Proof. By definition, the fact that $\{p_{q,(S,q_S)}\}$ is a non-signaling distribution for the 2-player game $\mathcal{T}(\mathcal{G})$, implies that there is a family of distributions $\{\mathrm{Sim}_q\} \cup \{\mathrm{Sim}_{S,q_S}\} \cup \{\mathrm{Sim}_{q,(S,q_S)}\}$ such that for every $q \in \mathcal{Q}$, for every $S \subseteq [k]$ and every $a_S \in \mathcal{A}_S$,

$$\Pr[p_{q,(S,q_S)}|_{(S,q_S)} = a_S] = \Pr[\mathrm{Sim}_{S,q_S} = a_S].$$

We prove that $\{p_q\}$ is sub-non-signaling with respect to $\{\mathrm{Sim}_{S,q_S}\}$. Namely, we prove that for every $q \in \mathcal{Q}$, every $S \subseteq [k]$, and every $a_S \in \mathcal{A}_S$,

$$\Pr[p_q|_S = a_S] \leq \Pr[\mathrm{Sim}_{S,q_S} = a_S]. \tag{5.1}$$

We note that Equation (5.1) would clearly hold if we chose a corresponding to the specific set S in the equation. However, recall that p_q chooses a corresponding to a *random* subset $S' \subseteq [k]$.

Thus, we define for every (fixed) $S \subseteq [k]$ a strategy $\{p_q^S\}$ which is identical to $\{p_q\}$, except that if it doesn't abort then it always outputs a corresponding to the fixed subset S. Therefore, to conclude the proof that $\{p_q\}$ is sub-non-signaling it suffices to prove that for every $q \in \mathcal{Q}$, every $a \in \mathcal{A}$, and every subsets $S, S' \subseteq [k]$, it holds that

$$\Pr[p_q^S = a] = Pr[p_q^{S'} = a],$$

which follows directly from the the fact that $\{p_{q,(S,q_S)}\}$ is non-signaling (together with the definition of $\{p_q^S\}$). $\square$

Note that the sub-non-signaling strategy $\{p_q\}$ is rejected with probability at most $2^k \cdot \epsilon$ (by the union bound).

This in particular implies that the sub-non-signaling value of $\mathcal{G}$ is at least

$$1 - 2^k \cdot \epsilon \geq 1 - 2^k \cdot 2^{-5k^2} \geq 1 - 2^{-4k^2},$$

which by Theorem 5.1.2 implies that the non-signaling value of $\mathcal{G}$ is at least 2^{-ck^2}, contradicting our assumption.

$\square$

5.1.2 Approximating the Sub-Non-Signaling Value of k-Player Game via a Space Efficient Algorithm

Theorem 5.1.5. *There exists an algorithm $\mathcal{B}$ and a polynomial p such that for any k-player game $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$, and any $\epsilon > 0$, it holds that $\mathcal{B}(\mathcal{G}, \epsilon)$ runs in space $p(\log(|\mathcal{Q}, \mathcal{A}|), 1/\epsilon, 2^k)$ and outputs a value v such that $|v - \mathcal{V}_{\text{subNS}}(\mathcal{G})| \leq \epsilon$.*

Corollary 5.1.6. *Fix any language L and any k-prover one-round proof system $(P_1, \ldots, P_k, V)$ for L. For every x consider the game $\mathcal{G}_x = (\mathcal{Q}, \mathcal{A}, V, \pi_x)$, where $\mathcal{Q} = \mathcal{Q}_1 \times \ldots \times \mathcal{Q}_k$ and where $\mathcal{Q}_i$ is the set of possible queries sent by V to prover P_i, $\mathcal{A} = \mathcal{A}_1 \times, \ldots, \mathcal{A}_k$ where $\mathcal{A}_i$ is the set of possible answers sent by P_i, and π_x is the distribution of queries sent by $V(x)$.*

Denote by c the completeness of this proof system.[1] *If there exists a constant $d \in \mathbb{N}$, such that for every large enough $n \in \mathbb{N}$, and every $x \in \{0,1\}^n \setminus L$, $\mathcal{V}_{\text{subNS}}(\mathcal{G}_x) \leq c - \frac{1}{n^d}$, then $L \in \text{SPACE}\left(poly(n, 2^k)\right)$.*[2]

[1]A proof system is said to have completeness c if for every $x \in L$ the honest provers convince the verifier to accept $x \in L$ with probability at least c. In particular, this implies that $\mathcal{V}_{\text{subNS}}(\mathcal{G}_x) \geq c$ for every $x \in L$.

[2]This is assuming the communication complexity is $\text{poly}(n)$. In the general case, where the communication complexity is cc, we get that $L \in \text{SPACE}\left(\text{poly}(n, \text{cc}, 2^k)\right)$

Proof of Corollary 5.1.6. Fix any language L and any k-prover one-round proof system $(P_1, \dots, P_k, V)$ for L with completeness c. For every $x \in \{0,1\}^*$, consider the corresponding game $\mathcal{G}_x$ as defined in the corollary statement. Suppose that there exists a constant $d \in \mathbb{N}$, such that for every large enough $n \in \mathbb{N}$, and every $x \in \{0,1\}^n \setminus L$, $\mathcal{V}_{\mathrm{subNS}}(\mathcal{G}_x) \leq c - \frac{1}{n^d}$.

Fix $\epsilon = \frac{1}{10 \cdot n^d}$. From Theorem 5.1.5 we know that there exists an algorithm $\mathcal{B}$, that given any k-prover game $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$, and any parameter ϵ, approximates the value of $\mathcal{G}$ up to an additive ϵ error. Importantly $\mathcal{B}$ is an algorithm with space complexity $\mathrm{poly}(\log(|(\mathcal{Q}, \mathcal{A})|), 1/\epsilon, 2^k)$.

Given $x \in \{0,1\}^*$, we determine if $x \in L$ by running $\mathcal{B}(\mathcal{G}_x, 1/\epsilon)$, and if the value is at least $c - \epsilon$ then we conclude that $x \in L$, and otherwise conclude that $x \notin L$.

Note that $1/\epsilon$ is a polynomial in n since $\epsilon = \frac{1}{10n^d}$. In addition, the size of $\mathcal{Q}, \mathcal{A}$ is exponential in n, which implies that the space complexity of $\mathcal{B}(\mathcal{G}_x, 1/\epsilon)$ is $\mathrm{poly}(n, 2^k)$, as desired.[3] Finally, we note that there may be a finite number of n's for which we do not have the guarantee that $\mathcal{V}_{\mathrm{subNS}}(\mathcal{G}_x) \leq c - \frac{1}{n^d}$. For these n's, we can hard-wire the answers for whether $x \in L$. $\square$

We next prove Theorem 5.1.5. We use the approach of [52] which proves that the non-signaling value of a two-player, one-round game can be approximated in PSPACE.

Proof of Theorem 5.1.5. Fix any game $\mathcal{G} = (\mathcal{A}, \mathcal{Q}, V, \pi)$. The sub-non-signaling value of $\mathcal{G}$ is given by the following linear program (where the variables are $p_q(a)$ and

[3]More generally, if $(P_1, \dots, P_k, V)$ has communication complexity cc then $|(\mathcal{Q}, \mathcal{A})| \leq 2^{\mathrm{cc}}$, in which case the space complexity of $\mathcal{B}(\mathcal{G}_x, 1/\epsilon)$ is $\mathrm{poly}(n, \mathrm{cc}, 2^k)$, as desired.

$\mathrm{Sim}_{S,q_S}(a_S)$, for every $q \in \mathcal{Q}$, $a \in \mathcal{A}$, and nonempty $S \subseteq [k]$)

$$\begin{array}{lll} \text{Maximize} & \sum_{q\in\mathcal{Q}} \pi(q) \sum_{a\in\mathcal{A}} p_q(a)V(q,a) & \\ \text{Subject to} & \sum_{a^*\in\mathcal{A}:a^*_S=a_S} p_q(a^*)) \leq \mathrm{Sim}_{S,q_S}(a_S) & \forall S \subseteq [k], \forall a_S \in \mathcal{A}_S, \forall q \in \mathcal{Q}, \\ & \sum_{a_S\in\mathcal{A}_S} \mathrm{Sim}_{S,q_S}(a_S) = 1 & \forall S \subseteq [k], \forall q_S \in \mathcal{Q}_S \\ & p_q(a) \geq 0 & \forall a \in \mathcal{A}, \forall q \in \mathcal{Q} \end{array} \tag{5.2}$$

In what follows, we replace $p_q(a)$ with $x_q(a) = \pi(q)p_q(a)$ to simplify the expression of the objective value. This gives us the linear program

$$\begin{array}{lll} \text{Maximize} & \sum_{q\in\mathcal{Q}} \sum_{a\in\mathcal{A}} x_q(a)V(q,a) & \\ \text{Subject to} & \sum_{a^*\in\mathcal{A}:a^*_S=a_S} x_q(a^*) \leq \pi(q)\mathrm{Sim}_{S,q_S}(a_S) & \forall S \subseteq [k], \forall a_S \in \mathcal{A}_S \forall q, \in \mathcal{Q} \\ & \sum_{a_S} \mathrm{Sim}_{S,q_S}(a_S) = 1 & \forall S \subseteq [k], \forall q_S \in \mathcal{Q}_S \\ & x_q(a) \geq 0 & \forall a \in \mathcal{A}, \forall q \in \mathcal{Q} \end{array} \tag{5.3}$$

Observe that the constraints in this linear program above imply that $\mathrm{Sim}_{S,q_S}(a_S) \geq 0$ for every $S \subseteq [k]$, every $q_S \in \mathcal{Q}_S$ and every $a_S \in \mathcal{A}_S$. Namely, these constraints can be added without changing the value of the linear program. This implies (by Definition 2.3.7), that the dual to this linear program can be written as

$$\begin{array}{lll}
\text{Minimize} & \sum_{S\subseteq[k]}\sum_{q_S\in\mathcal{Q}_S} z_S(q_S) & \\
\text{Subject to} & \sum_{S\subseteq[k]} y_S(q,a_S) \geq V(q,a) & \forall q\in\mathcal{Q},\forall a\in\mathcal{A} \\
 & z_S(q_S) \geq \sum_{q^*\in\mathcal{Q}:q^*_S=q_S} \pi(q^*)y_S(q^*,a_S) & \forall S\subseteq[k],\forall q_S\in\mathcal{Q}_S,\forall a_S\in\mathcal{A}_S \\
 & y_S(q,a_S)\geq 0 & \forall S\subseteq[k],\forall q\in\mathcal{Q},\forall a_S\in\mathcal{A}_S
\end{array} \tag{5.4}$$

Observe that the constraints in this linear program imply that $z_S(q_S) \geq 0$ for every $S \subseteq [k]$ and every $q_S \in \mathcal{Q}_S$, and thus these constraints can be added without changing the value.

Next, transform this linear program into a linear program with non-negative coefficients. To do so, observe that the optimal solution to the above linear program satisfies that $y_S(q,a_S) \leq 1$ for every $S \subseteq [k]$, every $q \in \mathcal{Q}$ and every $a_S \in \mathcal{A}_S$. This follows from the fact that $V(a,q) \leq 1$ for every $a \in \mathcal{A}$ and every $q \in \mathcal{Q}$. Therefore, we can replace $y_S(q,a_S)$ by $\overline{y}_S(q,a_S) = 1 - y_S(q,a_S)$, without changing the value of the linear program. This gives us the linear program

$$\begin{array}{lll}
\text{Minimize} & \sum_{S\subseteq[k]}\sum_{q_S\in\mathcal{Q}_S} z_S(q_S) & \\
\text{Subject to} & \sum_{S\subseteq[k]} \overline{y}_S(q,a_S) \le 2^k - 1 - V(q,a) & \forall q\in\mathcal{Q}, \forall a\in\mathcal{A} \\
& z_S(q_S) + \sum_{q^*\in\mathcal{Q}:q^*_S=q_S} \pi(q^*)\overline{y}_S(q^*,a_S) \ge \sum_{q^*\in\mathcal{Q}:q^*_S=q_S}\pi(q^*) & \forall S\subseteq[k], \forall q_S\in\mathcal{Q}_S, \forall a_S\in\mathcal{A} \\
& \overline{y}_S(q,a_S)\le 1 & \forall S\subseteq[k], \forall q\in\mathcal{Q}, \forall a_S\in\mathcal{A}_S \\
& \overline{y}_S(q,a_S)\ge 0 & \forall S\subseteq[k], \forall q\in\mathcal{Q}, \forall a_S\in\mathcal{A}_S \\
& z_S(q_S)\ge 0 & \forall S\subseteq[k], \forall q_S\in\mathcal{Q}_S
\end{array} \tag{5.5}$$

Note that all of the coefficients of this linear program are non-negative.

Recall that our goal is to construct a poly$(\log(|(\mathcal{Q},\mathcal{A})|), 1/\epsilon, 2^k)$-space algorithm for computing v such that

$$|v - \mathcal{V}_{\text{subNS}}(\mathcal{G})| \le \epsilon.$$

To this end, we add to our linear program a constraint of the form

$$\sum_{S\subseteq[k]}\sum_{q_S\in\mathcal{Q}_S} z_S(q_S) \le v'$$

(for some value v'), and convert this (restricted) linear program into a mixed packing and covering program, with the guarantee that for $\delta = \frac{(\epsilon/2)}{2^k}$, a $(1+\delta)$-approximate solution to the mixed packing and covering program, implies a solution to the (restricted) linear program, which is $\epsilon/2$-close an optimal solution. We can then use binary search to find an ϵ-approximation to the original linear program.

To turn this restricted linear program into a mixed packing and covering problem, we use all of the constraints above and include the constraint $\sum_{S\subseteq[k]}\sum_{q_S\in\mathcal{Q}_S} z_S(q_S) \le v'$.

A $(1+\delta)$-approximate solution to a mixed packing and covering problem is (by definition) a solution to the problem where all of the inequalities of the form $a_i x_i \leq c$ are relaxed to $a_i x_i \leq c(1+\delta)$. In our case, it means that the above $\leq$ inequalities are replaced with

$$\overline{y}_S(q, a_S) \leq 1+\delta$$

and

$$\sum_{S \subseteq [k]} \overline{y}_S(q, a_S) \leq (2^k - 1 - V(q, a))(1+\delta).$$

We next argue that a $(1+\delta)$-approximate solution to our mixed packing and covering problem implies a solution to our (restricted) linear program with value at most $v' + \epsilon$.

To this end, suppose there these exists such a solution to the mixed packing and covering problem, and denote it by

$$\left(\{y_S(q, a_S)\}_{S \in [k], q \in \mathcal{Q}, a_S \in \mathcal{A}_S}, \{z_S(q_S)\}_{S \in [k], q_S \in \mathcal{Q}_S}\right).$$

Consider the solution

$$\left(\{y'_S(q, a_S)\}_{S \in [k], q \in \mathcal{Q}, a_S \in \mathcal{A}_S}, \{z'_S(q_S)\}_{S \in [k], q_S \in \mathcal{Q}_S}\right).$$

where

$$y'_S(q, a_S) = \frac{1}{1+\delta} y_S(q, a_S)$$

and

$$z'_S(q_S) = z_S(q_S) + \delta \sum_{q^* \in \mathcal{Q}: q^*_S = q_S} \pi(q^*).$$

It is easy to see that this solution satisfies the constraints of the (restricted) linear program, and thus is a solution to the linear program.

The value of this solution is

$$\begin{aligned}
&\sum_{S\subseteq[k]}\sum_{q_S\in\mathcal{Q}_S} z'_S(q_S) = \\
&\sum_{S\subseteq[k]}\sum_{q_S\in\mathcal{Q}_S}\left(z_S(q_S) + \delta \sum_{q^*\in\mathcal{Q}^*:q^*_S=q_S} \pi(q^*)\right) = \\
&\sum_{S\subseteq[k]}\sum_{q_S\in\mathcal{Q}_S} z_S(q_S) + \delta(2^k-1) \leq \\
&v' + 2^k\delta < v' + \epsilon/2
\end{aligned}$$

From Theorem 2.3.10 we can conclude that approximating the sub-non-signaling value of a game with a constant number of provers takes space $p(\log(|(\mathcal{Q},\mathcal{A})|), 1/\epsilon, 2^k)$.

□

5.1.3 Proof of Theorem 5.1.1 via Corollary 5.1.6

In what follows we prove Theorem 5.1.1. In the proof we rely on Corollary 5.1.6 which implies that if $L \notin \text{SPACE}(n, 2^k)$ then there is an infinite set $N \subseteq \mathbb{N}$ such that for every $n \in N$ there is an element $x \in \{0,1\}^n \setminus L$ such that $\mathcal{V}_{\text{subNS}}(\mathcal{G}_x) \geq c - \frac{1}{n}$. Consider the infinite set $N_0 \subseteq N$ such that for every $n \in N_0$ it holds that $n \geq 2^{5k^2}$. We conclude that for every $n \in N_0$ there exists $x \notin \{0,1\}^n \setminus L$ such that

$$\mathcal{V}_{\text{subNS}}(\mathcal{G}_x) \geq c - \frac{1}{n} \geq c - 2^{-5k^2} \geq 1 - 2^{-5k^2} - 2^{-5k^2} \geq 1 - 2^{-4k^2}.$$

Therefore, to prove Theorem 5.1.1 it suffices to prove the following theorem.

The rest of this section is devoted to the proof of Theorem 5.1.2.

We refer the reader to Section **??** for a high-level overview of the proof of Theorem 5.1.2.

5.2 The Proof of Theorem 5.1.2

In this section we prove Theorem 5.1.2, which is our main technical theorem. In the proof, we use the following theorem from [49].

Theorem 5.2.1. *[49] For every $k \in \mathbb{N}$ there exists a fixed value $\alpha_k \geq 2^{-O(k^2)}$ such that for any k-player game $\mathcal{G}$, $\mathcal{V}_{\mathrm{NS}}(\mathcal{G}) \geq \alpha_k \cdot \mathcal{V}_{\mathrm{hrNS}}(\mathcal{G})$.*

Proof of Theorem 5.1.2. Let $\mathcal{G} = (\mathcal{Q}, \mathcal{A}, V, \pi)$ be a k-player game such that $\mathcal{V}_{\mathrm{subNS}}(\mathcal{G}) \geq 1 - \epsilon$, for $\epsilon = 2^{-4k^2}$. Let $\{p_q\}_{q \in \mathcal{Q}}$ be a sub-non-signaling strategy, such that $\mathcal{G}$ has sub-non-signaling value $1 - \epsilon$ with respect to $\{p_q\}_{q \in \mathcal{Q}}$. Let

$$\mathrm{GOOD} = \{q | \Pr_{a \leftarrow p_q}[V(q,a) = 1] \geq 1 - 2\epsilon\}.$$

Claim 5.2.2. $\Pr_{q \leftarrow \pi}[q \in \mathit{GOOD}] \geq 1/2$.

Proof. We know that

$$\mathbb{E}_{q \leftarrow \pi}[\Pr_{a \leftarrow p_q}[V(q,a) = 0]] \leq \epsilon.$$

By Markov's inequality this implies that

$$\Pr_{q \leftarrow \pi}[\Pr_{a \leftarrow p_q}[V(q,a) = 0] \geq 2\epsilon] \leq 1/2$$

which in turn implies that $\Pr_{q \leftarrow \pi}[q \in \mathrm{GOOD}] \geq 1/2$. □

Consider the distribution $\pi^* = \pi | (q \in \mathrm{GOOD})$, and let $\mathcal{G}^* = (\mathcal{Q}, \mathcal{A}, V, \pi^*)$. Note that $\mathcal{G}^*$ is a game with sub-non-signaling value at least $1 - 2\epsilon$, since $\{p_q\}$ is a sub-non-signaling strategy for $\mathcal{G}^*$ that succeeds with probability at least $1 - 2\epsilon$.

We next define a sub-non-signaling strategy $\{\tilde{p}_q\}$ for the game $\mathcal{G}^*$, such that for every $q \in \mathrm{GOOD}$, every $S \subseteq [k]$, and every $a_S \in \mathcal{A}_S$, it holds that

$$\Pr[\tilde{p}_q|_S = a_S] \leq \mathbb{E}_{q^* \leftarrow \pi^* | (q^*_S = q_S)}[\Pr[p_{q^*}|_S = a_S]] \tag{5.6}$$

and for every $q \in \text{GOOD}$,

$$\Pr_{a \leftarrow \tilde{p}_q}[V(q,a)=1] \geq 1 - 2^{k+2}\epsilon. \tag{5.7}$$

To this end, we define $\tilde{p}_q$ in a greedy manner, so that Equation (5.6) holds, while keeping the invariant that for every $q \in \text{GOOD}$ and $a \in \mathcal{A}$ it holds that $\tilde{p}_q(a) \leq p_q(a)$. This is done as follows: Fix any $q \in \text{GOOD}$. Start with $\tilde{p}_q = p_q$. For every $S \subseteq [k]$ and every a_S, if

$$\Pr[\tilde{p}_q|_S = a_S] > \mathbb{E}_{q^* \leftarrow \pi^*|(q^*_S = q_S)}[\Pr[p_{q^*}|_S = a_S]]$$

then (arbitrarily) reduce $\tilde{p}_q(a^*)$ for every $a^* \in \mathcal{A}$ such that $a^*_S = a_S$ so that

$$\Pr[\tilde{p}_q|_S = a_S] = \mathbb{E}_{q^* \leftarrow \pi^*|(q^*_S = q_S)}[\Pr[p_{q^*}|_S = a_S]],$$

and in the remaining probability output $\perp$. For each S and a_S, this step reduces the probability that V accepts by at most

$$\delta_{S,a_S}(q) \triangleq \max\{0, \Pr[p_q|_S = a_S] - \mathbb{E}_{q^* \leftarrow \pi^*|(q^*_S = q_S)}[\Pr[p_{q^*}|_S = a_S]\}.$$

This follows from the invariant that for every a it holds that $\tilde{p}_q(a) \leq p_q(a)$. Since we do this for every $S \subseteq [k]$ and every a_S, in total the probability that V accepts is reduced by at most

$$\delta(q) = \sum_{S,a_S} \delta_{S,a_S}(q).$$

Note that Equation (5.6) holds by definition of $\{\tilde{p}_q\}$. To prove Equation (5.7), it suffices to prove the following claim.

Claim 5.2.3. *For every $q \in$ GOOD, it holds that $\delta(q) \leq 2^{k+1}\epsilon$.*

Proof. Since $\{p_q\}$ is a sub-non-signaling strategy, there exists a family of distributions

$\{\mathrm{Sim}_{S,q_S}\}$ such that for every $S \subseteq [k]$ and every $q_S \in \mathcal{Q}_S$ and $a_S \in \mathcal{A}_S$,

$$\max_{q^* \text{ s.t. } q^*_S = q_S} \Pr[p_{q^*}|_S = a_S] \leq \Pr[\mathrm{Sim}_{S,q_S} = a_S].$$

Therefore,

$$\Pr[\mathrm{Sim}_{S,q_S} = a_S] \geq \mathbb{E}_{q^* \leftarrow \pi^* | (q^*_S = q_S)}[\Pr[p_{q^*}|_S = a_S] + \delta_{S,a_S}(q),$$

which implies that

$$1 = \sum_{a_S} \Pr[\mathrm{Sim}_{S,q_S} = a_S] \geq \sum_{a_S} \mathbb{E}_{q^* \leftarrow \pi^* | (q^*_S = q_S)}[\Pr[p_{q^*}|_S = a_S] + \sum_{a_S} \delta_{S,a_S}(q) \geq 1 - 2\epsilon + \sum_{a_S} \delta_{S,a_S}(q).$$

We thus conclude that $\sum_{a_S} \delta_{S,a_S}(q) \leq 2\epsilon$, which in turn implies that $\delta(q) = \sum_{S,a_S} \delta_{S,a_S}(q) \leq 2^{k+1}\epsilon$, as desired. □

Thus, the strategy $\{\tilde{p}_q\}$ satisfies Equations (5.6) and (5.7).

We next define a family of sub-distributions[4] $\{\mathrm{Sim}^{(1)}_{S,q_S}\}$, where $\mathrm{Sim}^{(1)}_{S,q_S}$ is a sub-distribution over $\mathcal{A}_S$, such that for every $S, T \subseteq [k]$ such that $S \subset T$, and every $q \in \mathcal{Q}$ and $a \in \mathcal{A}$,

$$\Pr[\mathrm{Sim}^{(1)}_{T,q_T}|_S = a_S] \leq \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S] \tag{5.8}$$

and for every $q \in \mathrm{GOOD}$,

$$\Pr_{a \leftarrow \mathrm{Sim}^{(1)}_{[k],q}}[V(q,a) = 1] \geq 1 - 2^{3k^2}\epsilon. \tag{5.9}$$

We start by defining $\{\mathrm{Sim}'_{S,q_S}\}$ by

$$\Pr[\mathrm{Sim}'_{S,q_S} = a_S] \triangleq \max_{q^* \in \mathrm{GOOD} | (q^*_S = q_S)} \Pr[\tilde{p}_{q^*}|_S = a_S].$$

[4] A sub-distribution is a distribution where the total probability can be less than 1, but the probability of each event must still be non-negative.

Note that this is a sub-distribution since by Equation (5.6),

$$\Pr[\mathrm{Sim}'_{S,q_S} = a_S] = \max_{q^* \in \mathrm{GOOD}|(q^*_S = q_S)} \Pr[\tilde{p}_{q^*}|_S = a_S] \le \mathbb{E}_{q^* \leftarrow \pi^*|(q^*_S = q_S)} \Pr[p_{q^*}|_S = a_S],$$

which together with the linearity of expectation, implies that indeed

$$\sum_{a_S \in \mathcal{A}_S} \Pr[\mathrm{Sim}'_{S,q_S} = a_S] \le \sum_{a_S \in \mathcal{A}_S} \mathbb{E}_{q^* \leftarrow \pi^*|(q^*_S = q_S)} \Pr[p_{q^*}|_S = a_S] = \mathbb{E}_{q^* \leftarrow \pi^*|(q^*_S = q_S)} \sum_{a_S \in \mathcal{A}_S} \Pr[p_{q^*}|_S = a_S] \le 1.$$

Moreover, Equation (5.7), together with the definition of $\{\mathrm{Sim}'_{S,q_S}\}$, implies that for every $q \in \mathrm{GOOD}$,

$$\Pr_{a \leftarrow \mathrm{Sim}'_{[k],q}} [V(q,a) = 1] \ge 1 - 2^{k+2}\epsilon. \tag{5.10}$$

We next define $\{\mathrm{Sim}^{(1)}_{S,q_S}\}$ by modifying $\{\mathrm{Sim}'_{S,q_S}\}$ in a greedy manner, to ensure that Equation (5.8) is satisfied. This is done by induction starting with sets of size 1. For every set T of size 1, and for every q_T, define

$$\mathrm{Sim}^{(1)}_{T,q_T} \triangleq \mathrm{Sim}'_{T,q_T}.$$

Suppose we defined $\mathrm{Sim}^{(1)}_{S,q_S}$ for all sets S of size less than i. We next define $\mathrm{Sim}^{(1)}_{T,q_T}$ for sets T of size i. To this end, fix any T of size i and fix any q_T. Start by setting

$$\mathrm{Sim}^{(1)}_{T,q_T} = \mathrm{Sim}'_{T,q_T}.$$

For every $S \subset T$ and for every a_S, if

$$\Pr[\mathrm{Sim}^{(1)}_{T,q_T}|_S = a_S] > \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S]$$

then (arbitrarily) reduce the total probability of $\mathrm{Sim}^{(1)}_{T,q_T}$ by exactly

$$\xi_{S,a_S}(T,q_T) \triangleq \Pr[\mathrm{Sim}^{(1)}_{T,q_T}|_S = a_S] - \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S],$$

so that

$$\Pr[\mathrm{Sim}^{(1)}_{T,q_T}|_S = a_S] = \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S]$$

Clearly this ensures that Equation (5.8) holds. We next argue that despite this reduction in probability, Equation (5.9) holds. To this end, note that for every $S, T \subseteq [k]$ such that $S \subset T$, and every $q \in \mathcal{Q}$ and $a \in \mathcal{A}$,

$$\xi_{S,a_S}(T, q_T) \leq \max\left\{0, \Pr[\mathrm{Sim}'_{T,q_T}|_S = a_S] - \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S]\right\}.$$

Define

$$\xi_S(T, q_T) \triangleq \sum_{a_S} \xi_{S,a_S}(T, q_T) \quad \text{and} \quad \xi(T, q_T) \triangleq \sum_{S \subsetneq T} \xi_S(T, q_T).$$

Claim 5.2.4. *For every $T \subseteq [k]$ and every $q \in \mathcal{Q}$*

$$\xi(T, q_T) \leq 2^{2k^2}\epsilon.$$

Note that Claim 5.2.4, together with Equation (5.10), implies that for every $q \in$ GOOD,

$$\Pr_{a \leftarrow \mathrm{Sim}^{(1)}_{[k],q}}[V(q, a) = 1] \geq 1 - 2^{k+2}\epsilon - 2^{2k^2}\epsilon \geq 1 - 2^{3k^2}\epsilon,$$

thus establishing Equation (5.9), as desired.

Proof of Claim 5.2.4. Fix any $T \subseteq [k]$ and any $q \in \mathcal{Q}$. Note that for every $S \subset T$ and for every $a_S \in \mathcal{A}_S$,

$$\begin{aligned}
&\Pr[\mathrm{Sim}'_{T,q_T}|_S = a_S] \\
&= \sum_{a_T : a_T|_S = a_S} \Pr[\mathrm{Sim}'_{T,q_T} = a_T] \\
&= \sum_{a_T : a_T|_S = a_S} \max_{q^* \in \mathrm{GOOD}|q^*_T = q_T} \Pr[\tilde{p}_{q^*}|_T = a_T] \\
&\le \sum_{a_T : a_T|_S = a_S} \mathbb{E}_{q^* \leftarrow \pi^* | q^*|_T = q_T} \Pr[p_{q^*}|_T = a_T] \\
&= \mathbb{E}_{q^* \leftarrow \pi^* | q^*_T = q_T} \Pr[p_{q^*}|_S = a_S] \\
&\le \Pr[\mathrm{Sim}_{S,q_S} = a_S].
\end{aligned}$$

By the definition of Sim'_{S,q_S}, $\mathrm{Sim}^{(1)}_{S,q_S}$, and $\tilde{p}_q$, it holds that

$$\Pr[\mathrm{Sim}_{S,q_S} = a_S] \ge \Pr[\mathrm{Sim}'_{S,q_S} = a_S] \ge \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S].$$

Therefore, the equations above imply that

$$\xi_{S,a_S}(T, q_T) \le \Pr[\mathrm{Sim}_{S,q_S} = a_S] - \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S],$$

which in turn implies that

$$\xi_S(T, q_T) = \sum_{a_S} \xi_{S,a_S}(T, q_T) = 1 - \sum_{a_S} \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S]. \tag{5.11}$$

Note that by definition

$$\sum_{a_S} \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S] = \sum_{a_S} \Pr[\mathrm{Sim}'_{S,q_S} = a_S] - \xi(S, q_S). \tag{5.12}$$

Therefore

$$\xi_S(T, q_T) \le 1 - \sum_{a_S} \Pr[\mathrm{Sim}'_{S,q_S} = a_S] + \xi(S, q_S) \le 2^{k+2}\epsilon + \xi(S, q_S),$$

where the second inequality follows from Equation (5.10). This implies that

$$\xi(T, q_T) \le 2^{|T|} \cdot 2^{k+2}\epsilon + \sum_{S \subsetneq T} \xi(S, q_S). \tag{5.13}$$

We use Equation (5.13), to prove that for every $T \subseteq [k]$ and for every q_T,

$$\xi(T, q_T) \le 2^{2|T|k}\epsilon. \tag{5.14}$$

We prove Equation (5.14) by induction on the size of T, starting from $|T| = 1$. For every T of size 1 and for every q_T, by definition $\xi(T, q_T) = 0$.

Suppose Equation (5.14) holds for every T of size less than i, we prove that it holds for T of size i as follows:

$$\begin{aligned}
\xi(T, q_T) &\le 2^{|T|} \cdot 2^{k+2}\epsilon + \sum_{S \subsetneq T} \xi(S, q_S) \\
&\le 2^{|T|} \cdot 2^{k+2}\epsilon + 2^{|T|} 2^{2(|T|-1)k}\epsilon \\
&\le 2^{2k+2}\epsilon + 2^{|T|} 2^{2|T|k-2k}\epsilon \\
&\le 2^{2k+2}\epsilon + 2^{2|T|k-k}\epsilon \le 2^{2|T|k}\epsilon,
\end{aligned}$$

as desired, where the first inequality follows from Equation (5.13), the second inequality follows from the induction hypothesis, and the other inequalities follow from basic arithmetic.

□

We next modify $\mathrm{Sim}^{(1)}_{S,q_S}$ to ensure that its total probability is independent of q_S. More specifically, we define $\mathrm{Sim}^{(2)}_{S,q_S}$, which is a modification of $\mathrm{Sim}^{(1)}_{S,q_S}$, such that for every $\ell \in [k]$ there exists $\alpha_\ell \in [0, 1]$ such that for every $S \subseteq [k[$ of size ℓ and for every $q_S \in \mathcal{Q}_S$, it holds that

$$\sum_{a_S} \Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S] = \alpha_\ell \tag{5.15}$$

In addition, we still ensure that for every $S, T \subseteq [k]$ such that $S \subset T$, and for every

$q \in \mathcal{Q}$ and $a \in \mathcal{A}$,

$$\Pr[\mathrm{Sim}^{(2)}_{T,q_T}|_S = a_S] \le \Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S] \tag{5.16}$$

and for every $q \in \mathrm{GOOD}$,

$$\Pr_{a \leftarrow \mathrm{Sim}^{(2)}_{[k],q}}[V(q,a) = 1] \ge (1 - 2^{3k^2}\epsilon)^{k+1}\ .^{5} \tag{5.17}$$

To this end, for every $S \subseteq [k]$ and every $q_S \in \mathcal{Q}_S$ let

$$\beta_{q_S} = \sum_{a_S} \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S].$$

We argue that for every $S \subseteq [k]$,

$$\beta_{q_S} \ge 1 - 2^{3k^2}\epsilon, \tag{5.18}$$

as follows:

$$\begin{aligned}
&\sum_{a_S} \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S] \\
&= \sum_{a_S} \max_{q^* \in \mathrm{GOOD}|(q^*_S = q_S)} \Pr[\tilde{p}_{q^*}|_S = a_S] - \xi(S, q_S) \\
&\ge \sum_{a_S} \max_{q^* \in \mathrm{GOOD}|(q^*_S = q_S)} \Pr[\tilde{p}_{q^*}|_S = a_S] - 2^{2k^2}\epsilon \\
&\ge \sum_{a_S} \Pr[\tilde{p}_{q^*}|_S = a_S] - 2^{2k^2}\epsilon \\
&\ge 1 - 2^{k+2}\epsilon - 2^{2k^2}\epsilon \\
&\ge 1 - 2^{3k^2}\epsilon,
\end{aligned}$$

where the first equation follows from Equation (5.12) together with the definition of

[5] Note that by our assumption that $\epsilon < 2^{-4k^2}$, Equation (5.17) implies that for every $q \in \mathrm{GOOD}$,

$$\Pr_{a \leftarrow \mathrm{Sim}^{(2)}_{[k],q}}[V(q,a) = 1] \ge (1 - 2^{-k^2})^{k+1}.$$

Sim'_{S,q_S}, the second equation follows from Claim 5.2.4, the third equation holds for every q^* such that $q^*_S = q_S$, the forth equation follows from Equation (5.7), and the last equation follows from basic arithmetic.

For every $\ell \in [k]$, let

$$\alpha_\ell = (1 - 2^{-3k^2}\epsilon)^\ell.$$

For every $S \subseteq [k]$ of size ℓ, and for every $q_S \in \mathcal{Q}_S$ and $a_S \in \mathcal{A}_S$, define

$$\Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S] \triangleq \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S] \cdot \frac{\alpha_\ell}{\beta_{q_S}}.$$

Note that by definition

$$\sum_{a_S} \Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S] = \alpha_\ell,$$

as desired. Moreover, $\mathrm{Sim}^{(2)}_{S,q_S}$ is a sub-distribution, since

$$\Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S] = \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S] \cdot \frac{\alpha_\ell}{\beta_{q_S}} \leq \Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S],$$

where the first equality follows from the definition of $\mathrm{Sim}^{(2)}_{S,q_S}$ and the last inequality follows from Equation (5.18) together with the the definition of α_ℓ.

We next argue that $\mathrm{Sim}^{(2)}_{S,q_S}$ satisfies Equation (5.16). To this end, fix any $S \subset T \subseteq [k]$ and fix any $q \in \mathcal{Q}$ and $a \in \mathcal{A}$. Note that

$$\begin{aligned}
&\Pr[\mathrm{Sim}^{(2)}_{T,q_T}|_S = a_S] = \\
&\Pr[\mathrm{Sim}^{(1)}_{T,q_T}|_S = a_S] \cdot \frac{\alpha_{|T|}}{\beta_{q_T}} \leq \\
&\Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S] \cdot \frac{\alpha_{|T|}}{\beta_{q_T}} \leq \\
&\Pr[\mathrm{Sim}^{(1)}_{S,q_S} = a_S] \cdot \frac{\alpha_{|S|}}{\beta_{q_S}} = \\
&\Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S],
\end{aligned}$$

as desired, where the first equation follows from the definition of $\mathrm{Sim}^{(2)}_{T,q_T}$, the second equation follows from Equation (5.8), the third equation follows from the definition of α_ℓ together with Equation (5.18), and the last equation follows again from the definition of $\mathrm{Sim}^{(2)}_{S,q_S}$.

Finally, note that for every $q \in \mathrm{GOOD}$,

$$\Pr_{a \leftarrow \mathrm{Sim}^{(2)}_{[k],q}}[V(q,a)=1] = \Pr_{a \leftarrow \mathrm{Sim}^{(1)}_{[k],q}}[V(q,a)=1] \cdot \frac{\alpha_k}{\beta_q} \geq \Pr_{a \leftarrow \mathrm{Sim}^{(1)}_{[k],q}}[V(q,a)=1] \cdot \alpha_k \geq (1-2^{3k^2}\epsilon)^{k+1},$$

as desired, where the first equation follows from the definition of $\mathrm{Sim}^{(2)}_{[k],q}$, the second equation follows from the fact that $\beta_q \leq 1$, and the last inequality follows from Equation (5.9) and from the definition of α_k.

We next modify $\{\mathrm{Sim}^{(2)}_{S,q_S}\}$ to a new family of sub-distributions $\{\mathrm{Sim}^{(3)}_{S,q_S}\}$ that satisfies that for every $q \in \mathcal{Q}$ and every $S \subseteq [k]$,

$$\Pr[\mathrm{Sim}^{(3)}_{S,q_S} = a_S] \geq \sum_{i=1}^{k-|S|} (-1)^{i-1} \sum_{T \supsetneq S, |T|=|S|+i} \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_S = a_S]. \tag{5.19}$$

To this end, we define

$$\Pr[\mathrm{Sim}^{(3)}_{S,q_S} = a_S] \triangleq \frac{1}{k^{2|S|}} \Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S]$$

Claim 5.2.5. $\{\mathrm{Sim}^{(3)}_{S,q_S}\}$ *satisfies Equation* (5.19).

Proof. Fix any $q \in \mathcal{Q}$ and any subset $S \subseteq [k]$. Note that

$$\begin{aligned}
&\sum_{i=1}^{k-|S|}(-1)^{i-1}\sum_{T\supsetneq S,|T|=|S|+i}\Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_S = a_S]\\
&\leq \sum_{i=1}^{k-|S|}\sum_{T\supsetneq S,|T|=|S|+i}\Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_S = a_S]\\
&= \sum_{i=1}^{k-|S|}\sum_{T\supsetneq S,|T|=|S|+i}\frac{1}{k^{2(|S|+i)}}\Pr[\mathrm{Sim}^{(2)}_{T,q_T}|_S = a_S]\\
&\leq \sum_{i=1}^{k-|S|}\sum_{T\supsetneq S,|T|=|S|+i}\frac{1}{k^{2(|S|+i)}}\Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S]\\
&\leq \sum_{i=1}^{k-|S|}\binom{k-|S|}{i}\frac{1}{k^{2(|S|+i)}}\Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S]\\
&\leq \frac{1}{k^{2|S|}}\Pr[\mathrm{Sim}^{(2)}_{S,q_S} = a_S]\cdot\sum_{i=1}^{k-|S|}\binom{k-|S|}{i}\frac{1}{k^{2i}}\\
&\leq \Pr[\mathrm{Sim}^{(3)}_{S,q_S} = a_S]\cdot\sum_{i=1}^{k-|S|}\binom{k-|S|}{i}\frac{1}{k^{2i}}\\
&\leq \Pr[\mathrm{Sim}^{(3)}_{S,q_S} = a_S]
\end{aligned}$$

as desired, where the first equation follows from basic arithmetic, the second equation follows from the definition of $\mathrm{Sim}^{(3)}_{T,q_T}$, the third equation follows from Equation (5.16), the forth and fifth equations follow from basic arithmetic, the six follows from the definition of $\mathrm{Sim}^{(3)}_{T,q_T}$, and the last equation follows from the fact that

$$\sum_{i=1}^{k-|S|}\binom{k-|S|}{i}\frac{1}{k^{2i}} \leq \sum_{i=1}^{k-|S|}k^i\cdot\frac{1}{k^{2i}} = \sum_{i=1}^{k-|S|}\frac{1}{k^i} \leq \sum_{i=1}^{k-|S|}2^{-i} \leq \sum_{i=1}^{\infty}2^{-i} = 1.$$

□

Equation (5.17), together with the definition of $\{\mathrm{Sim}^{(3)}_{S,q_S}\}$ and the assumption that

$\epsilon \leq 2^{-4k^2}$, implies that for every $q \in \text{GOOD}$,

$$\Pr_{a \leftarrow \text{Sim}^{(3)}_{[k],q}}[V(q,a) = 1] \geq 2^{-3k\log k}. \tag{5.20}$$

Equation (5.16), together with the definition of $\{\text{Sim}^{(3)}_{S,q_S}\}$, implies that for every $S, T \subseteq [k]$ such that $S \subsetneq T$, and every $q \in \mathcal{Q}$ and $a \in \mathcal{A}$,

$$\Pr[\text{Sim}^{(3)}_{T,q_T}|_S = a_S] \leq \text{Sim}^{(3)}_{S,q_T|_S} = a_S] \tag{5.21}$$

since

$$\begin{aligned}
\Pr[\text{Sim}^{(3)}_{T,q_T}|_S = a_S] &= \frac{1}{k^{2|T|}} \Pr[\text{Sim}^{(2)}_{T,q_T}|_S = a_S] \\
&\leq \frac{1}{k^{2|T|}} \Pr[\text{Sim}^{(2)}_{S,q_S} = a_S] \\
&\leq \frac{1}{k^{2|S|}} \Pr[\text{Sim}^{(2)}_{S,q_S} = a_S] \\
&= \Pr[\text{Sim}^{(3)}_{S,q_S} = a_S].
\end{aligned}$$

In what follows, we define an honest-referee non-signaling strategy for the game $\mathcal{G}^*$ that convinces V to accept with probability at least $2^{-3k\log k}$. By Theorem 5.2.1 this implies that

$$\mathcal{V}_{\text{NS}}(\mathcal{G}^*) \geq 2^{-O(k^2)} \cdot 2^{-3k\log k} = 2^{-O(k^2)}.$$

By the definition of $\mathcal{G}^*$ (and by Claim 5.2.2), this implies that

$$\mathcal{V}_{\text{NS}}(\mathcal{G}) \geq \frac{1}{2} \cdot 2^{-O(k^2)} \geq 2^{-O(k^2)},$$

as desired.

Therefore, it suffices to define an honest-referee non-signaling strategy for the game $\mathcal{G}^*$ that convinces V to accept with probability at least $2^{-3k\log k}$. We do this in stages.

First, we define a strategy $\{p^{(1)}_q\}$. This strategy is not defined over $\mathcal{A}$ but over $\mathcal{A}^* = \mathcal{A}^*_1 \times \ldots \times \mathcal{A}^*_k$, where for each $i \in [k]$, $\mathcal{A}^*_i \triangleq \mathcal{A}_i \cup \{*\}$. Fix any $q \in \mathcal{Q}$.

For any non-empty subset $S \subseteq [k]$ and every $a_S \in \mathcal{A}_S$, we define

$$\Pr[p_q^{(1)} = (a_S, (*)^{k-|S|})] \triangleq \sum_{i=0}^{k-|S|} (-1)^i \sum_{T \supseteq S, |T|=|S|+i} \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_S = a_S].$$

Equation (5.19) implies that this value is non-negative. Moreover, note that for every $a \in \mathcal{A}$,

$$\Pr[p_q^{(1)} = a] = \Pr[\mathrm{Sim}^{(3)}_{[k],q} = a].$$

We next ensure that $p_q^{(1)}$ is a distribution. To this end, note that

$$\begin{aligned}
&\sum_{a \in \mathcal{A}^*} \Pr[p_q^{(1)} = a] = \\
&\sum_{S \subseteq [k]:|S| \geq 1} \sum_{a_S \in \mathcal{A}_S} \Pr[p_q^{(1)} = (a_S, (*)^{k-|S|})] = \\
&\sum_{S \subseteq [k]:|S| \geq 1} \sum_{a_S \in \mathcal{A}_S} \sum_{i=0}^{k-|S|} (-1)^i \sum_{T \supseteq S, |T|=|S|+i} \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_S = a_S] \leq \\
&\sum_{S \subseteq [k]:|S| \geq 1} \sum_{i=0}^{k-|S|} (-1)^i \sum_{T \supseteq S, |T|=|S|+i} \sum_{a_S \in \mathcal{A}_S} \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_S = a_S]
\end{aligned}$$

Note that $\sum_{a_S \in \mathcal{A}_S} \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_S = a_S] = \sum_{a_T \in \mathcal{A}_T} \Pr[\mathrm{Sim}^{(3)}_{T,q_T} = a_T]$ depends only on T, and is otherwise independent of q_T. This follows from Equation (5.15) and from the definition of $\mathrm{Sim}^{(3)}_{T,q_T}$. Therefore, $\sum_{a \in \mathcal{A}^*} \Pr[p_q^{(1)} = a]$ is independent of q. We denote by

$$\alpha \triangleq \sum_{a \in \mathcal{A}^*} \Pr[p_q^{(1)} = a].$$

If $\alpha > 1$ then we convert $p_q^{(1)}$ to a distribution $p_q^{(2)}$ defined as follows: For every $a \in \mathcal{A}^*$,

$$\Pr[p_q^{(2)} = a] \triangleq \frac{1}{\alpha} \Pr[p_q^{(1)} = a].$$

If $\alpha < 1$ then we convert $p_q^{(1)}$ to a distribution $p_q^{(2)}$ defined as follows: Let

$$\Pr[p_q^{(2)} = (*)^k] \triangleq 1 - \alpha,$$

and for every $a \in \mathcal{A}^* \setminus \{(*)^k\}$ let

$$\Pr[p_q^{(2)} = a] \triangleq \Pr[p_q^{(1)} = a]$$

It is easy to see that in either case, $p_q^{(2)}$ is a distribution.

Claim 5.2.6. $\{p_q^{(2)}\}$ *satisfies the honest referee non-signaling condition.*

Proof. In what follows, we use the following notation: If $p_q^{(1)}$ satisfies $\alpha = \sum_{a^* \in \mathcal{A}^*} \Pr[p_q^{(1)} = a] > 1$ then let $\gamma = \frac{1}{\alpha}$, and otherwise let $\gamma = 1$.

Fix any subset $S \subseteq [k]$. We argue that for every $q, q^* \in \text{GOOD}$ such that $q_S = q^*_S$, and for every $a_S \in \mathcal{A}^*_S$,

$$\Pr[p_q^{(2)}|_S = a_S] = \Pr[p_{q^*}^{(2)}|_S = a_S].$$

Define $S' \subseteq S$ to be the subset for which for every $i \in S'$ it holds that $a_i \in \mathcal{A}$, and for every $i \in S \setminus S'$ it holds that $a_i = *$.

$$
\begin{aligned}
&\Pr[p_q^{(2)}|_S = a_S] = \\
&\sum_{V \subseteq [k] \setminus S} \sum_{a_v \in \mathcal{A}_V} \Pr[p_q^{(2)} = (a_{S'}, a_V, (*)^{k-|S' \cup V|}) = \\
&\sum_{V \subseteq [k] \setminus S} \sum_{a_v \in \mathcal{A}_V} \sum_{i=0}^{k-|S' \cup V|} (-1)^i \sum_{T \supseteq S' \cup V, |T| = |S' \cup V| + i} \gamma \cdot \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_{S' \cup V} = a_{S' \cup V}] = \\
&\sum_{V \subseteq [k] \setminus S} \sum_{i=0}^{k-|S' \cup V|} (-1)^i \sum_{T \supseteq S' \cup V, |T| = |S' \cup V| + i} \gamma \cdot \sum_{a_v \in \mathcal{A}_V} \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_{S' \cup V} = a_{S' \cup V}] = \\
&\sum_{V \subseteq [k] \setminus S} \sum_{i=0}^{k-|S' \cup V|} (-1)^i \sum_{T \supseteq S' \cup V, |T| = |S' \cup V| + i} \gamma \cdot \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_{S'} = a_{S'}] = \\
&\gamma \cdot \sum_{T \supseteq S'} \Pr[\mathrm{Sim}^{(3)}_{T,q_T}|_{S'} = a_{S'}] \cdot \left(\sum_{V \subseteq T \setminus S} (-1)^{|T| - |S' \cup V|} \right)
\end{aligned}
$$

Therefore, to argue that indeed

$$\Pr[p_q^{(2)}|_S = a_S] = \Pr[p_{q^*}^{(2)}|_S = a_S]$$

it suffices to prove that for every $T \supseteq S'$ such that $\ell \triangleq |T \setminus S| \geq 1$, it holds that

$$\sum_{V \subseteq T \setminus S} (-1)^{|T| - |S' \cup V|} = 0,$$

or equivalently that for every such T,

$$\sum_{V \subseteq T \setminus S} (-1)^{|S' \cup V|} = 0.$$

This follows from the following calculation:

$$\sum_{V \subseteq T \setminus S} (-1)^{|S' \cup V|} = (-1)^{|S'|} \cdot \sum_{V \subseteq T \setminus S} (-1)^{|V|} = \sum_{j=0}^{\ell} \binom{\ell}{j} (-1)^j = (1-1)^\ell = 0,$$

as desired. $\square$

$\square$

Bibliography

[1] Leonard Adleman. Two theorems on random polynomial time. In *Foundations of Computer Science, 1978., 19th Annual Symposium on*, pages 75–83. IEEE, 1978.

[2] Dorit Aharonov and Oded Regev. Lattice problems in NP $\cap$ coNP. *Journal of the ACM (JACM)*, 52(5):749–765, 2005.

[3] László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429. ACM, 1985.

[4] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697. ACM, 2016.

[5] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[6] László Babai and Eugene M Luks. Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 171–183. ACM, 1983.

[7] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M Reischuk. Adsnark: Nearly practical and privacy-preserving proofs on authenticated data. In *2015 IEEE Symposium on Security and Privacy*, pages 271–286. IEEE, 2015.

[8] Michael Backes, Dario Fiore, and Raphael M Reischuk. Verifiable delegation of computation on outsourced data. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 863–874. ACM, 2013.

[9] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 709–721, 2018.

[10] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Efficient identification schemes using two prover interactive proofs. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 498–506, 1989.

[11] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Annual Cryptology Conference*, pages 90–108. Springer, 2013.

[12] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 781–796, 2014.

[13] Ingrid Biehl, Bernd Meyer, and Susanne Wetzel. Ensuring the integrity of agent-based computations by short proofs. In *Mobile Agents, Second International Workshop, MA'98, Stuttgart, Germany, September 1998, Proceedings*, pages 183–194, 1998.

[14] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016.

[15] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482, 2017.

[16] Benjamin Braun, Ariel J Feldman, Zuocheng Ren, Srinath Setty, Andrew J Blumberg, and Michael Walfish. Verifying computations with state. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 341–357. ACM, 2013.

[17] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Computational Complexity, 1998. Proceedings. Thirteenth Annual IEEE Conference on*, pages 8–12. IEEE, 1998.

[18] Jin-yi Cai, Anne Condon, and Richard J. Lipton. PSPACE is provable by two provers in one round. *J. Comput. Syst. Sci.*, 48(1):183–193, 1994.

[19] John J Cannon. Construction of defining relators for finite groups. *Discrete Mathematics*, 5(2):105–129, 1973.

[20] Marco L Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 261–270. ACM, 2016.

[21] Timothy M Chan. More logarithmic-factor speedups for 3sum,(median,+)-convolution, and some geometric 3sum-hard problems. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 881–897. SIAM, 2018.

[22] Alessandro Chiesa, Eran Tromer, and Madars Virza. Cluster computing in zero knowledge. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 371–403. Springer, 2015.

[23] Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*, pages 236–249, 2004.

[24] Michael Cohen, Yin-Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *CoRR*, abs/1810.07896, 2018.

[25] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112. ACM, 2012.

[26] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE, 2015.

[27] Mark De Berg, Marko M de Groot, and Mark H Overmars. Perfect binary space partitions. *Computational geometry*, 7(1-2):81–91, 1997.

[28] David P. Dobkin, Richard J. Lipton, and Steven P. Reiss. Linear programming is log-space hard for P. *Inf. Process. Lett.*, 8(2):96–97, 1979.

[29] Uriel Feige and Joe Kilian. Making games short (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 506–516, 1997.

[30] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 844–855. ACM, 2014.

[31] Anka Gajentaan and Mark H Overmars. On a class of o(nˆ2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[32] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 18, page 136, 2011.

[33] Michel Goemans. 18.415/6.854 advanced algorithms lecture 11, 2008. URL:`https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-854j-advanced-algorithms-fall-2008/lecture-notes/lec11.pdf`.

[34] Michel Goemans, Shafi Goldwasser, and Dhiraj Holden. Doubly-efficient pseudo-deterministic proofs. *arXiv preprint arXiv:1910.00994*, 2019.

[35] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17. Springer Science & Business Media, 1998.

[36] Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 127–138. ACM, 2013.

[37] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[38] Oded Goldreich and Guy N Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[39] Shafi Goldwasser and Ofer Grossman. Perfect bipartite matching in pseudo-deterministic RNC. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 208, 2015.

[40] Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. Pseudo-deterministic proofs. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[41] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[42] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

[43] Joachim Grollmann and Alan L Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.

[44] Ofer Grossman. Finding primitive roots pseudo-deterministically. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 207, 2015.

[45] Lane A Hemaspaandra, Ashish V Naik, Mitsunori Ogihara, and Alan L Selman. Computing solutions uniquely collapses the polynomial hierarchy. *SIAM Journal on Computing*, 25(4):697–708, 1996.

[46] Dhiraj Holden. A note on unconditional subexponential-time pseudo-deterministic algorithms for BPP search problems. *arXiv preprint arXiv:1707.05808*, 2017.

[47] Dhiraj Holden and Yael Tauman Kalai. Non-signaling proofs with o (log n) provers are in pspace. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1024–1037, 2020.

[48] Justin Holmgren and Lisa Yang. Characterizing parallel repetition of non-signaling games: Counterexamples and a dichotomy theorem. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:121, 2018.

[49] Justin Holmgren and Lisa Yang. The parallel repetition of non-signaling games: counterexamples and dichotomy. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 185–192. ACM, 2019.

[50] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on computing*, 17(5):935–938, 1988.

[51] Tsuyoshi Ito. Polynomial-space approximation of no-signaling provers. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, pages 140–151, 2010.

[52] Tsuyoshi Ito. Polynomial-space approximation of no-signaling provers. In *International Colloquium on Automata, Languages, and Programming*, pages 140–151. Springer, 2010.

[53] Tsuyoshi Ito, Hirotada Kobayashi, and Keiji Matsumoto. Oracularization and two-prover one-round interactive proofs against nonlocal strategies. In *IEEE Conference on Computational Complexity*, pages 217–228, 2009.

[54] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 91–118, 2016.

[55] Yael Tauman Kalai, Ran Raz, and Oded Regev. On the space complexity of linear programming with preprocessing. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 293–300, 2016.

[56] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 565–574, 2013.

[57] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 485–494, 2014.

[58] Leonid A. Khalfin and Boris S. Tsirelson. Quantum and quasi-classical analogs of Bell inequalities. In *In Symposium on the Foundations of Modern Physics*, pages 441–460, 1985.

[59] Ahmed E Kosba, Dimitrios Papadopoulos, Charalampos Papamanthou, Mahmoud F Sayed, Elaine Shi, and Nikos Triandopoulos. {TRUESET}: Faster verifiable set computations. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 765–780, 2014.

[60] Cecilia Lancien and Andreas Winter. Parallel repetition and concentration for (sub-)no-signalling games via a flexible constrained de finetti reduction. *CoRR*, abs/1506.07002, 2015.

[61] Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 448–457, 1993.

[62] Rudolf Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–136, 1979.

[63] Jiri Matousek and Bernd Gärtner. *Understanding and using linear programming.* Springer Science & Business Media, 2007.

[64] Peter Bro Miltersen, N Variyam Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *International Computing and Combinatorics Conference*, pages 210–220. Springer, 1999.

[65] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.

[66] Igor C Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. *arXiv preprint arXiv:1612.01817*, 2016.

[67] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.

[68] Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1065–1075. SIAM, 2010.

[69] Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994.

[70] Peter Rastall. Locality, Bell's theorem, and quantum mechanics. *Foundations of Physics*, 15(9):963–972, 1985.

[71] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 49–62. ACM, 2016.

[72] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[73] Maria J. Serna. Approximating linear programming is log-space complete for P. *Inf. Process. Lett.*, 37(4):233–236, 1991.

[74] Srinath Setty, Benjamin Braun, Victor Vu, Andrew J Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 71–84. ACM, 2013.

[75] Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pages 253–268, 2012.

[76] Srinath TV Setty, Richard McPherson, Andrew J Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *NDSS*, volume 1, page 17, 2012.

[77] Adi Shamir. IP=PSPACE. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.

[78] Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.

[79] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO (2)*, pages 71–89, 2013.

[80] Justin Thaler, Mike Roberts, Michael Mitzenmacher, and Hanspeter Pfister. Verifiable computation with massively parallel interactive proofs. In *HotCloud*, 2012.

[81] Pravin Vaidya. Speeding-up linear programming using fast matrix multiplication. In *FOCS*, 1989.

[82] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[83] Victor Vu, Srinath Setty, Andrew J Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 223–237. IEEE, 2013.

[84] Riad S Wahby, Srinath TV Setty, Zuocheng Ren, Andrew J Blumberg, and Michael Walfish. Efficient ram and control flow in verifiable outsourced computation. In *NDSS*, 2015.

[85] Michael Walfish and Andrew J. Blumberg. Verifying computations without re-executing them. *Commun. ACM*, 58(2):74–84, 2015.

[86] Ryan Williams. Faster decision of first-order graph properties. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 80. ACM, 2014.

[87] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 645–654. IEEE, 2010.

[88] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing*, 42(3):831–854, 2013.

[89] Neal E Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings 42nd IEEE symposium on foundations of computer science*, pages 538–546. IEEE, 2001.